



THÈSE

En vue de l'obtention du DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par
Jean-Luc HAK

Le 18 juillet 2019

**Engineering annotations for supporting the design process of
interactive systems: A model-based approach and a tool suite**

Ingénierie des annotations pour le support de processus de conception
de systèmes interactifs: une approche basée modèle et outillée

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et
Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Philippe PALANQUE et Marco WINCKLER

Jury

Mme Kathia MARCAL DE OLIVEIRA, Rapporteur
M. Jean VANDERDONCKT, Rapporteur
Mme Regina BERNHAUPT, Présidente du jury
M. Philippe PALANQUE, Co-directeur de thèse
M. Marco WINCKLER, Co-directeur de thèse

Table of content

Table of content	ii
Acknowledgements	ix
Résumé de la thèse	xi
Abstract	xii
Chapter 1. Introduction	1
Chapter 2. State of the art of development process	7
2.1. Introduction to development process for interactive systems	7
2.2. Overview of the development processes	8
2.2.1. Waterfall process (Royce, W. W., 1970)	8
2.2.2. V-model (McDermid & Ripken, 1983)	9
2.2.3. The Nabla model (Kolski, 1998)	10
2.2.4. Spiral model (Boehm, 1986)	11
2.2.5. Rational Unified Process (Kruchten, 2004)	11
2.2.6. AGILES methods (Cockburn, 2002)	12
2.2.7. Star model (Hartson, 1989)	14
2.2.8. The layered development process model (Curtis, 1994)	15
2.2.9. The Object-Oriented User Interface (OOUI) design process (Collins, 1995)	15
2.2.10. The Iterative-cyclic process (Rauterberg, 1992)	15
2.2.11. User-centered System Design (Gulliksen, 2003)	16
2.2.12. The Usage-Centered Design (Constantine & Lockwood, 2002)	18
2.2.13. The ISO User-Centered Design (UCD) process	19
2.2.13.1 Presentation of the tasks of the UCD	20
2.3. Synthetic analysis of the development processes	21
2.4. Conclusion	22
Chapter 3. Analysis of annotations	25
3.1. Introduction	25
3.2. Nature of annotations	25
3.3. Annotations classes and dimensions	27
3.4. Studies of annotations in text documents	28
3.4.1. Presentation of the roles involved in the design process	28
3.4.2. Uses of annotations	29
3.4.3. Synthesis of annotations definitions	33
3.4.4. The case of freeform annotations	34
3.5. Other uses of annotations on digital artefacts	35
3.5.1. The different types of digital annotations	35

3.5.2. Findings on the use of annotation for the UCD process	36
3.6. W3C Web Annotation Data Model.....	37
3.6.1. The Core of the annotation	37
3.6.2. The Metadata of the annotation.....	38
3.6.3. The Goal of the annotation	38
3.7. Conclusions.....	38
Chapter 4. Overview of tools supporting annotations and of prototyping tools.....	41
4.1. Introduction.....	41
4.2. Review of annotation tools	42
4.2.1. Selection of tools	42
4.2.2. Criteria used for the analysis of annotations tools	42
4.2.3. Results of the review of annotation support	43
4.2.3.1 Targeting support	43
4.2.3.2 Temporal evolution of artefacts.....	43
4.2.3.3 Semantic of annotations	44
4.2.3.4 Annotation management	44
4.2.3.5 Collaborative support.....	44
4.3. Review of prototyping tools	45
4.3.1. Selection of tools	45
4.3.2. Criteria used for analyzing prototyping tools.....	46
4.3.3. Source of information collected	46
4.3.4. Results of the review of prototyping tools.....	46
4.3.4.1 Features provided by prototyping tools for the specification and the design.....	46
4.3.4.2 Version control features provided by prototyping tools.....	47
4.3.5. Results of the review of annotations support in prototyping tools	47
4.3.5.1 Creation of annotations.....	47
4.3.5.2 Targeting support	48
4.3.5.3 Temporal evolution of artefacts.....	48
4.3.5.4 Semantic of annotations	48
4.3.5.5 Annotation management	49
4.3.5.6 Collaborative support.....	49
4.3.6. Findings on prototyping tools.....	50
4.3.6.1 The specification of the prototype is limited to the UI	50
4.3.6.2 Uses of annotations in prototyping tools.....	50
4.3.6.3 Targeting support in prototyping tools are limited to their tools	50
4.3.6.4 The three different period of the prototyping tool history.....	51

4.3.6.5 Analysis of the adoption of features by prototyping tools	51
4.3.6.6 Prototyping tools on smartphone	52
4.4. Other findings concerning both annotating and prototyping tools	52
4.4.1. Tasks for supporting annotations.....	52
4.4.2. Creation of annotations.....	53
4.4.3. Navigation between the annotation and the target	53
4.4.4. Dependence of the annotation on its artefact without acknowledging its evolution	53
4.4.5. Lack of support for the evolution of the context of the annotation	53
4.4.6. A few features dedicated to the management of the lifecycle of annotations	54
4.4.7. No support for documenting a custom semantic for annotations.....	54
4.4.8. Limits of annotations support for an environment that includes many types of artefacts	54
4.4.9. Limits of prototyping tools for the support of the traceability of the design	55
4.5. Conclusions.....	57
Chapter 5. Management and usage of annotations in the UCD process.....	59
5.1. Introduction.....	59
5.2. Artefacts and annotations lifecycle	59
5.2.1. Lifecycle of design artefact within the design process.....	59
5.2.2. Lifecycle of annotation within the design process	60
5.2.3. Relationship between the annotations and design artefacts: a reciprocal influence and repercussions.....	61
5.3. Usage of annotations during the UCD process	63
5.3.1. Annotations for connecting models of the interactive system	64
5.3.1.1 Usage of the targeting properties of annotations to connect models.....	64
5.3.1.2 Necessity to formalize artefacts within the project workspace	64
5.3.2. Annotations for assisting stakeholders in their activities	65
5.4. Management of the relationship of an annotation and its targets.....	65
5.4.1. Management of the targets based on the relevance of the annotation	65
5.4.2. Targeting support of versioned files and versioning of annotations	66
5.5. Conclusion	68
Chapter 6. Model-based approach for describing annotations	69
6.1. Introduction.....	69
6.2. Towards an Annotation Models for Artefacts used to build interactive systems.....	70
6.2.1. Features inherited from the Web Annotation Data Model	70
6.2.2. Specificities of the annotations on artefacts of the design of interactive systems	70
6.2.3. Description of the annotation model	72
6.3. Operationalization of the annotation model	75

6.3.1. DTD and XML files.....	75
6.3.2. Mapping between annotations and artefacts.....	80
6.4. Extensibility of the model.....	83
6.5. Conclusion	84
Chapter 7. An architecture for integrating annotations made on diverse artefacts	87
7.1. Introduction.....	87
7.2. Inner of the architecture	87
7.3. Presentation of the file repository for managing annotations of the project	89
7.4. Architecture of the annotation plug-in	90
7.4.1. Integration of the plugin for its support in a new editor and its artefacts	92
7.4.2. Current status of the integration of the plugin in editors and their artefacts	96
7.5. ARMADILLO: An annotation tool support for managing annotations of a project	96
7.5.1. Description of the ARMADILLO tool support	96
7.5.2. Architecture and extensibility of the ARMADILLO tool support	97
7.6. Conclusion	99
Chapter 8. The PANDA ecosystem	101
8.1. Introduction.....	101
8.2. The PANDA prototyping tool.....	101
8.2.1. Overview of PANDA.....	101
8.2.2. Integration of the annotations plugin in PANDA.....	102
8.3. Case study.....	104
8.3.1. Informal presentation of the case study	104
8.3.2. Automated test of prototypes with a scenario	104
8.4. Conclusions.....	108
Chapter 9. Integrated view of annotations on CIRCUS	109
9.1. Introduction.....	109
9.2. The different models and design artefacts representing the interactive system	110
9.2.1. Annotation using ARMADILLO.....	110
9.2.2. Prototypes using PANDA	110
9.2.3. Task models using HAMSTERS.....	111
9.2.4. Dialog models using ICO	111
9.3. Case study.....	111
9.3.1. Informal presentation of the case study	111
9.3.2. The various artefacts of the WXR project	112
9.3.3. Annotation of the prototype artefact of the case study	115
9.3.4. Use of ARMADILLO tool to manage annotations at project level	116

9.4. Conclusions.....	118
Chapter 10. Industrial transfer over eazly™	121
10.1. Introduction.....	121
10.2. Overview of the e-Citiz™ framework.....	121
10.3. The “eazly™” project.....	122
10.3.1. Overview of eazly™	122
10.3.1.1 The evolution of the design process	122
10.3.2. Using of eazly™ as a prototyping tool	127
10.3.2.1 Users of the eazly Studio™	128
10.3.2.2 The different models of the eazly™ framework.....	128
10.3.3. Annotation support for collaborating over the design	129
10.3.3.1 Motivations for adding annotations support in the eazly Studio™	129
10.3.3.2 Early prototypes of the annotation support in the eazly Studio™	129
10.3.3.3 Integration of the annotations in the eazly Studio™	133
10.3.4. Versioning of the evolutions to ensure the integrity of the application.....	135
10.3.4.1 Challenges imposed by the e-Citiz™ design process with eazly™ and its double iterations loop	135
10.3.4.2 The proposed solutions to tackle the issues of the design choices made for eazly™	136
10.3.5. Traceability of the evolutions of the design through the iterations	137
10.4. Conclusions.....	137
Chapter 11. Conclusions and Future Work	141
List of publications.....	145
List of figures	147
List of table	151
Annex 1. List of annotation tools examined for the state of the art.....	153
Annex 2. List of prototyping tools examined for the state of the art.....	155
Annex 3. Synthesis of the analysis of annotation support	157
Annex 4. XSD of the annotation file	159
Annex 5. XSD of the PANDA dialog file	167
Annex 6. XSD of the PANDA presentation file.....	169
Annex 7. Presentation of the PANDA ecosystem	171
7.1. Introduction.....	171
7.2. User interface models described in the PANDA ecosystem.....	171
7.3. The PANDA ecosystem: a framework for supporting traceability of the design process ...	173
7.3.1. Presentation of the framework.....	173

7.3.2.	A modular approach.....	174
7.3.3.	Modules of the PANDA ecosystem.....	174
7.3.4.	Combining the modules	177
7.3.5.	Integration of the modules of the PANDA ecosystem within the UCD process.....	180
7.4.	Creating and previewing a PANDA prototype	182
7.4.1.	Creating a prototype using PANDA	182
7.4.2.	Previewing the PANDA prototype	186
Annex 8.	Presentation of the e-Citiz™ framework.....	191
8.1.	Introduction.....	191
8.2.	Overview of the e-Citiz™ framework	191
8.2.1.	The actors of the e-Citiz™ framework	192
8.2.2.	The iterative design process for the design of an e-Citiz™ application	192
8.2.3.	The development of the e-Citiz™ application.....	192
8.3.	The different models of the application.....	195
8.3.1.	The e-Citiz™ model	195
8.3.2.	The user data structure	195
8.4.	Conclusions on the e-Citiz™ framework	196
8.4.1.	The lack of the e-Citiz™ framework	196
8.4.2.	The “Process 2.0” project	197
8.4.3.	Toward the “eazly™” project	200
Bibliography.....		203

Acknowledgements

I would like to thanks Pr Káthia Marçal de Oliveira and Pr Jean Vanderdonckt for accepting to review my PhD dissertation.

Thanks to Softeam for funding this PhD thesis and for allowing me to manage my work schedule with the IRIT with such flexibility.

Thanks to Marco for accepting me as his PhD student as well as his support and his availability throughout this PhD thesis despite all his works in parallel.

Thanks to Olivier, Cyril, Nicolas, and Laurent for supervising me during the development of eazly as well as all the R&D team, Frédéric, and Jérôme for welcoming me.

Thanks to Eric for his help during the successive developments of PANDA and ARMADILLO and thanks to Cyril, Laurent, Ali, Monia, Nicolas and Abdennacer for helping me develop eazly.

Thanks to everyone within the ICS team for their help and for keeping a good working atmosphere both during and after working hour.

Lastly, I would like to thank all my family for their support through all these years of study and for the years to come.

Résumé de la thèse

Au cours d'un processus de développement de système interactifs, différents acteurs collaborent lors des activités de ce processus et plusieurs choix de conceptions sont effectués afin de converger vers une solution répondant à la fois aux besoins des utilisateurs et aux exigences. Pour atteindre cette solution, de nombreux artefacts sont produits, utilisés et révisés par les différents intervenants du processus. Afin de communiquer sur des points particuliers d'un artefact, collaborer dans son élaboration ou tout simplement rajouter des informations complémentaires, des annotations peuvent être créés sur ces artefacts. En fonction des annotations et de leurs contenus, certains artefacts peuvent par la suite être amenés à évoluer, reflétant ainsi l'influence des annotations sur ces artefacts et donc leurs influences sur le projet de manière globale. Il est donc possible de considérer les annotations comme un outil versatile jouant un rôle non négligeable dans le processus de conception.

Néanmoins, plusieurs problèmes peuvent être identifiés concernant l'intégration des annotations au sein des activités d'un processus de conception de système interactifs. Premièrement, le rôle des annotations n'est pas clairement défini dans les différents processus de conceptions. En effet, bien qu'on observe l'usage omniprésent des annotations lors de la conception de systèmes interactif, les processus de conception actuels n'expliquent pas comment les relier aux tâches à accomplir et les artefacts à produire. Deuxièmement, une annotation peut concerner plusieurs artefacts car chacun modélise des points de vue complémentaires du système interactif. Néanmoins, la multiplicité des types d'artefacts et des outils pour la création de ces artefacts pose un problème car chaque outil qui offre la possibilité de créer des annotations propose son propre modèle d'annotation. Ce modèle est généralement restreint à un type d'artefact donné : celui manipulé par l'outil. Ceci implique que les annotations d'un projet sont éparpillées par lot et que chaque lot d'annotations est fermé à un seul type d'artefact.

Cette thèse s'appuie sur une analyse des annotations et des pratiques liées aux annotations ainsi que sur la recommandation "Web Annotation Data Model" du W3C pour proposer un modèle d'annotation et une architecture logicielle permettant de centraliser les annotations d'un projet et d'intégrer ces annotations dans divers types d'outils et d'artefacts. Ce modèle d'annotation et cette architecture logicielle a été appliquée dans trois études de cas différents afin d'explorer différentes intégrations possibles au sein d'un processus de conception. La première étude de cas démontre l'intégration et la personnalisation d'annotations au sein d'un outil de prototypage. La seconde étude de cas s'attarde sur la présentation d'un outil permettant de consulter dans une vue unique l'ensemble des annotations créés sur différents artefacts et sur les différents modèles d'un projet. La troisième étude de cas illustre une intégration des annotations dans un environnement industriel comprenant des outils et un processus de conception existant.

Ainsi, ces contributions autour des annotations servent de base pour la réalisation de travaux complémentaires tels que l'utilisation d'annotations pour structurer et connecter les différents modèles d'un système interactif, l'utilisation d'annotations en tant que ressource pour les processus de prises de décisions, et l'utilisation d'annotations pour étudier la traçabilité de l'évolution d'un système interactif. En effet, en reliant les artefacts entre eux en utilisant les annotations et en justifiant les choix de conceptions avec des annotations, il serait possible d'assurer la traçabilité des différents choix de design effectués au cours d'un projet ainsi que la traçabilité de l'impact de ces différents choix sur les artefacts.

MOTS-CLES : Processus de conception de système interactifs, Annotations, Techniques de prototypage, Conception Centrée Utilisateur, Gestion et évolution des prototypes, Outil de support à la conception de système interactifs

Abstract

During the development process of an interactive system, different actors collaborate in the activities of this process and several design choices are made to converge to a solution that meets both user needs and requirements. To achieve this solution, many artifacts are produced, used and reviewed by the various stakeholders of the process. In order to communicate on particular points of an artifact, to collaborate in its elaboration or simply to add additional information, annotations can be created on these artifacts. Depending on the annotations and their contents, some artefacts may subsequently evolve, thus reflecting the influence of annotations on these artifacts and therefore reflecting their influence on the project. Thus, it is possible to consider annotations as a versatile tool playing a significant role in the design process.

Nevertheless, several issues can be identified regarding the integration of annotations within the activities of the design process of interactive systems. First, the role of annotations is not clearly defined in the different design processes. While there is a widespread and a ubiquitous use of annotations in the design of interactive systems, current design processes do not address how to relate them to the tasks to be performed and the artifacts to be produced. Secondly, an annotation can be related to several artifacts as each models are giving a complementary representation of the interactive system. However, the multiplicity of artifact types and tools for creating these artifacts is a problem since each tool that provide features for annotations implements their own annotation model. These models are usually restricted to one type of artifact: the one handled by the tool. This implies that the annotations produced within a project are scattered by sets and that each these annotation set is closed to a single type of artifact.

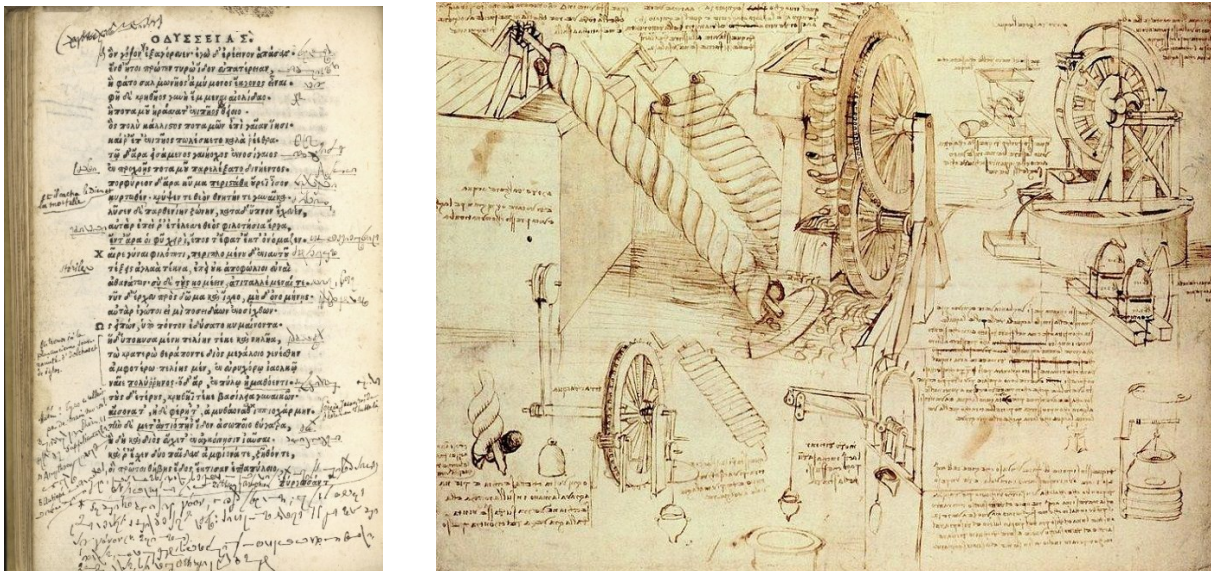
This PhD thesis is based on an analysis of annotations and their uses as well as on the W3C “Web Annotation Data Model” recommendation to propose an annotation model and an architecture to centralize the annotations of a project. This architecture also allows to include the annotations support on various tools and type of artifacts. This contribution has been applied on three different case studies to explore the possible integrations of annotations within a design process. The first case study demonstrates the integration and customization of annotations within a prototyping tool. The second case study focuses on the presentation of a tool allowing to consult in a single view all the annotations created on different artefacts and on different models of a project. The third case study illustrates an integration of annotations into an industrial environment that includes existing tools and an existing design process.

Thus, these contributions around annotations are used as a basis for the realization of complementary works such as the use of annotations to structure and connect the different models of an interactive system, the use of annotations as a resource for the decisions making processes, and the use of annotations to study the traceability of the evolution of an interactive system. Indeed, by linking the artifacts to each other using annotations and justifying the choice of designs with annotations, it would be possible to ensure the traceability of the different design choices made during a project as well as the traceability of the impact of these different choices on the artifacts.

KEYWORDS: Design of interactive systems, Annotations, Prototyping techniques, User Centered Design, Management and evolution of prototypes, Computer aided user interface design

Chapter 1. Introduction

Annotations always played an important role in human History as a mean to add complementary information to documents. Since late Antiquity, annotations in the margin of documents are testimonies to a critical and scientific way of dealing with texts: versions are compared, passages that seem to be corrupted are marked, contrasting opinions are highlighted, confronted and discussed [1]. Figure 1-1 shows two Early Renaissance documents containing annotations: a) annotations left by an anonymous scholar (a student, a school teacher or a professional translator) on an edition of Homer's *Odyssey* printed in 1504 (probably) for studying and preparing a future translation of the text [2]; and, b) annotations left by Leonardo da Vinci to explain the design of his water-lifting machine (1481) [3]. As we shall see, annotations might be written by the creator of the document at the time of writing or added later on by other people. Annotations have been demonstrated useful tools for communication; they can be used to explain an idea and also collect feedback from the readers. For that they are a commonplace in text documents¹. It is interesting to notice that annotations might contain information aimed to correct and/or improve a piece of work.



a) Annotated 1504 edition of Homer's *Odyssey*

b) Leonard da Vinci on "Water Lifting Devices" (1481)

Figure 1-1 Examples in Early Renaissance documents showing the use of annotations for explain a design/idea

However, annotations are not an exclusivity of text documents, we might find annotations associated to other types of documents including specifications used for building interactive systems [4; 5; 6]. Indeed, annotations are frequently found as a mean to complete the description of mockup prototypes with information that could not be specified otherwise due to the inner nature of the prototype itself. Figure 1-2 illustrates a paper-based mock-up featuring annotations to describe the nature of the graphical components and the expected behavior (dialog between windows) for a mini-game prototype.

¹ It is very likely that some readers of this document will make use of annotations to comment it

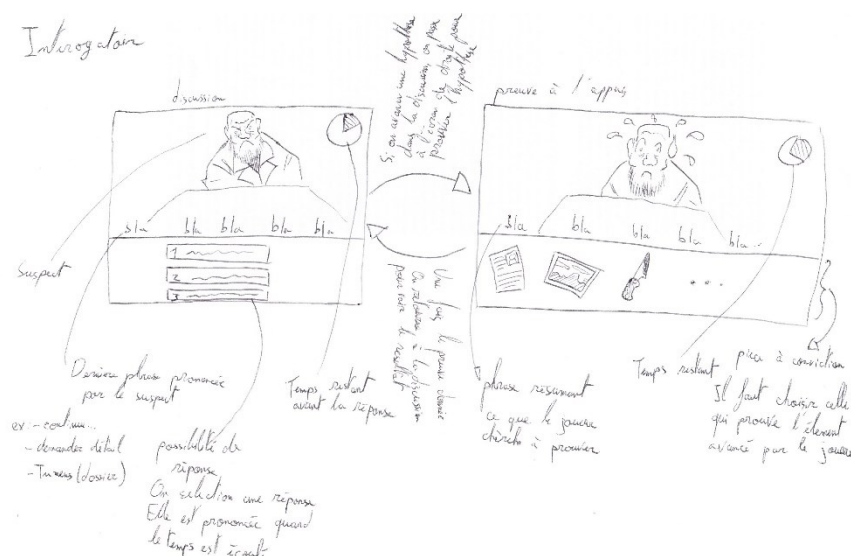


Figure 1-2 Prototype of a mini-game

Figure 1-3 illustrates a few uses of annotations on prototypes of interactive systems, including explaining the design (Figure 1-3.a), associating data available from other sources (in the example results of a usability evaluation featuring a heat map generated by eye-tracking) (Figure 1-3.b), and record design decisions such as opinions/preferences and instructions on how to improve the design (Figure 1-3.c). As we shall see, annotations might assume many forms (such as text, sketching, highlighting, etc.) making this a suitable mechanism to provide complimentary information with multiple uses.

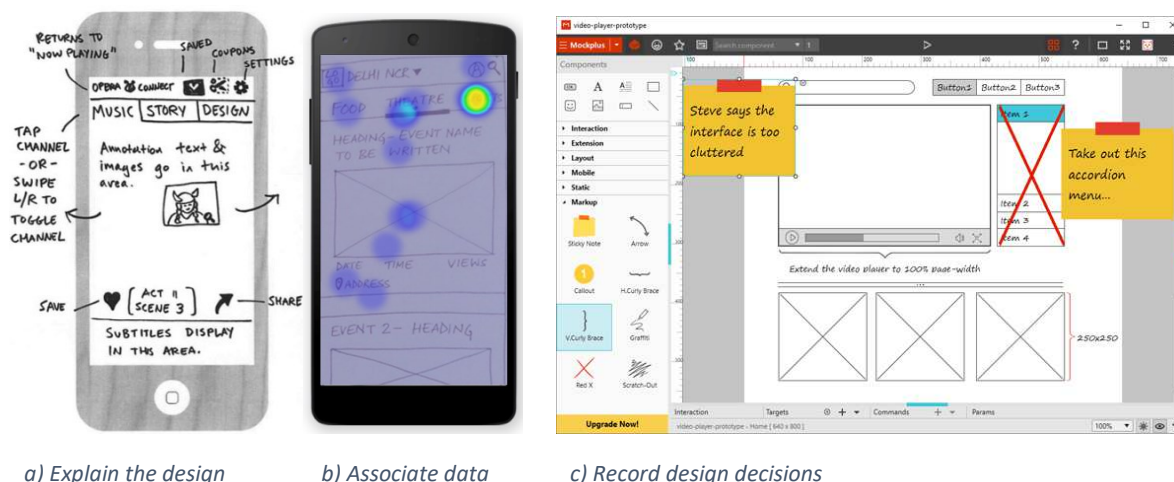


Figure 1-3 Three example of the use of annotations on prototype

Empirical observations have demonstrated that development teams often make an extensive use of annotations as a communication support. The study performed by Gutierrez et al. [5] pointed out that annotations are used by members of development teams to: record the results of discussion including decisions and upcoming tasks, communicate and inform other team members of the work done, gather internal and external feedback on artefacts stored in the workspace, conduct usability evaluations by documenting information and by recording conversation between design teams and UX experts, justify design choices, and document the design choices by describing them retrospectively.

Whilst annotations might be available in many integrated development environment (IDE), the study about the use of annotations during the development process of interactive systems is quite recent and many research question are still pending for example:

- What is the current state of the knowledge about tools supporting the use of annotations for building interactive systems?
- What are the attributes that make annotations essential for the design of interactive systems?
- How annotations are connected to the diverse artefacts produced along the development process?
- In which phases of the development process should annotations be employed?
- How annotations evolve along the phases of the development process?

In this PhD thesis, we investigate the role played by annotations during the development process of interactive systems. Given the associative nature of annotations, it is quite natural to consider annotations as a possible design solution to the problem of tracing design decisions to artefacts. Our ultimate goals with this work on annotations are:

- i) To investigate a systematic strategy for connecting ideas and design decisions to the multiple artefacts used to build interactive systems,
- ii) To promote the development of tools helping the development team to have a coherent view of the decisions made along the development process of interactive system.

The foundations of our proposal follow some premises:

- Annotations are formalized otherwise we cannot build tools to handle them in computer-aided design tools for building user interfaces;
- Annotations are considered as an artefact on its own right. We use the term artefact to refer pieces of work that contribute to the development of an interactive system, for example prototypes, models, specifications, etc.
- Annotations are dynamic and subject to change/evolve along the development process in iterative cycles.

We suggest that annotations can be connected to diverse type of artefacts. However, this PhD thesis focus on annotations made over user interface prototypes. In order to illustrate the connectivity of annotations to other artefacts, we extend the study to task models and dialog models and we show how annotations can be used to create a coherent view. We assume that the contents of annotations might affect the evolution of the prototypes and artefacts used to build interactive systems. If so, we should be able to trace design decisions by following the evolution of artefacts and the annotations connected to them. The scientific questions addressed in the PhD thesis are also relevant for the industry and have many practical applications. This manuscript is organized in eleven chapters as illustrated in the Figure 1-4.

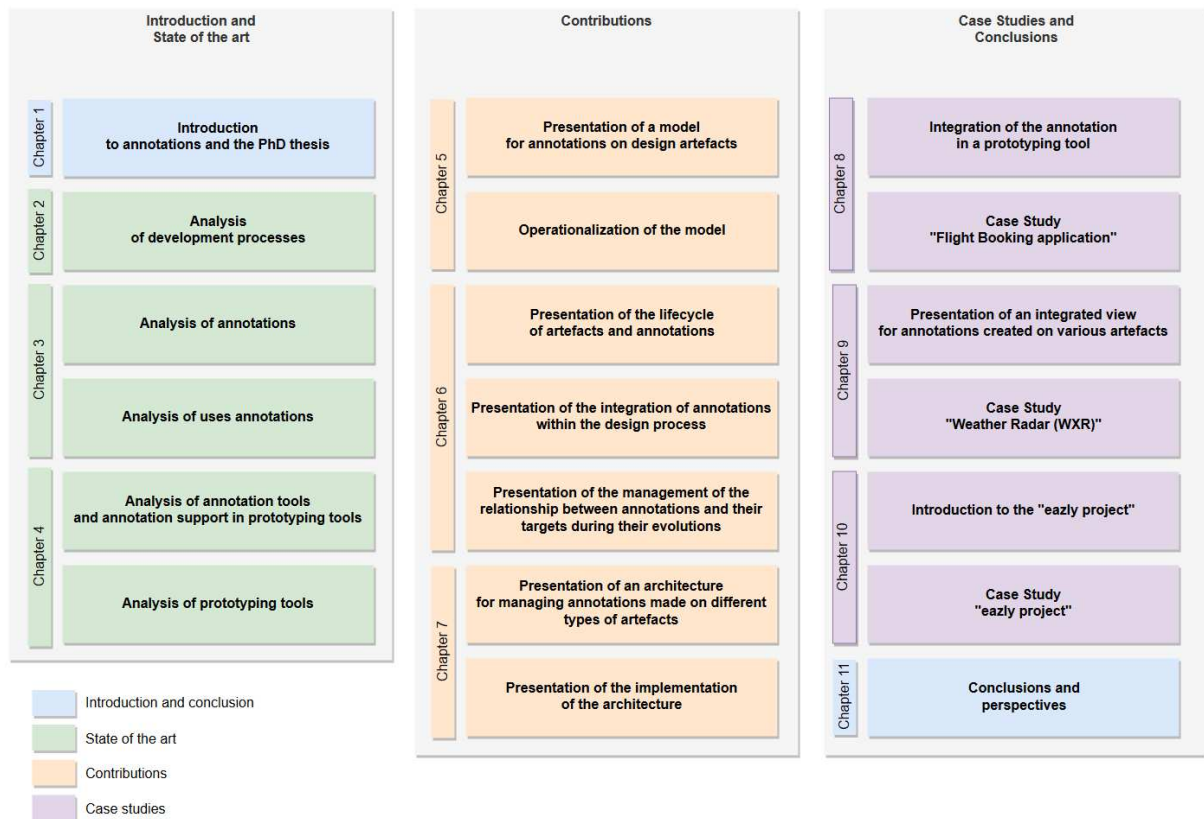


Figure 1-4 Chapters of the PhD thesis

The **Chapter 1** refers to this introduction and the **Chapter 11** presents the conclusions and future work. The other chapters are organized in three main parts, as follows:

STATE OF THE ART

The **Chapter 2** presents an overview of the development processes for interactive systems.

The **Chapter 3** revises the literature on annotations, starting with early works annotations on text documents and concluding with the most recent attempts of the W3C to create a standard for annotations of electronic documents.

The **Chapter 4** presents a comprehensive analysis of existing tools supporting annotations and prototyping activities.

CONTRIBUTIONS

The **Chapter 5** specifies the life cycle of annotations within a user-centered design process. This chapter proposes a micro process (i.e. a process that can be run as a small activity inside a development process) for the management of individual annotations (including creation, publication, validation, updating, disposal, and archival) and the co-evolution of annotations and artefacts along the development process.

The **Chapter 6** proposes a structured model for annotations. The goal of this model is to formalize annotations within the context of the design process of interactive systems while taking into account the specificities of annotations such as the targeting of artefacts, the alternative representation for annotations, and their corresponding lifecycle.

The **Chapter 7** presents an architecture for integrating annotations made on diverse artefacts. This architecture is illustrated with a tool support called ARMADILLO designed for the management of annotations. This tool is compliant with our annotation model described in the Chapter 6. The ARMADILLO tool encompass a project workspace featuring several type of artefacts and their respective editors. This chapter shows how to create and to edit annotations as independent entities, which can ultimately be connected to other design artefacts such as user interface prototypes using ARMADILLO.

CASE STUDIES

The **Chapter 8** presents a case study of annotation of prototypes. It illustrates the user interface of a flight booking system. In this chapter we introduce the PANDA ecosystem, which is a dedicated tool for prototyping user interfaces supporting annotations. PANDA is used here to illustrate the annotation model presented at the Chapter 6. We demonstrated how annotations created directly on PANDA are inserted into the ARMADILLO annotation system. We also demonstrate how ARMADILLO helps, thanks to the annotations, to track the evolution of prototypes build using PANDA.

The **Chapter 9** is aimed at showing how annotations created over diverse artefacts can be handled to provide an integrated view of the project. For that, we introduce a case study around the CIRCUS framework. The case study illustrate an application called “Weather Radar” (WXR) and the corresponding specifications built with models available in the CIRCUS framework for prototyping the user interface (PANDA), the users’ tasks (HAMSTERS), and the behavior of the application (PETSHOP). In this case study, the annotation model is used to structure the design artefacts of a project by binding them together and by featuring annotations on several design artefacts.

The **Chapter 10** demonstrates the transfer of our ideas to the industry. This chapter describes how some of our contributions were applied in a commercial tool called *eazly*TM developed within Softeam Software Business Unit of the Softeam Group which co-funded this PhD thesis under a CIFRE scholarship (ANRT co-funding). This tool is an editor that allows its users to edit e-service applications after their deployments on production servers. The main interest of this case study is to emphasize on the consequences and the tracking of the evolution of an interactive system after its deployment.

Some of the scientific contributions have been subject of publications (2 international journals, 2 full papers in international conferences with peer reviewing, and 1 late-breaking results in an international conference with peer reviewing). These publications are duly cited along the chapter.

Chapter 2. State of the art of development process

Summary

This chapter provides an overview of the development process for interactive systems. It presents a view at glance of the most referenced development process, giving a particular attention to the ISO standard User-Centered Design process. This chapter provides a synthetic analysis of the development process with respect to the identification of the actors of the processes, the involvements of the client and end-users, the iterative processes aspect, and the artefacts produced during the process. This analysis is mainly focused on the macro development processes. The micro process describing how annotations intervene along the development process is given at Chapter 5.

2.1. Introduction to development process for interactive systems

The development of interactive systems requires a plan and a systematic approach. A development process can be explained as a kind of black box where only the input (i.e. the requirements) and the output (i.e. the product delivered) are known, as illustrated by Figure 2-1.a. However, a black box process does not inform the steps that lead from requirements to the final product thus preventing a systematic analysis of problems and reducing the possibilities of a transparent coordinating the activities among members of the development team; as a consequence, it is difficult to monitor the progress of the development and when the product appears at the end of the process it is often too late to care about the quality. A black box process is even worse for dealing with software products because that development process often starts with informal requirements raised by clients who are not able to translate their perception of the business world into precise requirements [3]. In the beginning of the process, software requirements are often informal, incomplete, they can be sometimes contradictory or not reflecting the customers' needs. In a black box process, there is no guidance for revising initial requirements or to introduce new requirements identified along the way.

Conversely, when an explicit process is in place we can see through complexity: activities can be decomposed into smaller and more manageable steps, members of the development team know the steps which help to coordinate their activities, introduction of errors can be prevented by applying solutions to known problems in a systematic way, and quality can be checked at every step of the process (as illustrated by Figure 2-1.b). Moreover, progress can be monitored and communicated to clients who can help to clarify, complete, refine and possibly revise requirements that might change during the process, thus increasing the chances that the final product meets the customers' expectations. Thus, whilst is tempting to deal with complexity in software development by separating the product (what is visible to the client/costumers) from processes (how this quality products can be achieved), we should recognize that product and processes are intermingled because it is by controlling processes that developers can inject the required qualities of products, to reduce time to market and to manage development costs. Therefore, it is clear that to achieve quality and software correctness we have to go inside the black box, describe steps into details and make sure that the process is structured in such a way that makes the development systematic and less suitable to include errors.

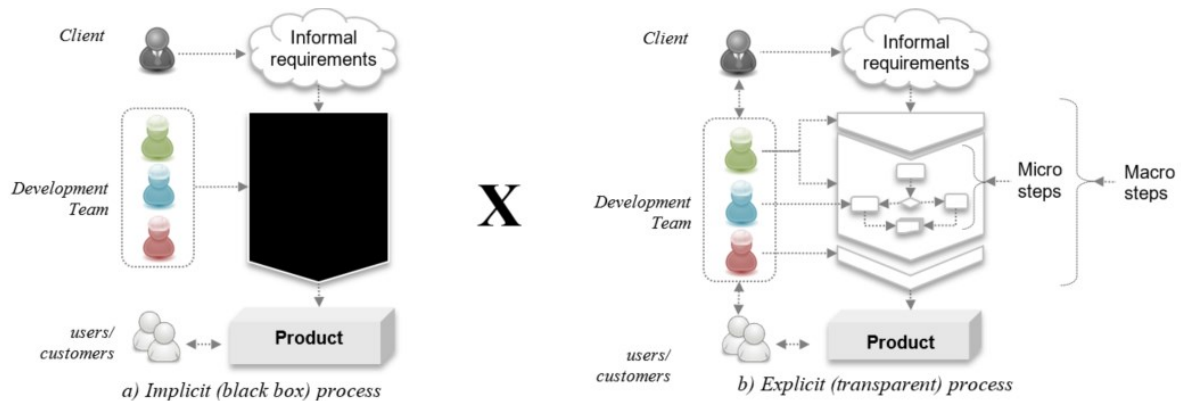


Figure 2-1 Overview of development process as an implicit black box (a) versus an explicit process (b)

Therefore, the main role of development process which aims at guiding the development team throughout their activities. The development process helps to describe what should be done by the different members of the team; how their activities are connected to each other; how they coordinate and communicate their work, and how they manage resources in order to produce the interactive product that meet users' requirements. The development processes can be analyzed in two levels [8]: the macro level that provide a global overview about the process, and the micro level that are aimed at providing details on how to accomplish specific tasks in a particular step.

This chapter presents a list of common macro development processes for building interactive systems and we pay a particular attention and we analyze them according to the following criteria:

- Identification of the actors of the processes; in our case these encompasses any of the possible role involved in the design process such as information architect, ergonomist, designers, etc.
- Involvements of the client (the person who contracts the development of the interactive system) and end-users (the person that actually uses the interactive system);
- Iterative processes that allow revision of artefacts produced along the process. This criterion is considered important to analyze how decisions made at one step may affect the elements produced in previous activities, allowing to revise them to support a consistent specification of the interactive system

These section 2.2 present a summary of design process for the development of interactive systems. Particular attention is given to ISO User-Centered Design (UCD) process because this is the reference design process for developing interactive system. The section 2.3 presents a synthesis of these development process and the section 2.4 the conclusions.

2.2. Overview of the development processes

There is a large literature on development process. This section reports the most frequent ones in the literature of engineering interactive systems. We reuse part of the analysis of development process performed in the PhD thesis of Célia Martinie [9] and we extend it to cope with respects to User-Centered-Design processes.

2.2.1. Waterfall process (Royce, W. W., 1970)

The waterfall process [10] illustrated in the Figure 2-2 is describing a sequence of activities in which each activity is reusing artefacts produced from the previous activity.

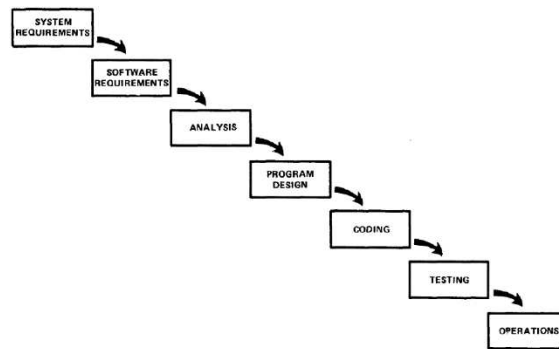


Figure 2-2 Waterfall process (Royce, 1970)

In this design process, the sequence of activities is performed internally by the design team until the delivery of the interactive system or until the final tests prior to the deployment of the application when users can access to the design solution. Thus, usability problems and technical issues are only identified in the final steps of this design process. In this design process, fixing these issues requires to go back to early phases of the process which can represent an expensive cost depending on the steps to redo.

Thus, this process has later been revised to integrate a verification step (McConnell, 1996) which control the design prior to proceed to the next step of the design process and to limit the cost of maintenance of the application.

2.2.2. V-model (McDermid & Ripken, 1983)

The V-model [11] has been designed to ensure that for each step producing an artefact (e.g. needs analysis, specification), a validation step corresponding to this artefact is performed. The steps of this design process can be sorted into three categories (Figure 2-3):

- The descending phases (left side of the V) is dedicated to the refinement of the needs until the implementation of the application.
- The implementation of the application (bottom side of the V) is done after the design phase.
- The ascending phase (right side of the V) is corresponding to the verification of the steps of the descending phase

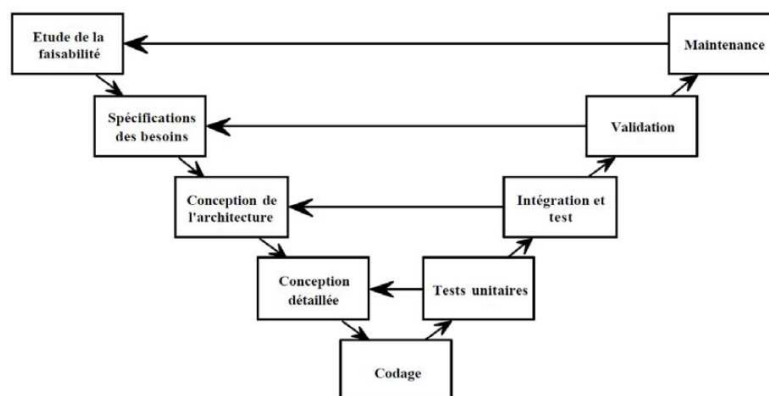


Figure 2-3 V-model development process (McDermid & Ripken, 1983)

In this model, each design step has a matching step for its verification. While some tests can be performed during the design step, most of those verification are done after the implementation. The tests to perform are specified during their matching steps in the descending phase. For instance, the tests dedicated to the verification of the compliance of the application toward the specifications noted

during the design phase will be specified by the same people who wrote the specification. If the implementation needs to be fixed, the design process must go back to the beginning of the cycle to update the artefacts produced in the following steps of the V-model which also include the tests.

While these approaches allow to support the traceability of the requirements of the system along the development process, they are not suited for the development of a usable interactive system since the end-user is not part of the process.

2.2.3. The Nabla model (Kolski, 1998)

The Nabla model [12] is inspired by the V model but make a distinction between the analysis of the human-machine system and the design of the system. The Figure 2-4 below presents the Nabla model. The left-side of the model is dedicated to the tasks concerned by the HCI while the right-side of the model is dedicated to the application module.

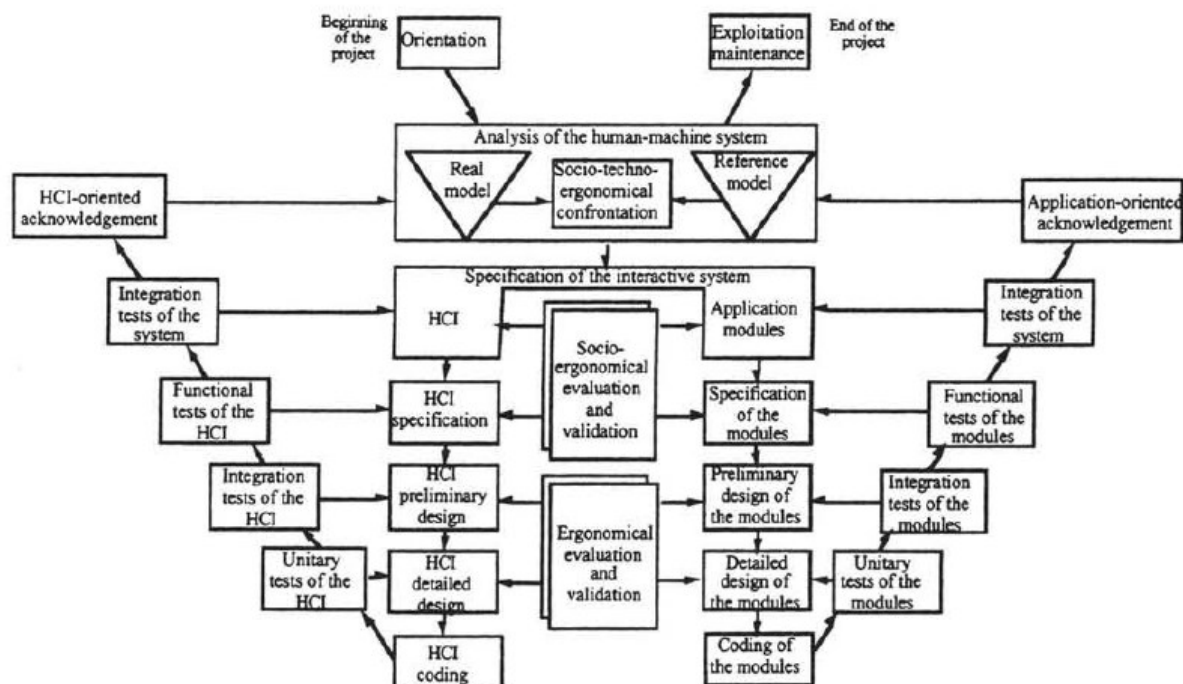


Figure 2-4 The Nabla model [12]

This model starts with the definition of the orientation of the project which include the identification of the objectives, the constraints, and the defining the project organization.

After that, an analysis of the system is performed by distinguishing the “real model” to the “reference model”. The real model is corresponding to existing or virtual human-machine system and defines constraints, its pros and cons. The reference model is the ideal human-machine system which include every point of views and every requirements of the different users. This reference model include a list of criteria to meet.

Then, a comparison of those two models is made for defining compromises to meet as much requirements as possible followed by the specification of the user interface and the specification of application modules. The specifications produced are then evaluated and validated.

After the specifications, a preliminary design and a detailed design are performed which corresponds to integration tests and unitary tests in the V model which is followed by the implementation. At this

stage, Kolski and Loslever [12] emphasize the importance of the evaluation and validation of the design.

The Nabla model ends with the exploitation and the maintenance of the application.

2.2.4. Spiral model (Boehm, 1986)

The spiral model [13] is a model based on the iteration of four steps (Figure 2-5). While these steps will be the same, the content they are producing is evolving through the iterations. Thus, this model is based on the following steps:

- Definition of the goals, possible alternatives and constraints of the project
- Evaluation of the alternatives based on the requirements and the constraints of the project. This step is including a prototyping activity which will feature a prototype more and more operational through the iterations
- Design of the more suitable solution from the previous step. This step is including the design and the verification tasks described in the V-model.
- Planning of the next step by affecting tasks to perform to the design team members

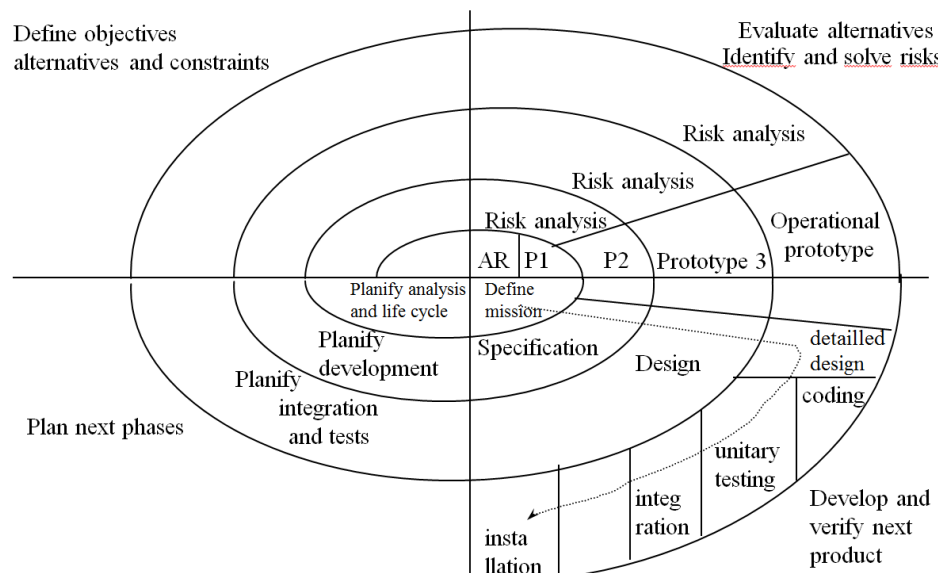


Figure 2-5 Spiral model development process (Boehm, 1986)

This development cycle can be used to explore alternative design. The iterations allow to refine the requirements, the needs and the specification of the application. Once the refinement is sufficiently mature for a decision making, the design team is proceeding to a development cycle such as the waterfall model or the V-model. While this development cycle is expensive and lengthy, the advantage of this development cycle is that it takes into account the usability of the final application.

2.2.5. Rational Unified Process (Kruchten, 2004)

The Rational Unified Process (RUP) [14] is a development process aimed at the creation of a product matching the users' needs. This design process is:

- Iterative and incremental: the project is divided into short-term iterations which produce an executable version of the application. This version is incremented during each iteration
- Model based: the architecture of the application must be graphically modeled. This process is based on the UML (Unified Modeling Language) notation [15].

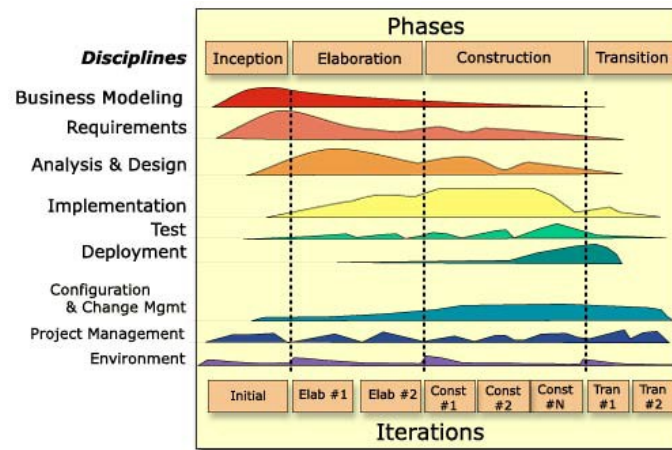


Figure 2-6 RUP development process

The Figure 2-6 is representing an overview of the process which is composed of two dimensions:

- The horizontal axis is corresponding to the time and the deployment lifecycle which include the inception, the elaboration, the construction and the transition
- The vertical axis is representing the discipline, the people and artefact involved in the design process. This include the business modeling, the requirements, the analysis and the design, the implementation, the tests, the deployment, the change management, the project management, and the environment

RUP is a turnkey approach proposed by IBM who provide the process and the tools. The tools offer a canvas of projects, a sharing of documents within the design team and promote the reuse of models. However, this approach is linked to Rational tools and while this approach is compatible with the production of usable systems, the required tools are expensive.

2.2.6. AGILES methods (Cockburn, 2002)

The AGILE methods (officialized in 2001 by the Agile manifesto [16]) are aimed toward the satisfaction of the client by integrating him within the design process. These methods are based on four fundamental values:

- The team ("Individuals and interactions over processes and tools")
- The application ("Working software over comprehensive documentation")
- The collaboration ("Customer collaboration over contract negotiation")
- The acceptance of changes ("Responding to change over following a plan")

The SCRUM approaches (for the management of the development) and eXtreme Programming (for the software development) are following these precepts.

- **SCRUM (Shwaber, 2002)**

SCRUM [17] is an approach for the management and the tracking of the software development. It is an iterative and incremental approach. A customer representative is called "product owner" is responsible to transmit the orientation of the project to the design team, to define the features to develop and suggest an order of implementation to provide a tool matching his needs. The product owner is entering this information in the product backlog which list the needs and the features desired by the customer. This list is ordered by the value granted by the product owner which is defined by the following criteria: the return on investment, the criticality of the feature for the system or the users, the development cost, etc. The product backlog is visible for the whole team and is used for a better

communication on the goals of the project since each team member can see the features asked by the product owner.

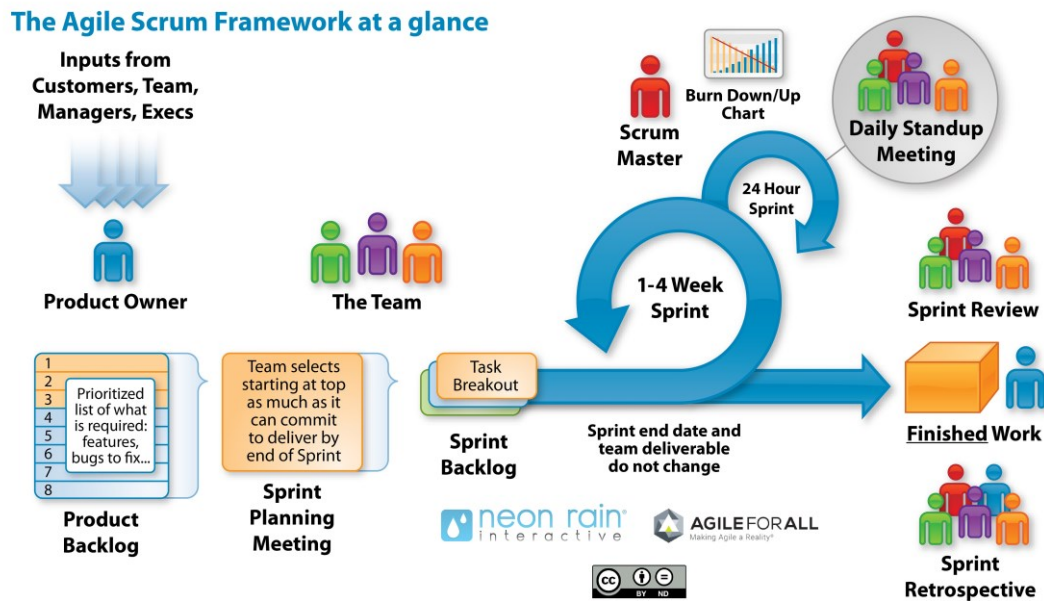


Figure 2-7 The steps of the software development with SCRUM retrieved from <https://www.neonrain.com/agile-scrum-web-development/>

The development is decomposed into iterations called “sprints” which last from three to four weeks. This short duration allows to quickly deliver releases of the software to the product owner which includes the features listed in the product backlog. Another advantage of this short duration of iterations is that it allows the product owner to adjust the features he ask as the project is progressing.

This approach is strongly focused on the customer’s needs but in the case where the customer is not the final user. The usability of the interactive system can be taken into account through the use of user stories. These user stories can be written to accurately describe the use of the system by an end-user and can be written by anyone, including an end-user.

- **eXtreme Programming (XP, Beck, 1999)**

The eXtreme Programming [18] (Figure 2-8) is defining a set of methods for guiding the production of a software. In XP, User stories are similar to use cases defined in UML and replace the specifications given by the customer. These user stories are written in natural language by stakeholders and are used by the designers for evaluating the duration of the iteration and for testing the design.

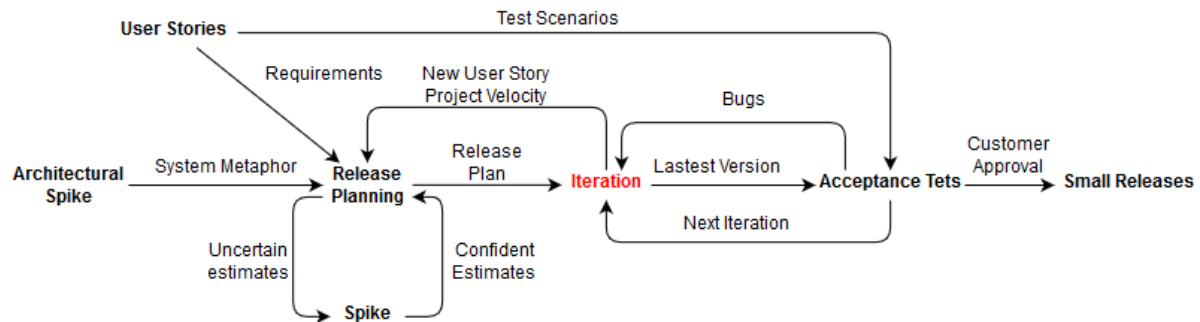


Figure 2-8 eXtreme Programming development process

An iteration starts with the release planning of the iteration, the scenarios and the tests to implement. The planning is done by the designers and the customers. The customers choose the scenarios to implement in the next iteration while the designers evaluate the duration of the iteration. The spike solution is done by exploring several solutions and are planned only for difficult problems to solve. A planning is imposed for each scenario: a scenario must be implemented within one to three weeks, otherwise, the scenario must be decomposed. A test is representing a specific behavior expected by the system. A user story is validated once each unit test associated to this scenario is valid. When tests are successfully performed, the current version of the application is updated with the last features (small releases).

XP also defines rules and methods for the process (planning, coding, design and test). The advantage of this method is that it is easy to apply, it is cost-effective and features numerous iterations and releases of versions. The time is essentially spent on the technical aspects: development and tests. However, this method does not cover the steps before and after the development such as the analysis of the users' needs and the maintenance of the application.

Overall, AGILE methods are interesting for ensuring the customer satisfaction by controlling the costs and the time dedicated to the development thanks to the iterative and incremental approach. However, these methods do not guarantee the reliability and the usability of the released product.

2.2.7. Star model (Hartson, 1989)

The star model [19] in the Figure 2-9 is featuring six different steps of the development process. The main principle of this model is that any step of this process must be evaluated before proceeding to the other steps. Thus, this process can start at any of the five steps located at the end of each branch of the star but the evaluation is always done after each steps of the process. This model is flexible and does not impose any path for the process.

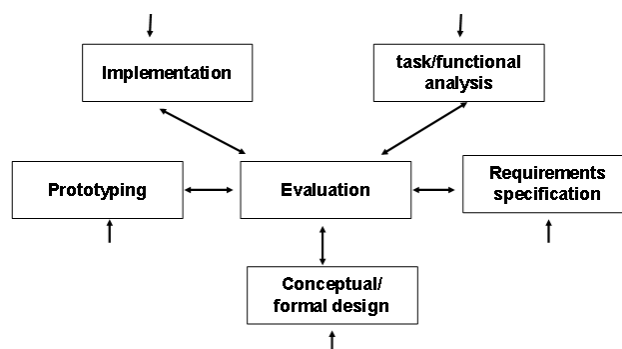


Figure 2-9 The Star model

2.2.8. The layered development process model (Curtis, 1994)

The layered process model (Figure 2-10) has been proposed by Curtis and Hefley in [20]. The main principle of this model is to separate the development into two steps: the “Software Engineering” corresponding to the standard software development and the “User Interface Engineering” focusing on the Human-computer interaction.

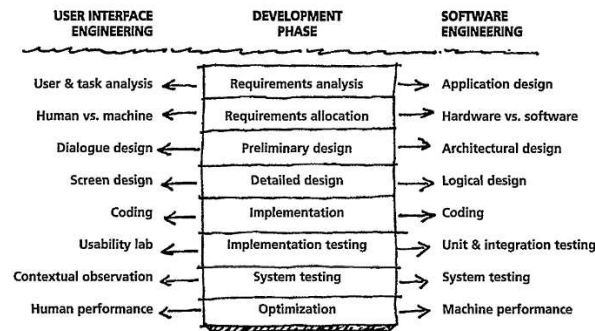


Figure 2-10 The layered process model [20]

This process is establishing a relationship for each steps of the development between the software engineering (e.g. system testing) and the user interface engineering (e.g. contextual observation). This process is defining a clear separation between these two parts of the interactive system.

2.2.9. The Object-Oriented User Interface (OOUI) design process (Collins, 1995)

The OOUI design process [21] represented in the Figure 2-11 is an iterative process. It is focused on the task analysis by considering it as a key step of the UCD process. Thus, human factors and software engineering factors are integrated within this process. This design process is more focused on describing the relationship between these factors rather than providing a method for developing an interactive system.

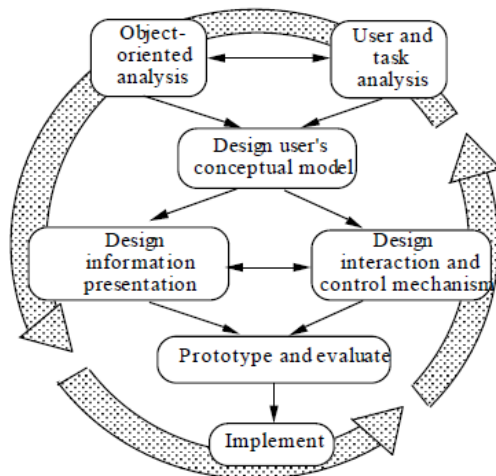


Figure 2-11 OOUI design process

2.2.10. The Iterative-cyclic process (Rauterberg, 1992)

The iterative-cyclic process [22] is a user-centered participative process and is composed of four quadrants (Figure 2-12):

- Upper-left quadrant: Analysis. The users' task and requirements are collected

- Bottom-left quadrant: Specification. Creation of the user-interface as well as its conceptual and organizational description
- Bottom-right quadrant: Implementation. The application is created
- Upper-right quadrant: Trial and Assessment. The application is tested and validated to check if it is compliant with the users' requirements.

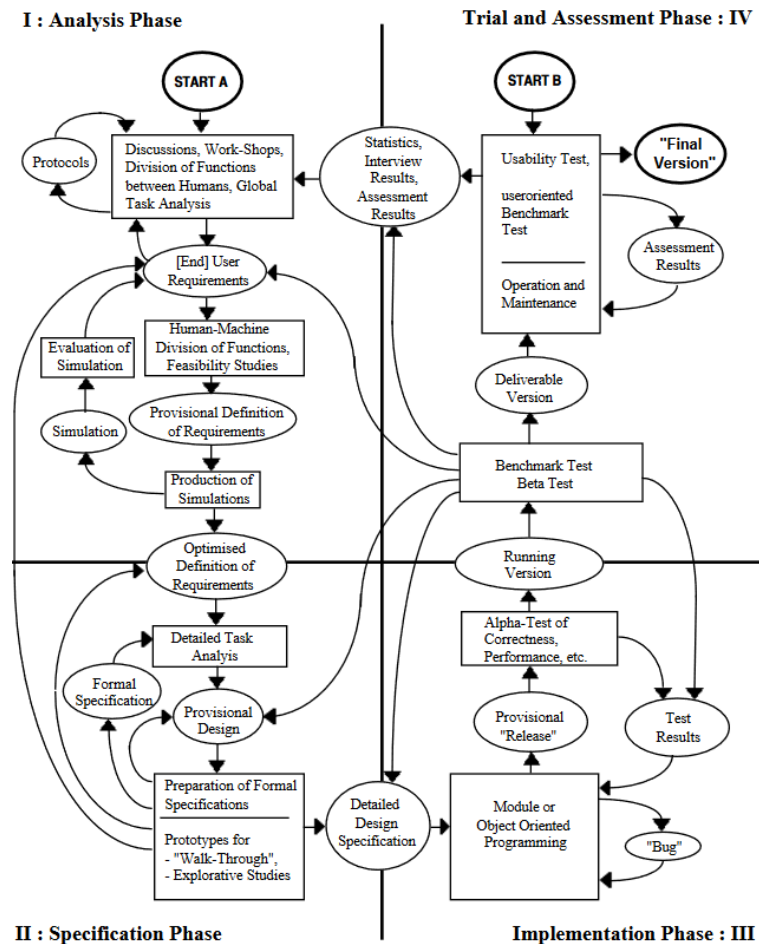


Figure 2-12 The iterative-cyclic process

The iterative-cyclic process is completing the standard design process with the following concepts:

- Final users' requirements during the needs analysis step
- The use of prototypes during the specification step
- A formal specification step during the specification
- Usability tests during the validation

This design process allows the development of usable interactive system.

2.2.11. User-centered System Design (Gulliksen, 2003)

The User-centered System Design (USCD) [23] is a process dedicated to support the usability of the system during the design process as well as during the lifecycle of the system.

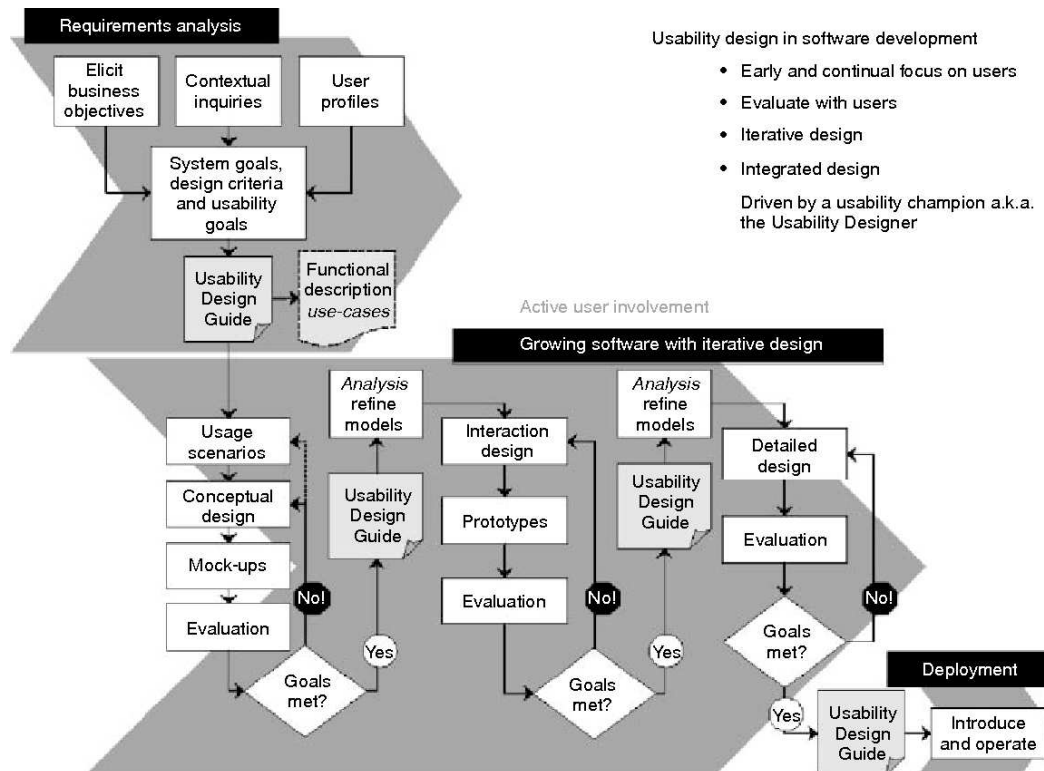


Figure 2-13 Usability Design Process

This process emphasizes the following concepts:

- The user-oriented aspect
- The active engagement of the user from the beginning of the process
- The iterative and incremental development
- The simple representation of the development understandable by users and other stakeholders
- Usage of prototypes
- Usability goals for leading the development
- Explicit and clear development activity
- Pluridisciplinary team including usability expert during the process
- A holistic development by developing in parallel the aspects that will influence the future usages
- The customization of the process depending on the organization

The goal of this process is to balance the interaction design, the analysis and the evaluation. Thus, this process can be divided into three steps (Figure 2-13): Requirements analysis, growing software with iterative design and deployment. This process is imposing the release of a Usability Design Guide which must feature information from the requirements analysis as well as the different design choices made during the process.

During the requirements analysis, the design team is focused on the understanding of the goals of the company, the final users' tasks and needs and on establishing usability and development goals. This part of the design process is continually evolving during the development process while the required information can be added.

The second step contains three main loops: the conceptual design, the interaction design and the detailed design. These iterative loops include the design and the evaluation.

The deployment step contains the different guides to help using the system such as user guide, online help and training. This step is not necessarily final and can be the end of an iteration cycle. The deployment can be used to present the different subparts of the system and allow to adjust either the system or the process.

While this design process is oriented toward the usability of the system, it does not feature all of the design steps. Thus, it must be integrated within a more complete process.

2.2.12. The Usage-Centered Design (Constantine & Lockwood, 2002)

The Usage-Centered Design process [24] is a design process focusing on the tasks to support as opposed to other user-centered design processes which focuses on the user experience and his satisfaction.

The Figure 2-14 below extracted from [24] is comparing the User-centered and the usage-centered design processes.

User-centered and usage-centered design: A comparison

User-centered design	Usage-centered design
Focus is on users: User experience and user satisfaction	Focus is on usage: Improved tools supporting task accomplishment
Driven by user input	Driven by models and modeling
Substantial user involvement	Selective user involvement
• User studies	• Exploratory modeling
• Participatory design	• Model validation
• User feedback	• Usability inspections
• User testing	
Design by iterative prototyping	Design by modeling
Highly varied, informal, or unspecified processes	Systematic, fully specified process
Design by trial and error, evolution	Design by engineering

Figure 2-14 Comparison table between the User-centered design and the Usage-centered design (retrieved from [24])

One of the key aspect of the usage-centered design process is that it is based on models:

- The “Role model” captures the characteristics of roles that users plays
- The “Task model” defines the structure of the work the user needs to accomplish
- The “content model” or “abstract prototype” describes the content of the UI and its organization

This design process starts with two preliminary steps:

- The “Essential purpose and preconception” which clarify business and define user purposes
- The “Exploratory modeling” in which the design team identify questions and uncertainty in user requirements by sketching draft of models

After these preliminary steps, the process is followed by iterations of the design of the different models which are roughly designed at first. The iterations include the design of the overall architecture of the UI (organization, navigation, look and feel, tasks to be covered), the modelling of the user role through card based modelling and the inspection of the models produced.

2.2.13. The ISO User-Centered Design (UCD) process

According to the ISO standard 9241-210 revised in 2018 by [25], the User-Centered Design (UCD) is an iterative design process in which the end-users and their needs are the main focus. This design process, as illustrated below in the Figure 2-15, is based on the study of the context of use of a software, the specification of the application, the production of a design solution and the evaluation of this solution until the solution designed is meeting users' goals and needs.

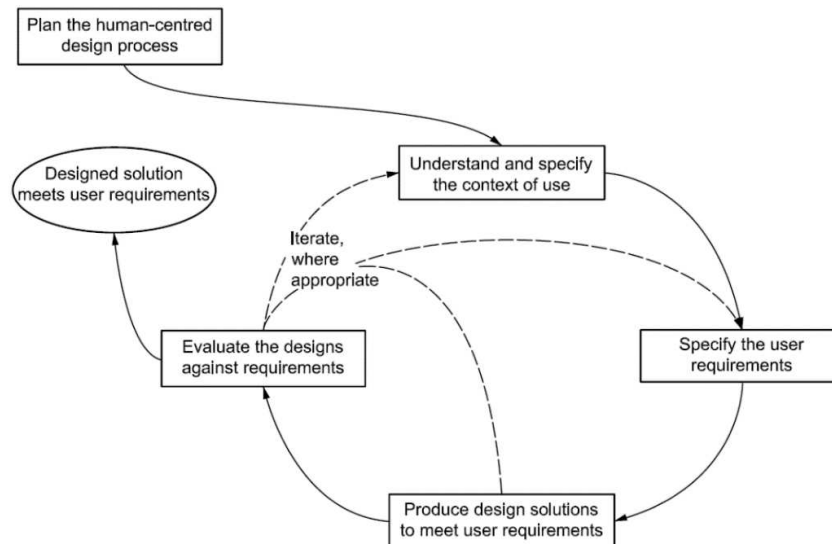


Figure 2-15 ISO 9241-210 standard for human-centred design processes for interactive systems

In the early iterations of the design process, the perimeter of the design solution is blurry and the specification has to be refined through the study of the context, the collection of data, the study of results from previous iterations, multiple tries on the design solution and validation with end-users.

The production of a design solution can either be a step in which the design team explores a new alternative of design or refine a solution and specify the final design adopted. On top of that, new needs can appear during the iterations. Those conditions imply that the first iterations and thus each step should be limited in terms of investment to get early feedback on what is wrong on the design choices or the understanding of the users' need. This approach will minimize the cost of maintenance and the development cost of unwanted features and ill-suited user interfaces and can be supported by using sketches and paper prototypes.

Indeed, the prototyping activity allows the stakeholders of the design process to collaborate to produce a partial representation of the interactive system. There are several advantages to use prototypes: it is a solution that is inexpensive to produce quickly and it allows to communicate over ideas by featuring a concrete representation of a solution.

In addition to prototypes as a communication medium, stakeholders of the design process also use annotations during the prototyping activity for various purposes such as adding complementary information on prototypes, listing concepts and ideas, noting relevant information, or reminders for future activities.

Later on, the iterations of the design process might require more accurate prototypes in terms of interaction or graphical representation which implies a change in the tools used for the creation of prototypes.

Thus, the needs of the design team regarding the creation of prototypes evolves during the design process and are depending on the goal of the prototype set by the design team.

2.2.13.1 Presentation of the tasks of the UCD

For the sake of simplicity in the presentation of the UCD, the activities “understanding of the context” and “specification of the users’ requirements” will be studied together as one task.

Thus, the UCD process can be broken down into the iterations of 3 main tasks: collecting and refining information from users, the production of a solution from the information collected that might meet user requirements and needs, and the evaluation of the solution by presenting it to users. However, the UCD does not show how to perform the tasks.

- **Collecting and refining information**

This task consists in collecting the information relevant for the interactive system being designed and refining the information in order to make it usable for the following tasks of the design process. This can be done through various activities (e.g. identification of users, analysis of the context, interviews with end-users, creation of personas, modeling of the users’ activities).

The goal of the integration of those activities in the studies is to have a better understanding of the context in which artefacts are produced and used during the design process as well as to identify a potential track for building a knowledge base of the design process using those artefacts. This knowledge base will then be used for the traceability of the decision made throughout the design process.

On top of the study of those activities, the evolutions of artefacts and resources used and produced during the UCD process will be examined. In each of the tasks of the design process, a set of artefacts are produced or used as a reference throughout the different iterations. Those artefacts can be of various nature like guidelines provided by the customer, the transcription of user interview, user test results, or prototypes of the solution. They contain valuable information about the context of the project but also about the progress of the design process.

- **Production of a solution**

This task consists in modeling the different aspect of the interactive system by following the information specified during the previous task. This modeling can be made by creating prototypes that will specify and respect the specification of the design of the interactive system. The creation of prototypes can be divided into several tasks to cover the different aspects of the interactive design: designing the UI, the behavior of the prototype, the data manipulated by the interactive system, the services communicating with the interactive system and so on.

The goal of the integration of this activity is to have the mean to study and identify the different goals of a prototype, what are the key elements of those prototype depending on the goal, evaluate the suitability of prototyping tools regarding those key elements, study the process of creation of a prototype and of the resources used for this creation as well as study the usage and the evolution of the prototypes produced.

Another aspect to examine is the exploratory approach enabled by the usage of prototypes while keeping in mind the traceability of design choices and evolutions through the design process. The design process can lead to different prototypes that can be developed in parallel given a limited set of resources. In the prototyping context, each potential solution aims to solve the same problem and they can be tested. Indeed, prototypes can be inexpensive to produce thus allowing a comparative study

between concurrent prototypes. Furthermore, thanks to user tests and their results, it is possible to provide concrete arguments to which ideas from prototypes the design team should promote during the decision process.

- **Evaluation of the solution**

This task consists in the evaluation with end-users of the solution produced during the previous step of the design process. The results of the evaluation will allow the design team to gather information on the adequacy of the design solution with the users' needs and requirements as well as to refine the existing needs and to inquire about eventual new needs and requirements.

The goal of integrating this activity in the study is to pursue investigations on the integration of the feedbacks within the design process and its impact on prototypes, the processing and evaluation of those feedback for the traceability of the design and for the management of the information by the design team.

2.3. Synthetic analysis of the development processes

The Table 2-1 below present a synthetic analysis of the different development process listed in the previous section. In that table, development processes are classified according to actors involved in the process, the involvement of the client with the project type of process (iterative or sequential) and the list of artifacts produced. This classification is based to the original description in the literature. However, we should assume that in real life projects these macro projects are not always instantiated as such and might be adapted to cope with the idiosyncrasies of specific projects.

	Actors of the process	Active involvement of the client	Active involvement of the end-users	Type of process	Artefacts produced
Waterfall process	Development team	No	No	Sequential	Requirements, Specifications, Code, Tests, Executable
V-Model	Development team	No	No	Sequential	Specification, Architecture design, Code, Tests, Executable
Nabla model	Development team, end-users	No	Yes (for the evaluation)	Sequential	Real model, Reference model, Evaluation results, Validation results, Code, Tests
Spiral model	Development team, Customer, Stakeholders, users, maintenance organization	Stakeholders for reviewing purpose	Optional for reviewing purpose	Iterative	Requirements, Specification, Alternative designs, Constraints, Risk analysis, Prototypes, Executable
Rational Unified Process	Development team, Tester team, Managers, Additional Workers, Customer	Yes for agreeing the description of the requirements	At the end of the process ("Product Release Milestone") during the "transition phase"	Iterative	Business case, Use Case model, Project plan, Risk assessment, Project description, Prototypes, Requirements, Specifications, Architecture description, Development plan, code, tests, Executable
Scrum	Scrum master, Development team, Product Owner	Yes as a Product Owner for prioritize requirements, tasks, and features through the Product Backlog	Only when the client is an end-user or when end-users participates to the creation of User Stories	Iterative	User stories, Product Backlog, Sprint Backlog, Increments, Releases
eXtreme Programming	Development team, Customer	Yes for planning and reviewing	Only when the client is an end-user or when end-users participates to the creation of User Stories	Iterative	Plans, User Stories, Releases, Tests, Code
Star model	Development team, End-Users	No	Yes for the evaluation	Iterative	Requirements, Specifications, Human-Computer Interface models, Prototypes, Implementation

Layered development process model	Development team, End-users	No	Yes	Iterative	Requirements, Users' task and analysis, Design, Implementation, Tests
The Object-Oriented User Interface	Development team, End-users	No	Yes for evaluating the prototypes	Iterative	Requirements, Users' task and analysis, Design, User's conceptual model, Prototypes, Implementation
Iterative cyclic process	Development team, End-users	No	Yes	Iterative	End-user requirements, Users' tasks analysis, Formal specifications, Design specifications Prototypes, Releases, Statistic, Interview, Assessment results, Tests
User-Centered System Design	Development team, End-users	No	Yes from the beginning to the end	Iterative	Usability Design Guide, Models, Scenarios, Conceptual design, prototypes, Evaluation results
Usage-Centered design	Manager, end-users, application domain experts, developers, and other stakeholders	Not mandatory	Yes	Iterative	Role models, Task models, Task cases, Content models, Visual and interaction design
ISO User-Centered Design	Development team, End-users	No	Yes from the beginning to the end	Iterative	End-Users' Requirements, Specification, Prototypes, Implementation, Evaluation results

Table 2-1 Synthetic analysis of the development process

As we shall see, all the development processes analyzed in this chapter acknowledge that many actors participate in the development process. Even when specific roles are not properly labelled, actors are considered part of a development team. This aspect justifies that communication between actors must occur somehow. In our classification, we consider the involvement of clients (people who order the development of the project and/or have knowledge about the inner business model but do not necessarily will use the software produced) and end-users (people who will interact with the resulting software). Whilst both client and end-users are sources of requirements for applications, many development processes such as the Waterfall process and V-Model do not mention the presence of these actors.

Most of development processes propose some level of iteration among the steps. Indeed, the various actors needs to collaborate toward a common goal through the different iterations. This specificity of the design process imply that the design must be understandable by each stakeholders of the design process and that the results of the evaluation done at the end of each iteration must be followed up by tasks compliant with the conclusions of these results.

This review of design process showed that various artefacts are produced and reused at different phases of the process by many actors of the design process for various purposes (e.g. planning an iteration, reviewing the design, establish a common understanding of the project). These artefacts support the activities of the design process and evolve depending on the results of evaluations and validations. Nonetheless, there is not a standard set of artifacts to be produced

2.4. Conclusion

In this chapter, we have presented a review of macro development processes for interactive systems which has been previously published in the PhD thesis of Célia Martinie [9]. In this review, we briefly presented the sequence of activities of the different design process and given an analysis of these design process. This analysis lists various properties of the design processes and especially the different actors involved and some of the artefacts produced during the activities of the process.

In [26], I. Khaddam, H. Barakat and J. Vanderdonckt explain through the method engineering that there is a dependency to the tools used to perform the different tasks of a design process. However, as explained in the Chapter 4, tools support of annotation is not suited for its use in the context of the design process. Indeed, annotation support is not homogenous since each tools propose their own implementation of annotations, thus making it difficult to process annotations on a global level.

Whilst annotations are often used to support the activities related to the evolution of artefacts, we did not have found in the literature any design process that make explicit reference to annotations. The practice shows that annotations are widely adopted as a tool for the design of artefacts, but their integration within the design process is still overlooked. This absence of articulation between annotations and macro development processes for interactive system motivates the development of micro processes, described in the Chapter 5, for handling annotations.

Chapter 3. Analysis of annotations

Summary

This chapter provides a presentation of annotations. It starts with annotations in physical text documents but the final target is their integrations on digital documents. This chapter gives a brief presentation of the structure of annotations and reviews the different classes and dimensions of annotations defined in the literature. After that, a presentation will be given on the different usages of the annotations on documents found in the literature. Finally, a presentation of a standard for web annotations (i.e. the Web Annotation Data Model [33]) is given.

3.1. Introduction

Annotations have been used for a while in human history as illustrated in the Chapter 1. The first studies about annotations started with the identification of common practices by university students on their paper textbooks [27; 28]. Many of the elements of paper-based annotations were then transposed to electronic documents. In the context of the design of a product, the annotations are answering several needs that show that design artefacts are not self-sufficient for their use by people involved in the design process.

The section 3.2 is giving a presentation of the structure of annotations. The section 3.3 introduce the different dimensions that characterize annotations as identified by C. Marshall in [28]. The section 3.4 gives an extensive presentation of uses of annotations on documents found in the literature followed by an analysis of the different approaches given by the literature for these uses. The section 3.4.4 identify other uses of annotations for digital artefacts. The section 3.6 is dedicated to the presentation an existing model, the Web Annotation Data Model [33] that define a structure for providing a standard of annotations over web resources.

3.2. Nature of annotations

A common definition of annotation such that provided by Bringay et al. [29] below, refers to documents:

*“An annotation is a particular **note** linked to a **target** by an **anchor**. The target can be a collection of documents, a document, a segment of a document (paragraph, group of words, image, part of image, etc.), and another annotation. Each annotation has content, materialized by an inscription. It is the trace of the mental representation elaborated by the annotator about the target. The content of the annotation can be interpreted by another reader. The **anchor** links the annotation to the target (a line, a surrounded sentence, etc.)”.*

Looking forward to more robust definition of annotations for digital documents, we have found the definition of Kahan and Koivunen [31] who define annotations as “user made statements”, consisting in a **body** (i.e. text note or graphical content), a link (the so-called **anchor**) to the **target** which include a location within the document as well as other metadata. In the Open Annotation Data model and the Web Annotation Data model proposed by Sanderson et al. [32; 33], annotations are considered as a set of resources in which the body is related to one or several targets (ex. document annotated) as illustrated by the Figure 3-1.

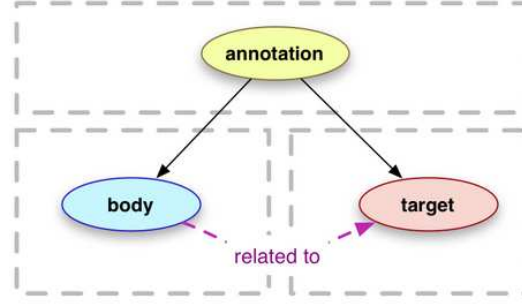


Figure 3-1 Simplified model of annotations defined by the Web Annotation Data Model [32]

A formal definition of these elements is provided by Agosti M. and Ferro N. [49] and illustrated in the Figure 3-2:

Definition 9.1. An annotation $a \in A(k)$ is a tuple:

$$a = (h_a \in H(k), au_a \in USR(k-1), G_a \in 2^{GR(k-1)} \times P, sp_a \in SP, \mathcal{A}_a \subseteq SN(k) \times LT \times ST(k) \times SM(k-1) \times H(k-1)),$$

where:

- h_a is the unique handle of the annotation a , i.e. $h(h_a) = a$.
- au_a is the author of the annotation a : i.e. $h_a \in au(au_a)$.
- G_a are the groups of users with their respective access permissions for the annotation a , specified by the pairs (G, p) with $G \in G_a$ and $p \in P$.
- sp_a is the scope of the annotation a .
- Each n-ple of the \mathcal{A}_a relation means that the annotation a by means of a sign in $SN(k)$ and a link type in LT is annotating or relating to a segment in $ST(k)$ of a stream in $SM(k-1)$ of a digital object identified by its handle in $H(k-1)$.
Note that since $\forall sm \in SM(k-1) \mid \exists \alpha \in \mathcal{A}_a, \alpha = (sn, t, st_{sm}, sm, h)$ must be $sm \in hsm(h)$; in other words, the stream sm must be contained in the digital object identified by the handle h .

Figure 3-2 Formal definition of annotations according to Agosti M. and Ferro N. in [49]

All those definitions acknowledge the distinction between two elements being in relation among them: the annotation body and the annotated document, which are linked somehow by an **anchor**. Agosti and N. Ferro [49] stress the importance of the linking mechanism to formally describe annotations. This linking can be made in several ways, depending on the support of each part of the annotation. For example, when annotating books, notes can be written in the margins or in post-its. Marshall [27] identified 3 different mechanisms of association on textbooks including: arrows to connect the document to its annotation, marks such as bracket and brace, and the proximity of the annotation and the target by writing in the margin or the interline. However, as for digital annotations, the artefact annotated and the annotation can be located in the same file or in two different files. We add to this list of linking mechanisms the usage of reference, which allows a nonintrusive way to annotate a document. The linking between a **body** and a **target** created by an **anchor** might convey particular meaning for the annotations, as suggested by S. Bringay et al. [29]:

- The placement of an annotation can be explicit (when the target is clearly visible in the document), or tacit anchor (when placed in the document but not connected to a particular element in the document).
- One-target versus multi-target anchor, w.r.t. the many possible targets for an annotation;
- Conventional and nonconventional anchor, w.r.t. the existence (or not) of an agreement for interpreting annotations (ex. red marks mean might refer to a convention for important topics).

As we shall see, these three elements (**body**, **target** and **anchor**) are core concepts not only for paper-based or electronic documents but they are essential to understand how annotations applies to the many artefacts used to build interactive system as well.

3.3. Annotations classes and dimensions

While the previous section described the structure of an annotation, this section is dedicated to the presentation of dimensions identified by C. Marshall in [28]. These dimensions allow to distinguish annotations by evaluating them through different criteria and thus, to organize them into different classes of annotations. Those dimensions are not discrete but continuous.

- Formal/informal annotations

Formal annotations are annotations that follow a defined structure (e.g. using standards, conventional naming). The main advantage of formal annotations is that they are more adapted for interoperability. Informal annotations on the contrary are not restricted to their structure and what they should contain. Metadata are qualified as formal while marginalia are informal.

- Explicit/tacit annotations

Explicit annotations refer to annotations that can be easily understood and without misinterpretation while tacit annotations refer to annotations that are incomplete, unexplained and need to be interpreted in order to acknowledge the intentions of the author when he annotated the document. C. Marshall notes that even authors of tacit annotations could meet some difficulties while reading them after a long time. Example taken by C. Marshall are a note saying “No!” an unexplained link or a bookmark for tacit annotations. She also points out that many personal annotations are tacit while public annotations tend to be more explicit.

This dimension is quite similar to the “abstraction level” attribute identified by M. Zacklad in [51] for annotation used as criticism which consists in the level of details of the argument presented in the annotation.

- Annotation as a part of a writing activity/annotation as a part of a reading activity

This dimension is used to qualify the degree in which the annotation is a writing. Indeed, annotations can be made in a reading activity by extracting information from the target: the annotator “collect, organize, interpret” [28]. An example would be an attentional annotation as defined in [52] by Zacklad where annotation are used to draw the attention. Annotations can also be made in a writing activity in which the annotator can contribute to the document by adding more information, discussing about some part of the document or analyzing it.

- Hyperextensive/Extensive/Intensive annotations

C. Marshall has observed that the annotation activity can be similar to the reading activity regarding the focus given to one or several documents based on Levy’s work [53]. Hyperextensive annotations would link related information like a hypertext link and based on short fragment of information. Extensive annotations associate several documents in order to make a comparison or an analysis afterwards for instance. Intensive annotations are annotations that limit themselves to only one document.

- Permanent/Transient annotations

This dimension refers to the usefulness and the relevance of the annotation over time. An annotation could be relevant only for a short time like an annotation emphasizing a misspelled word or a question

asking a clarification for instance while another annotation can be relevant for a longer time like the explanation given since it could be useful for future reader. Though even when an annotation is no longer relevant, it can be archived or stored with its context.

- Published/Private annotations

This dimension is used to define the visibility of the annotation regardless its initial consignee. Indeed, even if an annotation is written in a tacit way only understandable for its author, it can always be shared to other people. Bringay in [29] add the notion of “sphere of the group” by referring to the term used by Zacklad et al. in [51] (i.e. the “private sphere” and the “public sphere”) in order to qualify annotations that are only accessible to a defined group of people.

- Global/Institutional/WorkGroup/Personal annotations

This dimension define the scope of the annotation. Depending on the targeted audience, the annotation might differ by its vocabulary and its purpose. For instance, a discussion about technical issues would be more relevant in a workgroup, a tacit memo could be used for a personal use and an explicit note could be used to vulgarize a concept.

- Autonomy of annotations

On top of the dimensions identified by C. Marshall, it is possible to note an additional dimension to qualify an annotation: the autonomy. Indeed, S. Bringay et al. [29] distinguish autonomous annotations from the others: an annotation that contains sufficient context element can be considered as autonomous. An autonomous annotation can be understandable even without its context.

3.4. Studies of annotations in text documents

In this part, we will consider the annotations in a collaborative and iterative context, which implies that several actors are involved and each actor can assume a different role. The collaborative and iterative context implies that the artefacts evolves over time, annotations can be read by anyone having access to the document and to the annotation (thus, annotations are not only written to oneself), and discussions can occur between the stakeholders of the design process that can impact the artefact and the annotation.

3.4.1. Presentation of the roles involved in the design process

In [30], Winckler et al. presented an analysis of the role of the different stakeholders involved during the design of web applications. In this analysis, the authors identified two sub-groups: the development team and the external members. The development team includes the role involved with the development of the web application while the external members gather stakeholders who interact with the development team for providing information and constraints on the project.

Within the development team, they also identified 9 roles as presented in the Table 3-1 extracted from [30]. The first column list the name of the role. The second column summarize the responsibilities of the role. The third column lists the main tasks performed by the actor. The last column reference the steps in which the role is active in the development life cycle they defined which consists in the following:

- | | |
|-----------------------------|------------------------------|
| 1. Requirements engineering | 4. Development |
| 2. Specification | 5. Site usage and Evaluation |
| 3. Design | 6. Maintenance |

Role Name	Responsibilities	Main Tasks	Phase
Manager	Oversee and manage the project	- Set project priorities, scope and budget; - Identify tasks and resources required; - Monitor schedule; - Coordinate individuals;	1 – 6
Web Analyst	Do requirements engineering	- Gathering information; - Interview with clients and end-users; - Identify goals and requirements;	1, 6
Web Designer	Specify and design structure, layout and navigation of website	- Specify overall and sub-sites architecture; - Specify content sources and update frequency; - Define level of end-user interaction; - Design navigation;	2, 3
Graphic Designer	Specify look and feel	- Design the appearance including page style, layout, graphic elements, colors ...	3
HTML Programmer	Convert design elements into pages	- Code designed elements in HTML; - Implement the web pages; - Integrate HTML and scripts;	4
Content Editor	Authoring web site content	- Identify content sources; - Collect content (images, sound and text files, etc.); - Revise and write content ;	1, 4, 6
Advanced programmer	Develop specialized code for the web site	- Implement scripts, applets, etc for the website; - Integrate HTML and scripts	4
Web Administrator (webmaster)	Oversees the entire website environment, ensuring that is fully operational and functions properly	- Install and configure the web server; - Manage the files within the site; Make backups; - Ensure privacy and security issues; - Keep web server working; - Provide audit traces;	4 – 6
Human factor expert	Provide support for ergonomics issues	- Perform and organize usability evaluations; - Help designers with ergonomic concerns; - Develop usability guidelines for the project;	2 – 5

Table 3-1 Table of development team roles extracted from [27]

As for the external members, Winckler et al. identified the following roles: client (website's owner), end-users, networking administrator, software engineering team.

As noted by the authors, these lists of roles is not exhaustive. Indeed, depending on specific project's need, other roles can be added.

Regarding the use of annotations, it is possible to identify two main roles: the writer of the annotation and the readers. These two roles are relative to one annotation and can be assumed by any actors involved in the design process. Thus, a member of the development team or of the external members can be writer or reader of annotations.

3.4.2. Uses of annotations

Based on the review of the literature, we summarize hereafter three main functions played by annotations: to enrich a document, to support communication and to support an intention/activity carried out by the author of the annotation. Whilst most of the literature in the matter refers to text documents, we suggest that the following classification is relevant for the development of interactive systems for two main reasons. First, the development of interactive systems is often based on text documents (e.g. specifications) which can contain annotations as described previously. Secondly, the elaboration activity and the uses of design artefacts such as prototypes can also feature a use of annotations similar to any other type of documents (e.g. text documents).

Thus, this classification can help to add semantics to the widget used for creating annotations over design artefacts.

- **A Mean to Enrich Documents**

When adding an annotation to a document, the author of the annotation is extending this document somehow. Based on the analysis of the writing contents of the body of an annotation and the relationship created to the target, Zacklad [52] suggests there are three types of annotation:

- An *attentional-annotation* draws the attention of future readers of the document by emphasizing and pointing out some part of the document. For example, highlighted text, underlined text, symbols. This annotation facilitates the rereading by focusing on interesting parts of the document.
- An *associative-annotation* connects an existing element to another one. This annotation can be represented with arrows or references that are not necessarily located in the same document.
- A *contributive-annotation* is a new information created in reaction or in response of a segment of the annotated document. This annotation either complete this document or discuss it and it requires a link with its initial document by using an associative-annotation. The contribution is either added to the document or in the next edition of the document (which is a similar concept of the elaboration annotations proposed by Lortal et al. [54]).

To this list, we add descriptive-annotation like the semantic annotation as defined in the W3C [55], which consists in adding metadata to a document in order to make it easier to process by computer (e.g. indexing, researching, and looking for similar annotations) and make them interoperable between different systems. For example, a document is extended with structured information such as the author, the title, the creation date. G. Lortal in [56] qualify those annotations as “Computational level” annotation.

- **A Mean to Support Communication**

Annotations play an important role for the communication between diverse actors. In an iterative design process, the interaction these between actors will ultimately make artefacts and annotations themselves to evolve over time. Bringay et al. in [29] defines collaborative annotations as a way to help actors to communicate in a collaborative work to accomplish three main goals:

- *Annotation for editorial help.* Annotations can be used as a guide for the creation of a document by indicating instruction or constraints to respect for instance. They can be used as a set of guidelines for the creation of the document or in a revision of the document after a review. M. Zacklad in [51] note that the annotation can be used as constraints, suggestions, critics or evaluation criteria. Indeed, references to other constraints, decisions, ideas or suggestions can arise from the argumentation. M. Zacklad identify three attributes for those annotations: reference register (i.e. the domain in which the annotation is relevant like “economical” or “technical” for instance), abstraction level (which describe the level of details of the annotation similarly to the tacit/explicit dimension detailed in the section 3.4.4) and the materials in support (e.g. usage of example, reference document, alternative solution)
- *Annotation for argumentation.* Annotations can also be used to discuss and argue between collaborators about the document. S. Bringay compares these annotations with critical annotation as defined by Zacklad in [51]. G. Lortal in [54] uses the term discursive annotations and considers annotations as a discourse that express the point of view of the annotator that depends on its context which is defined by G. Lortal as the condition of production and reception.
- *Annotation for planning.* Annotations can be used to coordinate the project, to plan tasks to do and to manage the people working in the project. Three attributes are related to this type

of annotation: the task to carry out, the time to complete the task and the person or group of persons in charge of the task [52].

- **A Mean to Convey the Large Variety of Authors' Intentions**

Another classification of annotations refers to the authors' intention and/or activity carried out by the author of the annotation. In this respect, the classifications proposed by Naghsh et al [57] and by Agosti and Ferro [4] are worthy of mention. Naghsh et al [57] identify 6 different usages of annotations that match with the categories defined above:

- Clarifying and explaining the design.
- Verifying and requesting a verification from other designers or users.
- Exploring by asking questions to obtain more details on end users' needs.
- Altering or requesting an alteration proposed by the end users.
- Confirming and giving feedback on a design.
- Understanding by asking questions to the designers.

Agosti and Ferro [4] encompasses three goals:

- *Comprehension and study*. The intention here is to understand and to analyze the document. It can relate to attentional-annotations in which the annotator highlights parts of the document he found interesting or asks questions to help his comprehension. Those annotations do not add information on the document.
- *Interpretation and elucidation*. This usage refers to the annotations made to add information, to explain a document according to the annotator understanding in order to make it easier to understand and then to discuss about it. It could be an analysis or an argumentation for instance.
- *Cooperation and revision*. Annotations can also be used for sharing ideas and opinions about a text. This can be done through evaluation of the document, feedbacks on it or tasks planning for example.

On top of those classification, eight more functions defined by J. Virbel in [58] can be noted:

- *Prioritize*. Annotation can be used as a way to highlight the degree of importance of its target.
- *Architecting*. This function consists in enriching different targets of an artefact with the intent to give a structure or to categorize the targets by emphasizing or by making their nature more explicit (e.g. "definition", "example", "analysis", "part 1 of..."). Thus, those annotations can be considered as a form of semantic annotations.
- *Contextualize*. This function corresponds to the usage of annotations for giving inputs on the context of a fragment of artefact, thus facilitating its understanding.
- *Planning*. This function is similar to the goal defined by Bringay in [7].
- *Reformulate*. Annotations can be used to associate a variant or an alternative formulation of a fragment. This variation can either give a new perspective on the fragment or can be concurrent and give a contrary information (e.g. for correcting a fragment according to J. Virbel)
- *Comment*. This function is only defined by J. Virbel as the action to associate "comments" to fragments of the artefact. However, there is no details on the content of those comments. Thus, it is possible to assume that those "comments" refers to any kind of content that does not match with the other function identified.
- *Document*. Annotations can be used for attaching related documents or artefacts to a fragment such as an illustration or a diagram as explained by J. Virbel.

- *Correlate*. The last function identified in [20] is to link a reference of a fragment (which can be from the same or another artefact) and to optionally comment on the relationship between the linked fragments.

These functions are then sorted by J. Virbel into two categories: “characterization” for the first four functions and “attachment” for the last four functions.

- **A mean to support traceability of a project and of the rational design**

In [59], F. Shipman and R. Mccall have identified three perspectives to capture design rationale: the argumentation, the communication and the documentation. Each of those perspective have their own goals and approach.

The argumentation approach is based on structuring the design process. In the Issue-Based Information System (IBIS) framework, this structure is based around “issues” to solve with “positions”. For each position, a list of pros and cons are defined and a decision has to be made on which positions are accepted or rejected for each issue. This approach can benefit the reuse of knowledge from the indexing of the rationale which is made possible thanks to the structure given to the argumentation. However, as stated by Shipman and Mccall, this approach requires that the design team adapt their methodology to follow the structure defined.

The communication approach consists in recording and retrieving every communication made among the member of the project team on the different channels of communication used. This recording can then be used retrospectively for studying the evolution of the design process. The main drawback of this approach is that the information captured can be difficult to retrieve, to process, to exploit and to index due to their nature (e.g. video record of oral discussion with the writing on a whiteboard) and the diversity of medium involved (e.g. e-mail, instant messaging, printed documents, drawing, phone call records).

The last approach is the documentation. This approach is focused on noting information on each design decisions which include what has been decided and in which circumstances those decisions have been made (i.e. “when?”, “by who?”, and “why?”). Shipman and Mccall point out that only the core information of decisions, their results, and an immediate explanation of the rationale used for these results are worth recording for this approach. Thus, they are excluding the reasoning process, the details of the decisions, the alternatives, and dead-ends from this approach.

Regarding the support of the rationale design, Gruber in [60] made several observations and conclusions related to the design information that should be captured to support rationale. Those elements are presented in the Table 3-2:

N°	Observation	Conclusion
1	Rationales are based on many kinds of underlying information	Capture the information that is used to answer designers’ questions, not the information that fits a preconceived model of the design process
2	Rationales are constructed and inferred	Acquire data rather than answers
3	Rationales are not just statements of fact, but explanations about dependencies among facts	Capture dependency relationships
4	Rationale explanations refer to real engineering data and models	Capture data and models used in engineering practice
5	Rationales can be reconstructed from the relevant data	Capture weak explanations (just the relevant data), when a complete justification for a decision is not available

Table 3-2 Observations and conclusions on the information necessary for the rationale design in [60]

Overall, those conclusions point toward the storage of the core and necessary data of the design process that can be then reused afterward for future processing.

On top of this analysis, the review of papers on annotations showed that a few authors have identified the annotations as a mean to support the traceability of a project and of the rational design used:

- Boujut observes in [61] that it is important to keep a trace of the decisions made by designers regardless the moment (i.e. whether the decision was made during a project meeting or not).
- Lortal notes in [56] that annotations enable tracing the design rationale, decisions' follow-up. Moreover, Lortal emphasizes the fact that annotations can facilitate the memorization and the recalling of what happened through the "collective sensemaking" by promoting a certain and collective interpretation of the artefact as explained by Weick in 1979 [62].

Thus, annotations could be used as a flexible and versatile tool for supporting both the traceability and the design rationale of the design process similarly to the communication perspective or the documentation perspective. Indeed, annotations can be used to structure and organize contents that can be attached to any artefacts. Thus, they can be used to record key decisions during the iterations of the UCD and be used as a reading assistant on relevant artefact. Moreover, annotations can also be used as a communication medium for arguing. This versatility of the tool however implies a rigorous usage, management, indexation, and versioning to produce an exploitable resource for future usage and retrieval.

3.4.3. Synthesis of annotations definitions

Overall, the different visions of uses of annotations given by Agosti and Ferro, Naghsh, and Virbel are providing three different perspectives upon which annotations for conveying intentions can be observed. These three different perspectives are compatible and does not contradict each other. Thus, it is possible to consider them to be overlapping categories. An analysis of the compatibility of those different types of annotations for expressing the different intention can be found in the following tables:

	Comprehension and study	Interpretation and elucidation	Cooperation and revision
Clarifying and explaining the design	No	Yes	No
Verifying and request a verification from other designers or users	Yes	No	Yes
Exploring by asking questions to obtain more details on end users' needs	Yes	No	No
Altering or requesting an alteration proposed by the end users	No	No	Yes
Confirming and give feedback on a design	No	No	Yes
Understanding by asking questions to the designers	Yes	No	No

Table 3-3 Compatibility between M. Agosti and N. Ferro's classification with AM. Naghsh's classification of annotation

	Attentional	Associative	Contributive	Editorial help	Argumentative	Planning
Prioritize	Yes	No	No	Yes	No	Yes
Architecting	Yes	Yes	No	Yes	No	No
Contextualize	No	Yes	No	No	No	No
Planning	No	No	No	No	No	Yes
Reformulate	No	No	Yes	No	Yes	No
Comment	Yes	Yes	No	No	No	No
Document	No	Yes	Yes	No	No	No
Correlate	No	Yes	Yes	No	No	No

Table 3-4 Compatibility between AM. Naghsh's classification with J. Virbel's classification of annotation

	Comprehension and study	Interpretation and elucidation	Cooperation and revision
Prioritize	Yes	No	Yes
Architecting	Yes	No	No

Contextualize	Yes	Yes	No
Planning	No	No	Yes
Reformulate	Yes	Yes	Yes
Comment	Yes	Yes	Yes
Document	Yes	Yes	Yes
Correlate	Yes	Yes	No

Table 3-5 Compatibility between M. Agosti and N. Ferro's classification with J. Virbel's classification of annotation

3.4.4. The case of freeform annotations

The following analysis is based on the work of C. J. Sutherland et al. in [63] which presents an analysis of freeform annotations over text documents such as HTML files, PDF files, program code or scanned documents.

It is interesting to note that freeform annotations impose specific challenges for their analysis and processing due to the liberty they offer in term of expressiveness. Indeed, anything can be drawn on a text document. However, the problem is that freeform annotations can be seen as a composition of a set of strokes that should be considered as a whole.

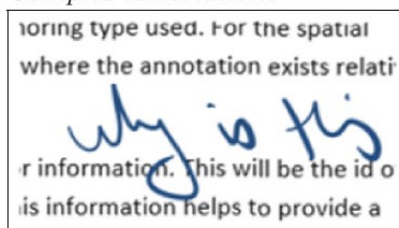
- **Taxonomies of freeform annotations on text documents**

In [63], C. J. Sutherland et al. defines a taxonomy of annotations based on the relative size of the annotation with the text and the purpose of the annotation. As a result, they identify the following categories:

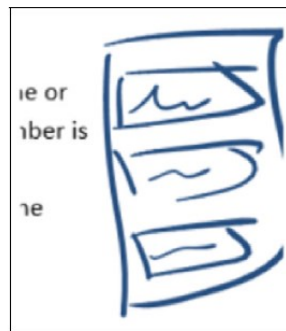
- Single line (e.g. underlines, highlighting)
- Multiple line (e.g. enclosures, margin bars)
- Connectors (e.g. arrows)
- Complex (e.g. text/symbol, drawings, marginalia)
- Commands (i.e. if the annotation is for a person or a computer)

The Figure 3-3 extracted from [63] illustrates a few example of freeform annotations.

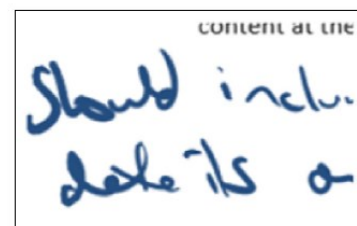
Complex Annotations



Text/symbol



Drawing



Marginalia

Figure 3-3 Example of freeform annotations (figure taken from [63])

On top of these different types of annotations, C. J. Sutherland et al. define a taxonomy of annotation support operations which list operations that can be made on freeform annotations:

- "Adding operations" regroup the following operations: Grouping, Recognizing, Anchoring, and Storing
- "Adapting operations" regroup the following operations: Repositioning, Refitting, Orphaning/deleting

“Adding operations” are operations that can occur when a freeform annotation is created over a document while “Adapting operations” concerns operations that can occur on existing annotations after the evolution of the target.

- **Targeting and anchoring systems**

C. J. Sutherland et al. identify 3 approaches for anchoring annotations over its target.

The most common approach identified is the use of a bounding box in which the document is seen as a graphical representation. The annotations are then placed using coordinates on this bounding box. The identified drawback of this approach is that it does not support the evolution of its target.

The second approach identified is the use of words written in the document as anchor. For this approach, several variations can be used: anchoring the annotation to the word, using numbers as identifier (e.g. from the 5th word of the document to the 10th) or giving the location by describing it (e.g. 5 first words of the second paragraph of the third page). As stated by the authors of [63], this anchoring method “support the reflow of the text” which means that changes such as the modification of the font-size does not impact this anchoring system.

The third approach identified is the use of the structure of HTML documents to describe a fragment. For instance, the id of the element or the path of the root element can be used to select the fragment within the document.

- **Evolution of artefacts**

In their study, C. J. Sutherland et al. acknowledge the fact that annotations are created over documents that evolve over time which might impact annotations. Thus, C. J. Sutherland identifies three different types of changes that can occur on text documents: “none”, “layout-only”, and “layout-and-content”.

Depending on the type of change that occurred on the document, the consistency between the freeform annotation and its target can be questioned. The authors of [63] note that the effectiveness of annotation repositioning is related to the anchoring mechanism used. However, they also note that repositioning annotations and adapting operations raise several issues with adding operations, and especially with the grouping and combining of multiple gestures into a single annotations.

3.5. Other uses of annotations on digital artefacts

3.5.1. The different types of digital annotations

Digital annotations differ from annotations on physical artefacts. Indeed, digital annotations gather three types of artefacts and data.

The first category of digital annotations are annotations for automatic processing. These digital annotations include annotations such as Java annotations, Semantic annotations [64], or automatic image annotations [65]. These annotations are acting as an enrichment of the digital artefact by specifying metadata. These metadata are used for various purposes: facilitate the processing of the artefact, indexing the documents or parts of the document and so on.

The second category of digital annotations refers to digital notes. Digital notes allow users to capture, create and share any piece of digital information. These digital notes are supported by note taking tools such as OneNote or Evernote.

The third category of digital annotations is closer from the annotations created during the activities related to text documents. These annotations include any digital content created using the tools and features provided in editors. Indeed, many software editing artefacts (e.g. Microsoft Word, Adobe

Acrobat Reader) include a set of “Comment” features allowing users to mimic the creation of physical annotations on physical artefacts. Thus, these “Comment features” generally includes highlighter tools, texts areas. The details of tools supporting annotations will be given in the Chapter 4.

This chapter will not include the study of the annotations belonging to the first and second category. Indeed, digital annotations for automatic processing does not involve any users: their study is not relevant for the integration of annotations in a UCD approach. As for the second category of digital annotations, notes tend to be used for a personal use and often features information not related to other targets. Thus, they will not be included to this study.

3.5.2. Findings on the use of annotation for the UCD process

In a UCD process, the fully-fledged system is the result of iterations of the design and the evaluation of the artefact produced. In these iterations, the development team gathers information about users to produce artefacts representing either the information collected, an analysis of these information, specifications, prototypes of the interactive system, or the interactive system itself.

Along the development process, design decisions are made based on the experience of previous project, the results of the evaluations, on the knowledge of the context of use of the interactive system, on constraints and requirements of the project. These decisions are influencing the following steps of the design process.

Similarly to the usefulness of annotations on paper-based documents annotations, annotations of digital artefacts might have multiple uses during the development of digital documents and user interface prototypes. Gutierrez et al. [66] conducted a study to investigate how annotations could affect the development of interactive systems in a UCD process. For the purpose of that study, they developed an independent tool called Helaba, which allows the design teams to store and to organize artefacts in a common workspace and to connect their decisions to them. The ultimate goal was the support of the traceability of design along the design process. Annotations are materialized by Heleba by the means of “Decision cards”, “Notes” and “Conversation thread” that can be attached as references to artefacts in order to add content or discuss about an artefact or a specific part of it. They observed that participants stored a curated selection of artefacts that were representative of the process in the shared workspace during the different activities of the UCD and that the annotations provided in her tool were used by the participants of the study to “build a narrative of their design process, especially in relation to how artefacts linked to each other”. In her study, Gutierrez noted that annotations were used:

- To record the results of discussion including the outcome of those discussion, decisions and upcoming tasks.
- To communicate and inform other team members of the work done.
- To gather internal and external feedback on artefacts stored in the workspace.
- To conduct usability evaluations by documenting information and by recording conversations members of the design team.
- To remind and to justify choices that were made during the process “in the late stages of the project”.
- To help to document the design choices by describing them retrospectively.

Overall, this study showed that the usage of a shared workspace and of annotations were used through the different activities of the design process (User analysis, Task analysis, Lo-Fi prototype and Hi-Fi prototype). Nonetheless, Heleba works as a repository of artefacts and annotations that are not directly connected to the tools used to build the artefacts.

Thus, while many tools exist for prototyping interactive systems, we believe that annotations should also be supported in those tools so that design teams can enrich their prototypes and communicate on it.

3.6. W3C Web Annotation Data Model

In [50], Li et al. have defined a classification of annotation approach for Computer-aided Design. This classification identifies the following categories of attributes that complete the specification of annotation: targeted media, audience, rendering system, usage and function, representation, and storage location. This classification of annotations brings another complementary view of annotations.

The Web Annotation Data Model was created for specifying a model and a format to ensure the sharing and the reuse of annotations across different hardware and platforms. This model is defined by a set of classes, properties and relationship with other resources on top of the core concepts of the annotation (i.e. the body and the targets).

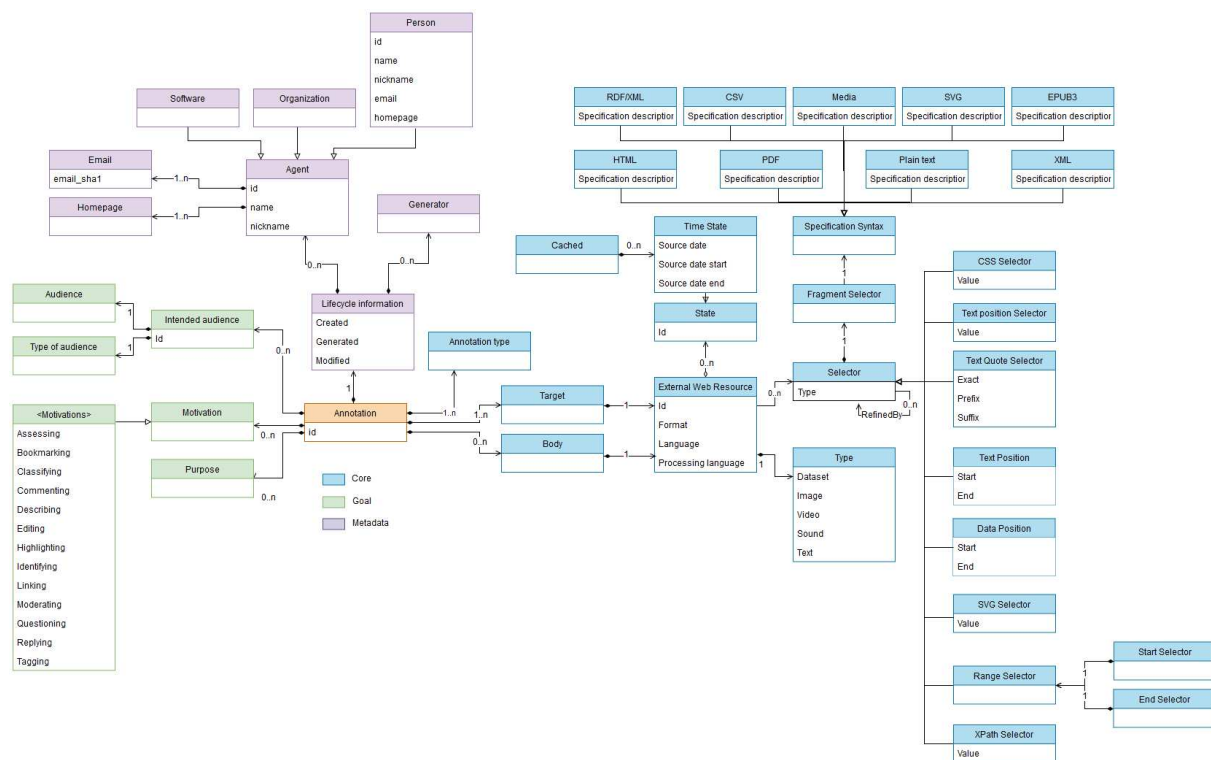


Figure 3-4 Representation of the Web Annotation Data Model

The Figure 3-4 is a partial representation of the W3C Web Annotation Model using the UML modeling language. This model is showcasing the attributes and entities in relationship with the annotation represented in orange in the center of the figure. From this representation, it is possible to identify three main groups of entities represented in the figure with a different color. These elements are explained hereafter.

3.6.1. The Core of the annotation

The first group corresponds to the structure of the annotation called “Core” and is represented in blue. This group of entities defines the type of the annotation, the target(s) and the body.

- **Presentation of the Target**

The target refers to the artefact related to the annotation and is always defined as an “External Web Resource” identified by an IRI and by a type (e.g. Dataset, Image, Text, Video). The targeting of the external web resource can be specified with a “State” (e.g. a time state) and a “Selector”.

As explained in [33], web resources are evolving over time and an annotation might be related to a particular state of the web resource. Thus, the state of the target is used to describe how to retrieve the intended state or format of the target by the consumer user agent of the annotation.

The selector is used when necessary to refine the targeting of the web resource to a fragment or set of fragments of the web resource. Thus, the annotation can be associated only with parts of the target that are relevant to the annotation. The selector is dependent on the type of the targeted resource. For instance, the selector of an annotation targeting a fragment of text will be different from the selector targeting a fragment of an image.

- **Presentation of the Body**

The body of the annotation refers to the content of the annotation that is related to the target. The body can be embedded within the annotation description or represented by an External Web Resource. An annotation can feature none or several bodies depending on annotations. This cardinality is used to specify annotations that does not require any body such as highlight or bookmark, to specify annotations with several bodies of different types (e.g. Image, text) or to specify alternative bodies (e.g. a set of body for supporting the localization of the annotation in several languages).

3.6.2. The Metadata of the annotation

The second group of entities presented in the Figure 3-4 is the “Metadata” represented in mauve. This group of entities specifies information on the annotation itself. The metadata regroup information on the lifecycle of the annotation which include its creation, its generation and its editing.

The creation of the annotation is done by an “Agent” who can be a person, an organization or a software. After its creation, the annotation can be generated by a generator for its serialization.

3.6.3. The Goal of the annotation

The third group of entities of this model of annotation is the “Goal” of the annotation represented in green. While this group of entities can also be identified as metadata, we decided to separate them from the lifecycle information to emphasize their focus on the intended usage of the annotation.

This set of metadata defines the “Intended Audience”, the “Motivation” and the “Purpose” of the annotation.

The audience of the annotation identifies the people who are concerned with the annotation. This identification can be done by defining a group or a geographic area for instance. The Motivation is defined by a set of instances describing why the annotation has been created (e.g. assessing, bookmarking, identifying, etc.). The purpose is similar to the motivation but it is used to justify the inclusion of a textual body in an annotation.

3.7. Conclusions

This chapter reviewed the literature for studying the annotations and their uses on both textual documents and digital documents. From this review, we noted that there is a kind of agreement on the structure of an annotation (i.e. body, target and a link). This generic structure matches with the different types of annotations we observed in the various papers including the representation of the body (e.g. highlight, sketches, text), the different targets and the different linking mechanisms (whether explicit or implicit). We also noted various dimensions that can be used to qualify an

annotation. However, we did not find any formal definition of annotations that includes the definition of all the attributes of an annotation. This might be explained by the variety of representation of the body and linking of annotations.

Regarding the uses of annotations, we noted in the literature that there were several activities around documents and depending on these activities, the goals and purposes of the annotations can differ. These different activities include the following:

- The active reading in which annotations are used for facilitating future readings, analyzing, memorizing, and studying a document
- The creation of a document in which annotations are used for explaining, suggesting, collaborating, contributing and discussing
- The review of a document in which annotations are used for giving an opinion, for understanding

On top of these goals, we can add the use of annotations for planning and discussing which can be done independently from the activity.

After this review of annotations, we refined the study to digital annotations on electronic artefact for their potential use on the artefacts produced and used during the design process.

The W3C Web Annotation Working Group has published on the 23rd of February 2017 three different Recommendations for annotations on the web: the Web Annotation Data model [33], the Web Annotation Vocabulary [34] and the Web Annotation Protocol [35].

These standards specify a model for independent annotations and a protocol for supporting these annotations. As mentioned in [35], these standards are supported by many projects and tools such as MIT's Annotation Studio [36], Hypothes.is [37] or Pundit Annotator [88].

This support is done using "Annotator" [38] which is a JavaScript library allowing to manage annotations on any website. This library features 27 plugin [39] that allow to enrich the support of annotations in various ways such as UI enhancements (e.g. "annotator-imgselect" for selecting portions of images and annotate them), its integration to other environments (e.g. "Annotator Wordpress Plugin" for Wordpress websites), or for enriching annotations (e.g. "Geolocation-annotator" for adding the user position with HTML5 geolocation on annotations).

While this standard is currently being adopted by many organizations, this standard is dedicated to the annotation of web documents. Thus, this standard cannot be adopted for the integration of annotations on design artefacts of the UCD process.

Chapter 4. Overview of tools supporting annotations and of prototyping tools

Summary

This chapter presents an overview of the existing tool support for dealing with annotations as well as an overview of prototyping tools through two reviews. One of the review, namely the review of prototyping tools, has been previously published in the Journal of Software Engineering and Applications in 2017 [68]. This chapter is oriented toward the annotations on design artefacts and on prototyping tools due to the importance of prototypes in the UCD process.

The first part of this chapter consists of the presentation of the review on annotations tools and their support in prototyping tools. This presentation includes a brief introduction of the designers needs regarding annotations, a presentation of the methodology used for this review and an analysis of the results.

The second part of this chapter gives a presentation of the review on prototyping tools. Similarly to the first review, this part includes the presentation of the designers needs regarding prototyping tools, the presentation of the methodology, a brief presentation of the results and an analysis of these results.

The results of these review will be used as a base for designing a model-based approach for integrating annotations within the project workspace in the Chapter 6.

4.1. Introduction

The tool support for annotations can be sorted into two categories: standalone tools that allow to create annotations over a specific type of artefacts and integrated annotation support within artefacts editor. Thus, we performed two studies for analyzing the annotations tool support.

The first study consists in a comparative study of annotations tools. We are particularly interested in how tools support annotations for their usage either as a communication medium, as a contribution of the creation of an artefact or as an organizational tool in a context of the process of designing an interactive system in which a design team builds several prototypes featuring alternative designs, newer versions of one design and other artefacts related to the interactive system or the end-users (e.g. task model). On top of the creation of annotations, their sharing and the retrieval of the context of the annotation (e.g. identify its author, the relevant artefact or part of the artefact, identify the moment of the creation of the annotation), we wanted to find out if tools implement other features on annotations to handle the management of annotations for sorting them, processing them and storing them. A synthesis of the results of this analysis can be found in the Annex 3.

The second study will focus on the prototyping tools. On top of the study of annotations features integrated in those tools, this review also includes an analysis of a larger scope of features such as the support of prototyping the different aspects of the UI of an interactive system (i.e. dialog and presentation). This review has already been published in 2017 in the Journal of Software Engineering and Applications [68]. This study is mainly focused on prototypes and prototyping tool since prototypes are the recommended artefacts to use during the UCD process. To illustrate the importance of prototypes in the UCD, we can note that Garcia et al. [69] found in their studies that prototypes are the most used artefacts in the context of communication between Agile methods and User-Centered Design process. Moreover, in this study, Garcia et al. does not count mockups, sketches, and wireframe as prototypes but as distinct artefacts. Those prototypes are also the most used artefacts during what they call the “Iterative Cycle” which consists in the development and design tasks.

4.2. Review of annotation tools

In this section, we will only analyze annotations tools that don't have any support for editing prototypes and artefacts used along the design process. Indeed, since annotations are the main concern on these tools, we expect to find a different support of annotations from their support in prototyping tools. Thus, this study is focused on the analysis of features and attributes specifics to annotations.

During this review of the annotations, we noted several lacks regarding the integration of annotations on digital artefacts in the context of the design of interactive systems. The analysis of the results of this study is given in the section 4.4 which also contains results from the analysis of annotations in prototyping tools. Indeed, we identified similar findings in both categories of tools.

4.2.1. Selection of tools

We reviewed a total of 25 annotating tools listed in the Annex 1.

The selected annotating tools regroup both academic and industrial tools related to the annotation of artefact by a user for communication purpose. In those 25 tools, 14 annotations tools were retrieved from papers published in CHI conferences from 2009 to 2017 which presented annotations features. 7 tools were picked from older publications referenced in those published papers in the CHI conferences. 4 tools were picked from IAnnotate conferences from 2014 to 2017.

4.2.2. Criteria used for the analysis of annotations tools

For this analysis, we were interested in the following questions:

- Does the tools feature the structure of annotations found in the literature (i.e. body and target)?
- How does tools manage the evolution of the targeted artefacts?
- Does tools provide ways to support and specify the uses of annotations?
- What features are proposed by tools for managing the annotations?

Thus, we defined a set of topic and features of interest to inspect the tools, as shown in the Table 4-1:

Topic	Features investigated
Targeting support	Identify the supported target by the tool
	Identify the method used to identify the target of the annotation
Temporal evolution of artefacts	Identify the support of the evolution of the artefact and its impact of the annotation
	Identify the support of the handling of the lifecycle of the annotation
Type and semantic of annotations	Identify the type of annotation proposed by the tool
	Identify the support of the possible existence of a semantic for annotations in a tool or the ability to specify a semantic
	Identify the support of the customization features proposed to appropriate the tool
Annotation management	Identify the feature for the management of the growing number of annotation
	Identify the feature for the visibility issues when displaying both the annotation and the artefact
	Identify the feature for the cleaning of irrelevant annotations
	Identify the support of mechanisms to weight the importance of annotations between each other (e.g. based on the role of the author of the annotation in the design process or the frequency of a feedback)
Collaboration support	Identify the tools that feature a collaboration support
	Identify the features associated with the collaborative aspect of the tool

Table 4-1 Investigated topics and features for the state of the art on annotation tools

In order to fill this analysis grid, we retrieved information from three main sources of information for each tool when available. The first one is the publications in which the tool is presented for academic tools. Each publication retrieved from CHI and IAnnotate conferences has been analyzed in order to find if the different features and concepts related to our grid of evaluation was implemented or

intended for the annotation tool. The second source of information was the tool itself in its usage. When available, the tool presented has been tested in order to get a view of features that might not be presented in the paper. Lastly, we consulted online documentation on the tool website when available which consists in tutorials, features presentations, videos, help pages and community forums.

4.2.3. Results of the review of annotation support

4.2.3.1 Targeting support

Among the 25 annotating tools we reviewed, 12 were dedicated to annotate HTML pages, 8 were annotating text file. The remaining tools were used to either annotate images or a visual representation of data.

In 19 of the tools the user can annotate a fragment of the artefact and only 4 tools supported the annotation of the artefact as a whole while the one remaining tool was not specified how it handled the targeting selection.

6 tools allowed the linking of several targets to one annotation.

As for the selection of the fragment, the trend was to create a selection of the text that was related to the annotation. This feature was available in 16 tools. In the other tools, the selection was made by defining an area for 5 tools, positioning the annotation for 3 tools. Other method of selection consisted in citing the text or associating the annotation as a property of the text.

19 out of the 25 tools prompt the user to select the fragment of artefact first and then create the annotation while the others consisted in creating the annotation first and then attach it to the fragment.

4.2.3.2 Temporal evolution of artefacts

- **Evolution of the targets**

One of the particularities of the context of the design of an interactive design is that the annotations are made on artefacts that are not frozen in time and those artefacts evolve through the iterations. Moreover, the context of the artefact is also evolving through the production of new artefact, acknowledgement of new information on the interactive system, the evolution of the user's needs, the current management of the project etc. Thus, the actual context of an annotation may not be available at all time if this context is not versioned.

While the context of an annotation can give a meaning or help understanding an annotation, it can also affect its validity (e.g. a note indicating a task to perform, a highlight of an error that has been fixed).

5 out of the 24 annotating tools acknowledge the problem of the evolution of the artefact by either extracting a snapshot the fragment targeted by the annotation to dispose of a static target (e.g. Diigo [70]) or by trying to recognize the initial target of the annotation in the new version of the document (e.g. Annozilla [71]).

- **Lifecycle of the annotation**

3 of the annotating tools has considered the lifecycle of the annotations. In Authorea [80] and Protonotes [81], user can set a status of an annotation to mark it as "Unsolved/Solved" or "Not Reviewed/Reviewed" and "Not completed/Completed" while in Elias' prototype [82], it is possible to define rules that will control the expiration of the annotation based on a time of expiration or on data.

4.2.3.3 *Semantic of annotations*

- **Type of annotations according to the structure of the body**

The main structure of the body supported was the textual form. 23 out of 25 tools featured a way to add a text as an annotation.

Regarding the graphical body for annotations, we noticed that the most adopted annotation for annotating tools was the highlighting as a form of annotation which is supported by 10 tools. The second most adopted type of graphical body is the freeform sketches which was available in 6 tools.

As for other type of structure of body for annotations, we identified 7 tools that allows to make a reference to other artefacts through the use of hyperlinks, or excerpts such as LiquidText [42], GatherReader [77] or Ponga [78].

3 annotating tools featured a semantic annotation for linking parts of the artefact to an external definition: Neonion [86], Domeo [87] and Pundit Annotator [88].

- **Semantic given to annotations**

During our review, we found out that tools specify a limited semantic to annotations using tags (e.g. Amaya [89], Quilt [92], and Neonion [86]) to characterize and classify the different annotations.

Other than those tags, the annotations features were flexible and did not impose a semantic for a type of annotation. Indeed, users can define their own semantic by customizing annotations if they need to by defining their own tags, modifying the color used or the font (e.g. Amaya [89], Domeo [87]). However, there was no integrated support for documenting a custom semantic by users. Thus, this semantic is either implicit inside the design team or it is defined in an external document.

4.2.3.4 *Annotation management*

During our review, we noticed several features that are related to the annotation management.

The most adopted feature was the search of annotations which is supported by 12 tools. Following the search feature, we noted that 11 tools supported a filtering option and 10 tools listed the annotations created in the artefact.

The status of annotation as described in the section “Lifecycle of annotation” and usage of tags in the section “Semantic given to annotations” can also be used as a way to manage the annotations for sorting them, to reduce the number of annotations or to give a weight to one annotation.

We also noted that reviewed tools provided a navigation feature to easily retrieve an annotation or to retrieve a fragment targeted by an annotation when the annotations were not located directly on the artefact (e.g. navigating from the list of annotations). Thus, 16 tools supported the navigation from an annotation to its target and 13 tools supported the navigation from the target to the annotation.

4.2.3.5 *Collaborative support*

Among the tools reviewed, we noted that 21 annotations tools supported multiple users and 14 tools provide a way to identify the author of an annotation. In 13 tools, it is possible to restrict the access to annotations by defining users’ rights.

Other features for the collaboration over a document have been found during this review. A notification system is proposed by several tools such as Dokieli [45], Hypothes.is [37], Authorea [80], or Quilt [92].

4.3. Review of prototyping tools

This section presents our previous work on the state of the art of GUI prototyping tools initially performed in 2016 and gives an overview of its analysis which was published in an article in 2017 for the Journal of Software Engineering and Applications (JSEA) [68] called “A comparative Study of Milestones for Featuring GUI Prototyping Tools”. All the details of the study as well as their results can be found in this article.

This study focuses on the design of desktop applications and does not cover the prototyping of interactions devices nor prototyping of new interaction techniques. On top of that, this review covers only needs related to the creation of a prototype within the design process. The needs of a designer regarding the usage of the produced prototype during the evaluation activities of the design process will not be examined. Indeed, even though the prototypes are produced for various purpose (e.g. exploration, communication, sharing of ideas, getting a feedback, validating a design, getting a proof of concept), the goal of this review on prototyping tools is to provide a comparative overview of the different prototyping tools available.

This study of prototyping tools is completed with a more detailed analysis on annotations support in the section 4.3.5. This extension of study is based on the criteria identified in the section 4.2.2.

Overall, this state of the art has been used in this PhD thesis in order to identify features and gaps in GUI prototyping tools regarding the traceability of the evolution of the design of interactive prototypes.

4.3.1. Selection of tools

For this state of the art, we decided to investigate a broad range of GUI prototyping tools to get a global vision of tools currently available as well as a vision of the evolution of those tools. Thus, we investigated both academic and commercial prototyping tools.

On one hand, academic tools represent a proof of concepts to support scientific claims raised by the research and thus, their development were heavily focused on a few selected aspects of the prototyping activity. On the other hand, commercial tools with their releases show that they have a sufficient degree of maturity in order to be used on a large scale. On top of that, they are evolving through regular updates. An analysis of available features of those released tool as well as their advertisement is giving a look of what features are favored and important for the commercial groups who owns the prototyping tool.

The main features we are interested in are as follow:

- Features for designing a prototype which includes the specification and the representation of the dialog and the presentation of the interactive system to represent
- Annotations as a mean for enriching the specification on top of the features dedicated to the design of the dialog and the presentation
- Version control features and tracking of the evolution of the prototype for the investigation of traceability of the design process

For the selection of the tools, we examined publications of main HCI conferences since 1982 for the academic tools and looked available tools on the web for commercial tools. Thus, we listed and examined a total of 121 prototyping tools and among those tools, there is 17 academic tools.

As mentioned in [68], the selection of academic tools was made in 2016 by reviewing papers that propose or describe prototyping tools and which were submitted on HCI conferences sponsored or co-sponsored by ACM, IEEE and/or IFIP since 1982. For these conferences, we only considered English

proceedings available in digital form. Thus, we considered the following conferences: ACM CHI (1982-2016), ACM UIST (1988-2016), ACM DIS (1995-2016), ACM EICS (2009-2016), IFIP INTERACT (1984-2015). For these conferences, we looked for papers that contained the following keywords in their title and/or abstract: prototype, prototyping tool, prototyping interface, wireframe, wireframing, sketch, sketching, draws and drawing. The selection was then refined by excluding papers presenting tools for specific environments (e.g. sketches of building for architects) and by limiting the selection to tools for user interface prototyping. The final selection of academic tools includes 17 papers.

Regarding the selection of commercial tool, this was made by using a search engine and looking for tools matching keywords like prototyping tool, wireframe, sketching. To refine this selection, we searched for information on the tool to answer the following questions: “Is the tool a standalone software or an extension/library/framework?”, “Is prototyping generic interfaces possible?”, “Is there a free trial of the tool?”, “Is the tool still updated and documented?” and “Does the prototype produced with the tool support any interaction?”. This refinement lead to the exclusion of some tools since some of them were libraries or stencil themes that could be used in paper-based prototyping or were no longer updated nor documented for years. More details on this refinement of the selection of commercial tools is available in [68]. We also included in the analysis tools which are not dedicated to prototyping but were used as such which includes tools like Powerpoint and Photoshop. The final selection of commercial tools includes 104 prototyping tools.

The extended study on annotations has been applied to the 56 prototyping tools that support annotations features. The results and the analysis of annotations in prototyping tools is featured in the section 4.3.5.

4.3.2. Criteria used for analyzing prototyping tools

In this step, we identified and grouped several criteria to inspect on prototyping tools. The results of the inspection of prototyping tools based on those groups of criteria allowed us to classify the different tools. The groups of criteria we used are as follow: description of the tools, features for specifying the prototype, features for the preview of the prototype, and management of the versions.

Regarding the extended study of prototyping tools focusing on annotations, we used the same criteria described in the section 4.2.2. To these criteria, we also added an analysis on how the creation of annotations is enabled since the annotations support is a secondary feature to prototyping tools.

4.3.3. Source of information collected

The data we collected during this review of prototyping tools came from several sources.

The first source of information we considered is the online documentation available on the tool website and on the internet. This documentation includes presentation of the features supported by the tool, news presenting the support of new features, tutorials, F.A.Q., videos, and community forums.

When available, the prototyping tool was manually inspected to investigate the different features and their implementation. While this inspection was often limited to free trials of the tools when they were available, the online documentation completed the information required to fulfill the analysis.

4.3.4. Results of the review of prototyping tools

4.3.4.1 Features provided by prototyping tools for the specification and the design

The features provided by prototyping tools for the specification and the design of the interactive system is divided into two parts. The first part covers features related to the design of the UI of the prototype. The second part covers features associated with the specification of the dialog of the

prototype. Overall, we observed from our study that the ability to create a prototype without programming skills is a well-established feature.

- **Features for designing the UI**

At the end of our analysis on prototyping tools, we noted two ways to design the UI of the prototype: pen-based interaction supported by 12 tools, usage of widgets which is supported by 93 tools.

- **Features for designing the behavior of the prototype**

In most of the prototyping tools we studied, most of the behavior specification were limited to the navigation from one screen to another triggered by a mouse click. However, we identified three ways to add the interactivity on prototypes: usage of hotspots on images, events handling on widgets and scripting in models.

While the ability to specify the dialog was mostly adopted by prototyping tools, only a few of them proposed a representation of the link between the different screens of the prototype.

4.3.4.2 Version control features provided by prototyping tools

In the UCD, design teams are iterating over a prototype until their design solution is satisfactory for each actors of the design process. Many prototypes are produced or edited depending on the results of the validation of the prototype, on the feedback from users, on the emerging need and so on. On top of that, the creation of a prototype can be a collaborative activity within the design team.

Thus, version control could be applied to prototypes and to the prototyping activity. By doing so, the design team will be able to manage the different versions of the prototypes produced during the design process and control the evolution of the prototypes.

During our review of the tools, we noted that 29 prototyping tools proposed a version control feature.

A few tools like Concept.Ly also featured a comparison tool used for a side-by-side comparison of two version of one screen of the prototype.

4.3.5. Results of the review of annotations support in prototyping tools

Annotations is a medium used to add information on a given artefact. The annotations can be materialized through different forms, they can be used for various purpose and the information contained in annotations is only limited to the expressivity of the type of annotation.

In the different prototyping tools we reviewed, we noted that 56 out of the 121 tools supported annotations features.

4.3.5.1 Creation of annotations

In those 56 tools, we were able to identify 3 different stages in which the annotations could be used: Prototype Building, Annotation Mode and Usability Testing. The creation of annotations during the Prototype Building is the most adopted stage by prototyping tools.

In this stage, prototyping tools provide two ways of annotating the prototype. The first method of annotating is by adding widgets on the prototype the same way as the other widgets representing the UI of the interactive system. The second method of annotating the prototype is to edit a dedicated property of either a page of the prototype or a component of the prototype.

The second stage in which annotations features are available correspond to a specific mode provided by some of the prototyping tools we reviewed. In this mode, the prototype cannot be edited but tools for creating annotations are available.

The last stage we identified correspond to the test of the prototype. In this stage, data is collected from users who test the prototype. Those data are then stored as annotations on the prototype for a further analysis.

4.3.5.2 Targeting support

We noted that all 56 prototyping tools featured a view of the presentation of the prototype but only 10 of them featured also a view of the dialog of the interactive system. It should be noted that even if only 10 tools feature a visual representation of the dialog, other tools still provide a way for specifying it. All of the tools studied allowed the user to annotate the presentation of the prototype. Even if the dialog is not represented, it is possible to comment on it by adding explanation on how the system should react for instance.

14 tools allowed the linking of several targets to one annotation.

As for the selection method, 48 tools used the geographical proximity with markup or pin associated to the annotation with the relevant part of the prototype to make the link between the annotation and the fragment. Other method adopted by tools to link the annotation to the fragment were noted like the use of a shape to delimit or point to an area in 21 tools and a dedicated property of the graphical widget for writing notes was featured in 7 tools.

It is interesting to note that contrary to annotating tools, the annotations are created first in prototyping tools and then they are connected to the relevant part of the artefact if required. Indeed, 43 tools feature the creation of the annotation first, 10 tools support the selection first and 3 tools support both methods.

4.3.5.3 Temporal evolution of artefacts

- **Evolution of the target**

Only 4 out of the 56 tools reviewed considered this aspect of the artefact though the features associated with it were limited to specify a status for the target. Thus, InVision [72], NinjaMock [73] or Notism [74] allow the designers to set a status on pages of the prototype by choosing one of the proposed status (i.e. "In progress", "Review requested" or "Design approved"). Silk [79] on the other hand provide the history of the design.

This feature is rather limited in regard of the impact on annotation since the designer have to manually manage annotations when the artefact evolves and forces users to create a backup of both annotations and artefacts if they want to keep a trace of the result of the evaluation for each iteration.

- **Lifecycle of the annotation**

16 of prototyping tools support the lifecycle of annotations by allowing users to mark an annotation as solved or by closing and archiving an annotation such as UxPin [41], Pidoco [47], InVision [72], NinjaMock [73], Notism [74] or ProtoShare [46]. LucidChart [83] features a history of the evolution of the status of the annotation that enrich the support of the lifecycle of annotation.

4.3.5.4 Semantic of annotations

- **Type of annotations according to the structure of the body**

The support of textual notes in prototyping tool is provided in the 56 tools we reviewed. However, only 23 of them allowed to create discussion threads.

Regarding graphical structure for the body of annotations, this review of prototyping tools showed that these tools were more oriented toward the use of callouts (e.g. pins, icons, or arrows) which is

supported by in 21 tools. Other graphical representations are available in prototyping tools: the support of the creation of various shapes (e.g. circles, rectangles, braces, and scratches) is included in 14 tools. As for the freeform sketches, it is only supported by 11 prototyping tools.

We also identified other types of annotation body though they were only adopted by a few tools: 3 prototyping tools allowed users to enclose files in annotations (JustInMind [84], MockFlow [85] and Notism [74]).

- **Semantic given to annotations**

Similarly to annotations tools, it is possible to specify a semantic to annotations using tags in prototyping tools such as iRise [90], Concept.Ly [91], Cacao [93]. These tags can then be used to sort the different annotations created by designers.

Like the annotations tools, prototyping tools does not specify a semantic for the different type of annotations. Thus, designers are also able to define their own semantic through customization of the annotations (e.g. Amaya [89], Domeo [87], and iPlotz [94]) although prototyping tools do not feature a way to document the semantic given by its users.

4.3.5.5 Annotation management

The review of prototyping tools showed that the features provided for annotations management were similar to annotations tools. However, the adoption of the different features were different.

Thus, the most adopted feature in prototyping tools for facilitating the retrieval of annotation was the use of a list gathering annotations. This feature was adopted by 19 prototyping tools (it was adopted by 10 annotating tools). The second feature is the filtering of annotations which is supported by only 6 prototyping tools (the filtering of annotations is supported in 11 annotating tools). The third feature which is the search support is adopted in only 3 prototyping tools. This feature was the most adopted method for retrieving an annotation in annotations tools with 12 tools supporting it.

Regarding the navigation, it is interesting to note the shift of concern compared to annotations tools: prototyping tools favors the navigation from the target to annotations with 17 tools supporting the navigation in this way and only 8 tools support the navigation from the annotation to its target. As a reminder, the number of annotating tools supporting for the navigation is 13 while the support for the other way around was 16.

4.3.5.6 Collaborative support

34 prototyping tools allow to collaborate over a prototype. In 28 tools, it is possible to identify the author of an annotations and only in 14 tools, it is possible to restrict access to annotations by defining users' rights.

The notification system has also been adopted by some prototyping tools similarly to annotations tools such as iRise [90], HotGloo [75], LucidChart [83], Moqups [76], NinjaMock [73], or Notism [74]. For instance, iRise sends notification upon the creation of an annotation or upon the creation of a reply on an annotation.

The analysis of prototyping tools showed a few features noticeable for the collaboration over annotations. For instance, it is possible to assign an annotation to a collaborator who will receive a notification in UxPin [41] and in LucidChart [83].

4.3.6. Findings on prototyping tools

4.3.6.1 The specification of the prototype is limited to the UI

The results of our analysis for the specification and the design of the prototype show that the main goal for prototyping tools is to provide an interactive preview of the application which can be designed with various fidelity depending on the tool used. However, the features for the specification of the behavior of the prototype are limited to the navigation for every tools and animation on a few exceptions.

4.3.6.2 Uses of annotations in prototyping tools

Overall, we noted in our study that the annotations were mainly promoted by prototyping tools as a tool for collaborating within the design team. Indeed, the presentation of the annotation features are focused on the usage of annotations for discussing ideas and giving feedback on design choices.

The usage of annotations for adding specification details on the prototype were never mentioned in the tools we analyzed. Even though it was not promoted by the different sources of presentation and tutorial of prototyping tools, it is still possible to use annotations to add a layer of informal specification which can cover both the presentation and the behavior of the interactive system.

4.3.6.3 Targeting support in prototyping tools are limited to their tools

For prototyping tools, an annotation can only be attached to one artefact and depends on this artefact. However, this artefact might represent one aspect of the design process that can be completed with another perspective or with complementary information that might be located in other artefacts.

While it is possible in tools to link several fragments to one annotation, several artefacts cannot be linked to one annotation. Even if it is possible to reference other artefacts loosely by citing them or enclosing a copy of the reference, annotations are still dependent of the annotated artefact. Connecting artefacts this way only gives a unidirectional relationship between those artefacts while the connection could be bidirectional. On top of that, this solution creates duplicates of the artefacts that might be referenced by other annotations for each annotation referencing them. Thus, if the targeted artefact is updated, this update should be reflected on every copy of the artefact contained in the annotations that could be affected by this update.

We suggest that while this referencing system is an interesting feature when working on several artefacts, considering annotations as an independent artefact in a design process context could provide a way to build bidirectional connections between the different artefacts giving different points of view on similar aspects of the interactive system. The idea is to integrate a system similar to semantic annotations but for the design of an interactive system. This semantic annotation for interactive system would be made manually by the design team through the creation of annotations.

Building this semantic annotations for the design system could benefit the design team by featuring ways to navigate through the different artefacts by exploring related descriptions of the interactive system located in different artefacts, to analyze the consistency of artefacts, to identify the repercussions of a modification or the addition of information on existing artefacts, and to help the design team during decision making by providing further insights when looking on one aspect of the interactive system.

Thus, annotations created within these tools are closed to the environment in which they are created. For solving this issue, the Web Annotation Data Model defines an annotation as an independent entity that can be related to several targets and to fragments within those targets. Thus, the annotations are not limited to one artefact.

4.3.6.4 The three different period of the prototyping tool history

By sorting the tools we analyzed by release date and by ordering them in a timeline, we were able to observe three main periods of interest in the history of prototyping tools as illustrated in the Figure 4-1 below extracted from our study [68].

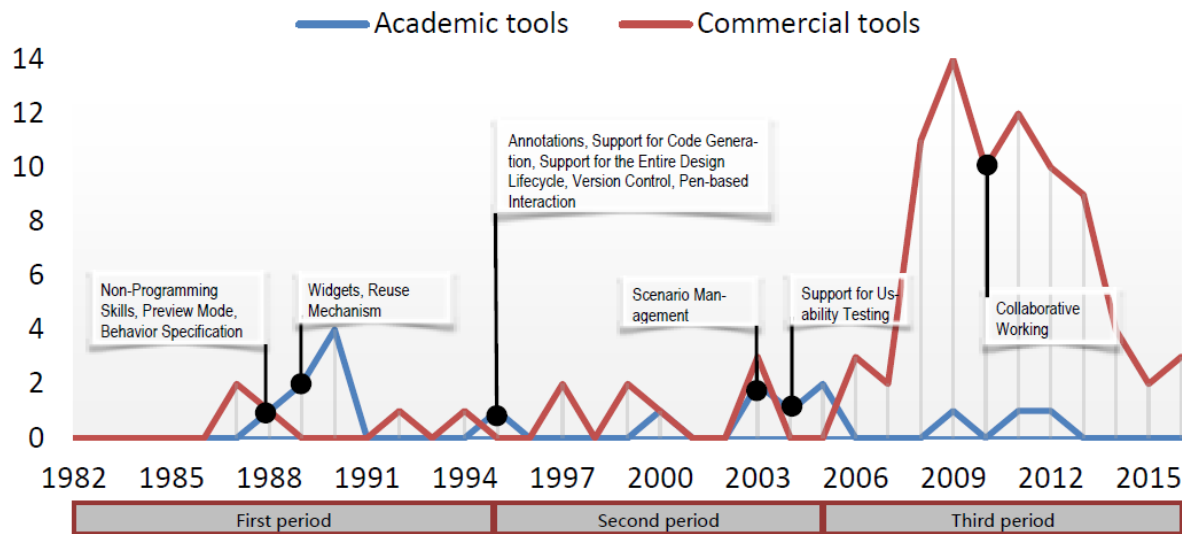


Figure 4-1 Number of both academic and commercial tools per year

The first period for prototyping tools corresponds to tools released before 1995 which coincides with the emergence of UIMS tools. Prototyping tools released before this year were dedicated to high-fidelity prototyping by using design elements from the final interface and were strongly dependent on the hardware. While the interface prototypes produced by those tools could be integrated in the application after their development and testing, UIMS tools are not suited for early stages of prototyping in which flexibility is needed.

The second period starts in 1995 and ends 2005. In this period, tools supporting the design of fully functional prototypes appeared. This period was also marked by an increase of interest in the diversity of way of creating prototypes like sketching coupled with a sketch recognition system as well as a new interest in prototyping the behavior of the prototype.

The third and last period we identified starts in 2006 and is currently in progress. This period is marked by a big increase of the number of commercial tools. This period is also marked by the appearance of features for collaborative working and by a trend in proposing prototyping tools as a “Software as a Service” (SaaS).

4.3.6.5 Analysis of the adoption of features by prototyping tools

In the review of the prototyping tools, we noted three main characteristics adopted by more than 70% of the tools: the ability to design prototypes without any programming skills, the use of widgets for building a prototype and a minimal specification of the behavior consisting in the navigation between pages of the prototypes.

The usage of widgets has been widely adopted by prototyping tools and provide a way to build prototypes from a library of existing and recognizable components that can be repositioned, resized and customized. However, the design of prototypes based on free-hand sketching was not adopted by a majority of prototyping tools since it is available in only 9.92% of the tools analyzed. This can be explained by several factors: the necessity to use specific devices for drawing accurately since mouse are not adequate for drawing, the difficulty to maintain drawn sketches, the difficulty to follow the

evolution of sketches through the design process, and the expectation of a higher fidelity prototyping for digital tools compared to paper prototypes.

We also noted that features associated to the whole lifecycle of prototype (e.g. Scenario Management, support for Usability Testing) were proposed by less than 10% of the tools and there was no significant increase in the adoption of those features.

From the results of this analysis, we prospect that features improving the user experience were promoted over other features related to the design lifecycle process and the models of the prototypes.

4.3.6.6 Prototyping tools on smartphone

The work described in this section presents a technical report in [96] which provides a view at glance on prototyping tools running on smartphone. This complementary study consists in a review of 30 additional prototyping tools and focuses on free applications available on Android smartphones in the Google Play store. This review exclude paid applications, applications downloaded less than 5 000 times, applications rated below 3 stars, applications not maintained for more than six month, model based prototyping applications, and is limited to application supporting the English localization.

The choice of studying smartphone applications and especially Android applications has been motivated by the large adoption of Android operating system which was revolving around 75% of the worldwide market share during 2018 according to StatCounter [97] and by the trends of providing mobile versions of web-based applications.

In this study, Naqvi has found a few differences with desktop tools. Due to the nature of the device, mobile applications dedicated to prototyping features different interactions techniques for creating and specifying prototypes.

Regarding the creation of prototypes, tools like Prott [98] and Marvel [99] let designers take a photo of sketches and use them as the presentation of the prototype. Other tools like Quick Proto [100] are based on free-hand drawing for creating prototypes, thus exploiting the interactive screen as a sketch canvas.

As for specifying the dialog, those prototyping tools rely on the specification of hotspots that will trigger the navigation through the different pages of the prototype. However, one of the difference with desktop applications is that some mobile applications such as Prott support the specification of several interaction techniques unique to mobile applications (e.g. tap, double tap, hold, swipe) as well as the specification of transition animations (e.g. push left, push right, slide left)

4.4. Other findings concerning both annotating and prototyping tools

4.4.1. Tasks for supporting annotations

As presented in the Chapter 3, there are many types of representation of annotations and many uses of annotations. From this review of the literature, we identified generic tasks that are performed when stakeholders are using annotations:

- Specify the body of the annotation to express themselves
- Examine the body of the annotation to understand its content, its meaning and its purpose
- Specify and identify the targets of the annotation (whether it is another annotation, an artefact, or a set fragments)
- Share the annotation for getting responses and feedback
- Update an annotation (for instance, indicating that a planned task has been done or indicating that a modification on the target has been performed)

4.4.2. Creation of annotations

During the analysis of annotation support on both prototyping tools (56 tools) and annotating tools (25 tools), we noted that there are two strategies concerning the creation of annotations. The first strategy is to create the annotation first and to attach the targets concerned by the annotation. This strategy has been adopted by 47 out of 81 different tools (see Annex 3) and 43 of these tools are prototyping tools. This approach has been adopted by tools like List-It [40], Balsamiq [102], or UxPin [41].

The second strategy consists in selecting first a fragment to annotate and then in creating the annotation. This strategy has been adopted by 29 tools and 10 of these tools were prototyping tools. Example of tools supporting this strategy are: LiquidText [42], Dokieli [45], or ProtoShare [46].

It is interesting to notice that annotations tools in which annotations are mainly made on existing and static documents focus more on the selection first whereas prototyping tools consider annotations as objects that should be created first.

4.4.3. Navigation between the annotation and the target

While the annotations and their targets need to be viewed together for a better understanding of the content of the annotation, these elements can be separated depending on the view or the integration of the annotation within the view of the artefact (e.g. by consulting a list grouping every annotations, if the annotations stored in the margins and are linked to their targets using markers).

To tackle this issue, the tools we reviewed allowed to navigate between the annotations and their targets. Thus, 24 tools (16 annotating tools and 8 prototyping tools) provided a feature to navigate from the annotation to the target. The navigation in the opposite way (i.e. from the target to the annotation) is supported by 30 tools (13 annotating tools and 17 prototyping tools).

4.4.4. Dependence of the annotation on its artefact without acknowledging its evolution

Among the annotating tools, we counted that 19 out of 25 tools allowed to create annotations that are independent from their targets. In these tools, we can note SparTag.us [43] in which annotations are stored in a remote server, Dokieli [45] which provides a unique URL for each annotations, or Annotatorjs [38] that stores annotations in a separate JSON file from the annotated artefact. However, for 5 of the remaining tools, we could not determine the architecture of the annotations storage.

As for prototyping tools, we did not find out how annotations were stored and if they were independent from the artefact they annotate. However, it is possible to assume that annotations are deeply tied to the artefact they annotate in some prototyping tools supporting annotation features such as Balsamiq which stores each prototypes in “.bmp” files. In these tools, annotations are considered as part of the artefacts.

4.4.5. Lack of support for the evolution of the context of the annotation

When creating an annotation on an artefact, it is done on the current state of the artefact. This state can change through the evolution of the artefact. Thus, the content of the annotation can be questioned: “Is the annotation still relevant?”, “Does the annotation has been taken into account and thus can it be closed?”. Indeed, the understanding and the validity of an annotation is deeply connected with its context of creation and the state of the artefact annotated at the moment of the creation of the annotation.

The evolution of the artefact has also an impact on the targeting of the annotation. Indeed, depending on the targeting method, the annotation system might not recognize the target of the annotation after

the evolution of the artefact such as SparTag.us [43] which will not load the annotation if the target has not been recognized.

During our review, we noted that a few tools tried to tackle this issue either by extracting a snapshot of the annotated artefact (Diigo [70]), annotating a state of the target (sense.us [44]), or through an anchoring algorithm trying to find the initial target prior its modifications (Hypothes.is [37]). Amaya [89] has only acknowledge this problem and handled this issue by marking annotations whose target has not been recognized as “Orphan Annotations”. These annotations are then listed for a manual management.

However, concerning the relevance of the annotation or its status of processing, they have to be managed manually. Once the annotations are processed, they can either discarded or archived like in Pidoco [47] or Protoshare [46] which allow its user to set a status on annotations to mark them as “opened”, “resolved”.

By considering that annotations are related to a certain set of versions of an artefact, their traceability and their archiving could be more understandable since their context could be better identified.

4.4.6. A few features dedicated to the management of the lifecycle of annotations

Regarding the management of the lifecycle of annotations, we noted two main features for managing the lifecycle of annotations.

The first feature is the usage of an attribute to manually assign a status on the annotation. This status usually refers to its processing (i.e. marking the annotation as solved or closing the annotation). The second feature is the recording of the events that occurs on the annotation. For instance, LucidChart [83] keeps a history of the actions made on the annotations by noting the action and the user who interacted with the annotation (e.g. “User1 has resolved the annotation”, “User2 has reopened the annotation”).

Overall, only a few tools support the management of the lifecycle of annotations. As for the other tools, the annotations must be disposed manually when they have been processed. The inconvenient of this management is that the traceability of the annotations created and processed is not ensured. Indeed, the annotations that were created during the lifecycle of the artefacts are lost when they are deleted.

4.4.7. No support for documenting a custom semantic for annotations

In the review of annotations support, we did not find any suggestion of usage of annotations nor a specific semantic given to a type of annotation.

While the lack of semantic given to annotation is not problematic, we noted that there were no way to document a convention or a semantic for annotations when a design team is using one. Indeed, the representation of an annotation can be associated to a convention as noted in the section 3.4.4. This convention can be shared on several levels: private level, team level, public level.

4.4.8. Limits of annotations support for an environment that includes many types of artefacts

The data contained in annotations may be valuable for the design process and can be related to many different design artefacts. However, several drawbacks can be identified with the support of annotations by prototyping tools in the context of the UCD process:

- Annotations are only a feature of a main editing tool which make them not interoperable between different artefacts
- Annotations rarely support targeting several artefacts

- The tools do not allow to track the modifications made on an artefact due to an annotation: an annotation can be closed or marked as solved but it is not possible to trace the modifications to ensure that they are sufficient or to trace them retrospectively.
- Annotation management is limited

We suggest that annotations should be considered as an independent artefact on the same level as any models or document in a project workspace. Promoting annotations will allow to tackle the issues mentioned above by providing a proper tool support for features related to the annotation activity within the project workspace. Moreover, in the context of the design of an interactive system in which several type of artefacts can be annotated by various stakeholders for achieving many different goals, annotations should be centralized.

4.4.9. Limits of prototyping tools for the support of the traceability of the design

- **Needs and specification tracking not supported and not directly tied to the design of prototypes**

The prototypes are designed for several purpose in the design process ranging from exploration of ideas to the refinement of a design solution while ensuring that the design meet users' requirements and their needs. Design choices and iterations are made until the prototype produced is satisfactory. Thus, each design choice is motivated by the necessity to cover one or several needs while respecting one or several requirements.

However, the requirements and the needs of a project are not fixed, especially in the early stages of the design process. Indeed, they can evolve, be refined or appear during the design process. The evolution of the requirements of an interactive system can have an impact on the design choices made during the design process. Each artefact produced during the design process must be checked in order to verify if they are compliant with the new specification and the new requirements. Thus, prototypes are affected by the evolution of the specification and of the requirements.

This dependence between the design and the requirements could be used for documenting the design process and its evolution. Indeed, it could be possible to document their co-evolution along the design process and thus enabling the traceability of the design process of the interactive system. Moreover, by tying design choices to their associated specification document, the design team could constitute a rationale for the design of the interactive system.

Among the different tools we examined for the state of the art, we did not encounter any features related to the management of the specifications and the requirements of a project.

- **Lack of representation of the dialog**

An interactive system can be decomposed into several components. The Seeheim model, presented in the Figure 4-2 identifies three parts that constitute the user interface of an interactive system.

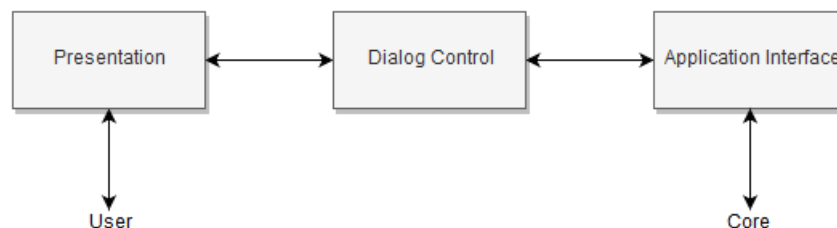


Figure 4-2 Seeheim model

The first part is the “Presentation” which regroup elements that manage both the inputs of the user and the output of the interactive system. The second part is the “Dialog Control” which handle the sequence of the different screen of the interactive system and manage the states of the interface. The third part is the “Application Interface” that is responsible for the communication between the user interface and core of the application.

During our review of prototyping tools, we noted that most of the tools were focused on giving its users a way to produce a lightweight version of an interactive system with varying degree of fidelity.

The Presentation is the most represented part of the interactive system in prototyping tools. Indeed, prototyping tools offers a large variety of widgets which can be customized by the designer. This freedom allows users to design any prototypes ranging from a low-fidelity prototype to a high-fidelity prototype. Higher fidelity prototypes tends to require more time to build but they can later be used as a reference by the rest of the design team and be used to evaluate design choices before their confirmation.

Regarding the dialog, its integration in prototyping tools is very limited. We noted that many tools included the navigation on prototypes using different methods such as the usage of hotspots, the usage of hyperlink and so on. However, the specification of this navigation is limited to the change of page to display when the user clicks on a specific area of the prototype. Thus, it is possible to note several aspects of the dialog that can't be represented accurately on prototypes due to the limitations of the prototyping tools we examined:

- Interaction techniques handled by the interactive system
- Change of state of a component or a set of components
- Conditions to fulfill for triggering a process
- Description of a process
- Data manipulated by the prototype
- Business Rules

On top of that, only a few tools presented a visual representation of the navigation possible in the prototype. The Figure 4-3 below is an example of such representation in the prototyping tool Pidoco [47].

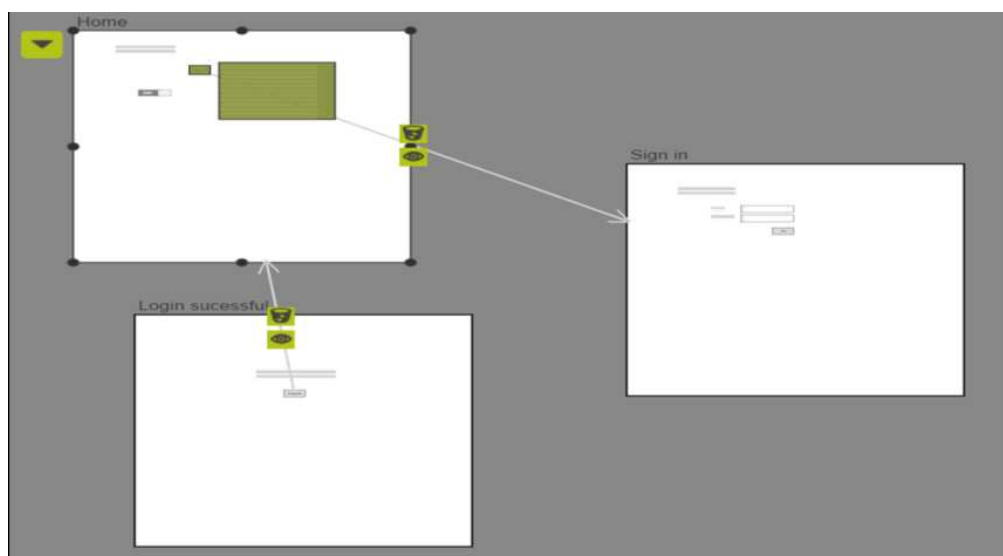


Figure 4-3 Visual representation of the navigation in Pidoco

4.5. Conclusions

In this chapter, we established a review of annotations and prototyping tools. Parts of the work presented in this chapter has been previously presented in a workshop at INTERACT 2015 [67] and published in the “Journal of Software Engineering and Applications” in 2017 [68].

Regarding the review on annotations support, we identified different ways to interact with annotations, to select a target and their different representations. The main objective of the annotating tools we reviewed is to provide a medium that can be used to communicate over a document. Most of the annotating tools we reviewed were dedicated to annotate web pages.

In the second review dedicated to the analysis prototyping tools, we presented the different trends we identified from both the academic and the commercial tools. Those trends range from the architecture of the tool with the emergence of SaaS prototyping tools to the features provided for specifying the prototypes such as the usage of widgets for specifying the UI of the interactive system or the minimal specification for the dialog. As for the annotation support in prototyping tools, annotations features were either promoted as a tool for helping collaboration within the design team or to gather feedback from external users. Overall, we observed a few tools that support annotation but their implementation was limited especially regarding the targeting of annotations on other artefacts beside the prototypes themselves as well as their management and their lifecycle support.

The overall result of these study is that while annotations are supported in many tools, their implementations are limited and not always suited in the context of the iterative process of a UCD approach.

Chapter 5. Management and usage of annotations in the UCD process

Summary

This chapter provides a presentation of the management of annotations within the UCD process and their relationship towards design artefacts specifying the interactive system. The first part of this chapter defines the lifecycle of design artefact, of annotations and showcase their interaction within the project workspace. The second part of this chapter list a few examples of uses of annotations in the context of the design of interactive systems in a UCD process and introduce a feature for connecting models using annotations inspired by semantic annotations. The third part is dedicated to the presentation of the current implementation of the annotation model for the management of the linking of design artefacts and annotations through their evolution and for the management of the different versions of artefacts.

5.1. Introduction

The state of the art on annotations and their tool support gave an overview of annotations, their characteristics and their uses. However, we did not find studies on the relationship of annotations and their targets. Thus, this chapter aims to provide a contribution to the analysis of annotations by focusing on the lifecycle of annotations and artefacts in the context of the design process of an interactive system.

Thus, the section 5.2 presents lifecycle of artefacts, of annotations and identify their relationships by emphasizing their mutual influence through their lifecycle. The section 5.3 reviews potential uses of the annotations for the design process. These uses require that the annotation model features specificities which will be presented in the Chapter 6. The section 5.4 will briefly present the management of annotations required for dealing with the evolution of both an annotation and its targets.

5.2. Artefacts and annotations lifecycle

5.2.1. Lifecycle of design artefact within the design process

The Figure 5-1 below represent the lifecycle of a design artefact within a UCD process.

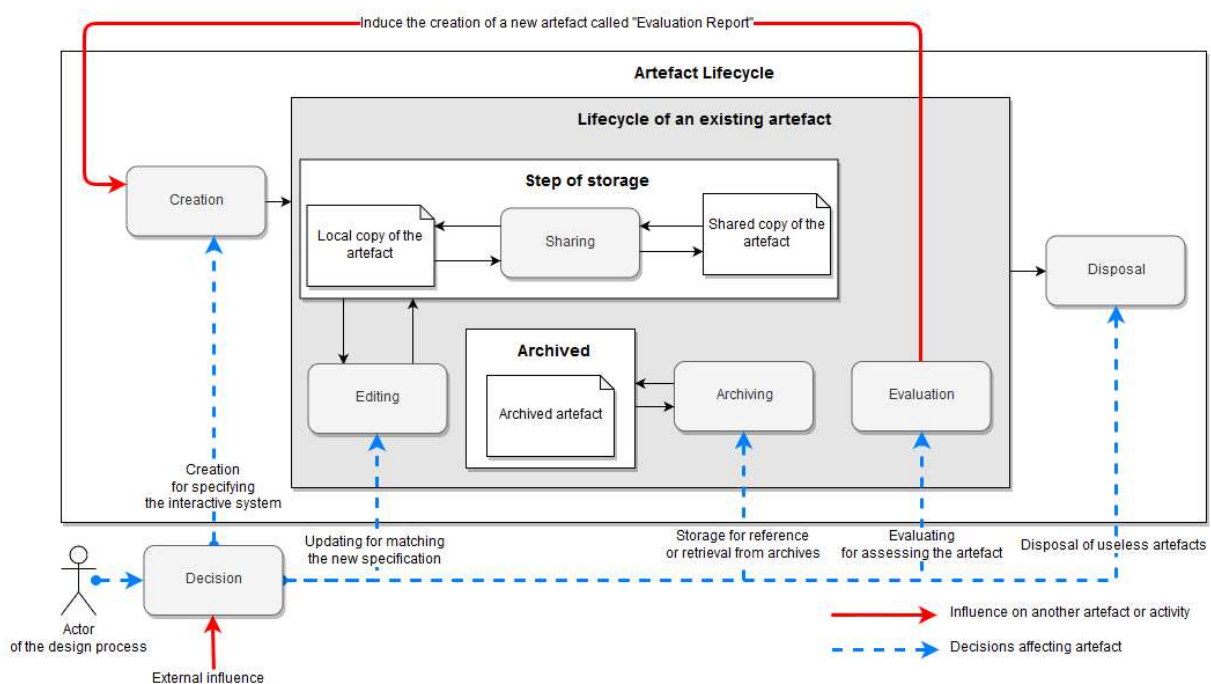


Figure 5-1 Artefact lifecycle diagram

This lifecycle starts with a decision from an actor of the design process to create an artefact within the workspace of the project. This creation can be motivated by the need of specifying the interactive system using a particular notation or representation and is based on the knowledge of the design team about the users' need and requirements. The decision is shown in the bottom left corner.

At this point, the artefact enters the "Existing" state represented by the grey rectangle of the figure. Depending on the collaborative tools used by the design team, a distinction can be made between a local copy and a shared copy of the artefact. This duplication of artefacts requires an effort of synchronization for the design team who have to manage the consistency between the local copies of each contributor with the shared copy. The details of the management of the consistency of the copies are not shown in this figure which simplify it with a "sharing" activity allowing to either share or retrieve a copy of the artefact.

After that, there is four different options for the design team that can be done at any time within the lifecycle of the artefact: the artefact can be edited, it can be stored for reference/future usage, the artefact can be submitted to end-users for its evaluation, or it can be disposed if the artefact is no longer useful.

The first option is the editing of the artefact. This editing can be done for updating an artefact, for enriching it, for correcting it, or to make it match new specifications.

The second option consists in keeping the design artefact in its current state within the workspace and make it accessible when required. This archiving can be made implicitly or explicitly by the design team. An archived artefact can be retrieved and used as a reference on future iterations or it can be reused and updated for its reintegration in the current iteration of the design process.

The third option is the evaluation of the artefact which will lead to the creation of a new or a set of new design artefacts: the "Evaluation Report" whose creation is represented with a red arrow. The results of the evaluation have their own lifecycle within the design process. They can be analyzed by the design team and depending on those analysis, several outcomes can occur such as the update of the evaluated artefact to reflect on the analysis or the creation of another artefact. These outcomes are represented by the red arrow labelled "External influence" on the decision which also includes other events or elements that can influence decisions more or less directly (e.g. new requirements, adjustment of existing requirements, references for the editing).

The last option is the disposal of the artefact when it is no longer useful for the project. For example, artefacts such as prototypes produced during brainstorming sessions on a whiteboard or a rough draft of paper prototype can be disposed.

5.2.2. Lifecycle of annotation within the design process

While annotations can depend on the artefact they are attached to, they possess their own lifecycle which can be evolving independently from the design artefact.

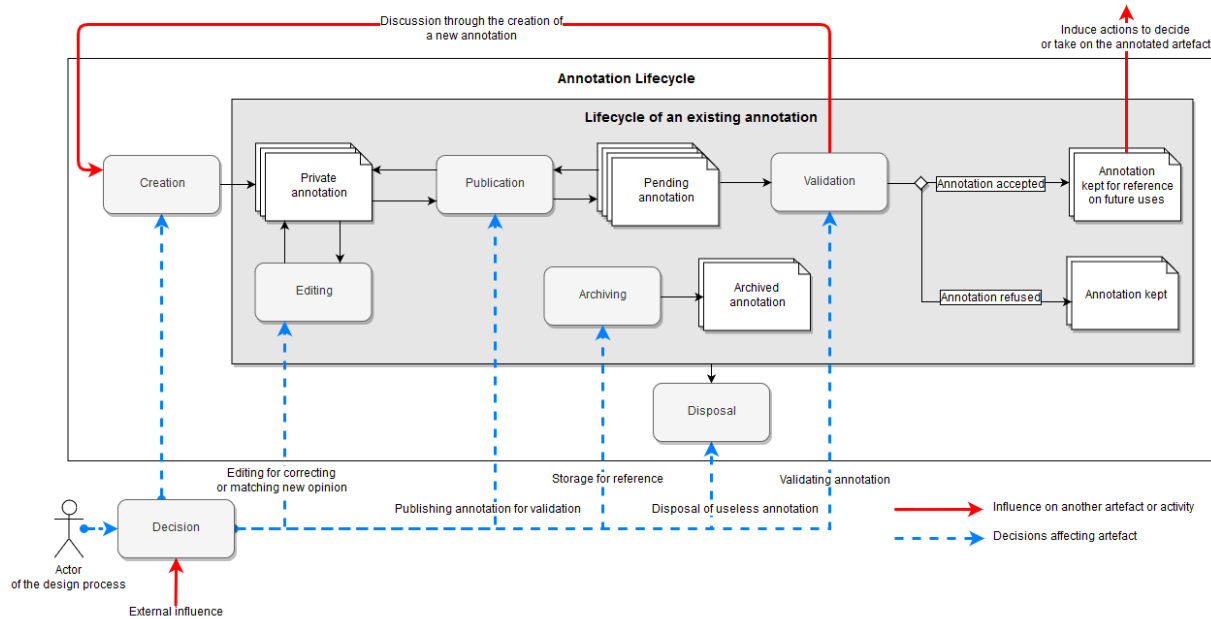


Figure 5-2 Annotation lifecycle diagram

Similarly to artefacts, the annotation lifecycle start with the decision to create it. This creation is done on the artefact and can be motivated by a variety of reasons (see 3.4) and can be influenced by external influences (e.g. in reaction of other annotations, of the content of an artefact).

This creation can occur when the artefact is being consulted, edited or evaluated. After its creation, the annotation is in a private state and only visible to its author. In this state, the annotation can be updated and reviewed anytime by its author. Depending on the annotation, its author can decide to publish it to make it visible to other members of the design team.

Published annotations are presented to the different actors of the design process who can argue with the information contained in the annotation (which can lead to the creation of an annotation as a response) or who can validate the annotation to ensure its relevance toward the artefact and to assess its content.

If the annotation is validated by the different collaborator of the design process, it can be kept as a reference. Indeed, the annotation can contain useful information for the activities of the design process (e.g. information on the requirements, appreciation marks of the design, highlight of problems). This information can have an impact on other design artefacts which is represented by the red arrow. For instance, a problem on a prototype identified with an annotation can motivate and justify a decision to edit the prototype in order to fix it. After that, annotations can be managed by indicating that it has been processed.

If the annotation is not validated, the annotation will not have an impact on other artefacts or for future uses in its current state.

Similarly to artefacts, annotations can be archived for keeping the annotation in its current state or disposed when it is no longer useful.

5.2.3. Relationship between the annotations and design artefacts: a reciprocal influence and repercussions

As presented in the two previous parts, the annotations and the design artefacts have their own lifecycle. However, those two lifecycles are interacting with each other.

An annotation is created or updated on an artefact in reaction to the content of the artefact as illustrated by the red arrow “Induce the creation of annotations” in the Figure 5-3 below. After the creation of the annotation, the annotation can be attached or detached to any artefact or fragment of artefact to include it to its target list represented with orange arrows.

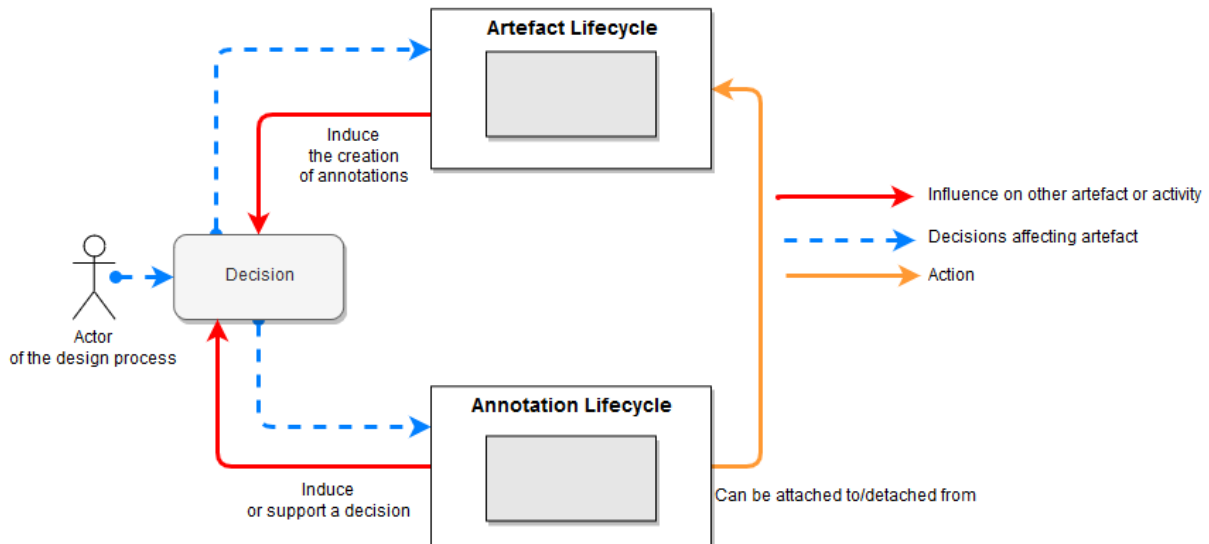


Figure 5-3 Artefact lifecycle interacting with the annotation lifecycle through decisions

In return, the annotation can have an impact on the artefacts it is attached to. Indeed, annotation can be used a medium of communication for discussing, for contributing to the elaboration of an artefact, to point out modifications to make on the artefact (see 3.4). The content annotations can be varied from the topic discussed, the intentions of the persons involved in the annotations, the precision of the information, and the quantity of information contained. Depending on this content, several type of impact can be identified: no modification required (e.g. for informative annotations), localized modification restricted to the artefact (e.g. correcting a typo, adding a precision) or global modification that can impact other artefacts (e.g. appearance of new requirements or adjustment of existing requirements). While annotations may have an impact on design artefacts, they are not always factual and can reflect opinions that should be nuanced and cross-referenced with others opinions or concrete facts prior to taking decisions.

Thus, annotations can be used to motivate or support a decision regarding the artefact as illustrated by the red arrow “Induce or support a decision”.

Regarding the impacts of an annotation to the update of an artefact, their weight can depends on several factors. Indeed, annotations can point out problems directly, reflect an opinion or unverified data from different sources and thus, the information conveyed needs to be validated. This can be done by several means such as checking the person involved in the discussion, analyzing the relevance or the trustworthiness of the information. After the validation, another aspect can be taken into account that can influence the impact on targeted artefacts. Indeed, a decision process can be integrated prior to the editing of the artefact. This decision process can assess the cost of the editing and its planning if the editing has been adopted by the design team.

Another interaction between the artefact and the annotation is the mutual update they can trigger. Indeed, when updating an artefact, the content of each annotations attached to the artefact may be questioned or the state of the annotation can be updated to match it with the new state of the artefact.

However, it should be noted that an annotation can be related to several artefacts. Thus, the update of the annotation should also consider every other artefacts it is attached to.

5.3. Usage of annotations during the UCD process

In the section 3.4, we reviewed the literature and listed various types and uses of annotations. In the context of the design of interactive system, stakeholders of the project may be brought to create annotations. In this section, we will list the different uses of annotations that can be made during the different steps of the UCD process. However, it should be noted that at any moments of the UCD process, annotations can be used to plan tasks to perform in future activities (i.e. annotations for planning [52]) or to keep a trace of any information that can be useful in the future.

- **Uses of annotations during the study of the context and of the users**

During this step of the UCD, the goal of the designers is to identify the users, understand their needs and their environment. Thus, the designers are carrying out activities such as interviewing users or analyzing existing documents of the project to gather data and to take notes.

Depending on these activities, several types of annotations can be used to achieve different purposes.

When reviewing existing documents, designers might need annotations as an active reading tool for showcasing important parts of the documents and for marking notes on the analysis of document being read. This use of annotation is favorable for collaborating and facilitate the reuse of the document since the essential fragments are standing out for others. Thus, attentional annotations and contributive annotations [52] can be used for studying, for elucidating and for cooperating [4].

In the other activities of this step in the UCD, designers can use annotations by writing down notes on collected data and information. These data can then be attached to other artefacts in which they might be relevant such as draft of specification (ex. Scenarios and User Stories).

- **Usage of annotations during the specification of the user requirements**

During the specification of users' requirements, the design team will work together to analyze and synthesize the information collected in the previous step into specification. For this step, two different uses of annotations can be identified.

The first use consists in the retrieval of annotations created during the study of the context and of the users as the other documents and information collected or produced. This information will be used to produce the specification of the interactive system.

The second use of annotations in this step correspond to the cooperation in the creation of the specification by using contributive annotations [52]. Indeed, during the elaboration of the design artefacts, the design team can communicate through annotations, plan task to perform, share the tasks to perform, write down memos and so on.

- **Usage of annotations during the production of a design solution**

For the production of a design solution, the usage of annotations is similar to the usages during the specification. However, since prototypes are partials representations of the interactive system, annotations can also be used to clarify and explain the design as pointed out by Naghsh in [57] or to attach more information on the prototypes through associative and contributive annotations.

- **Usage of annotations during the evaluation of the design**

After the production of a prototype, it can be evaluated by future users of the final application. During this activity, the prototype can be annotated in various ways.

Firstly, annotations can be created to help the design team to plan and highlight important information to remember for the evaluation.

Secondly, users from the evaluation session can leave annotations to communicate with the design team by giving their opinion, their suggestions, by pointing out a specific aspect of the prototype they want to discuss (e.g. for understanding, for correcting, for confirming [57]).

Thirdly, the design team can assume the role of annotating the prototype depending on the results of the evaluation or the observation made during the evaluation.

Overall, annotations are a versatile tool that can be used all along the iterations of UCD process. For instance, annotations can be used for documenting the design by associating documents or by explaining it, they can be used for collaborating by communicating and plan tasks, and they can also be used for reviewing the design by allowing stakeholders to highlight problems, to question the design or to give an opinion.

5.3.1. Annotations for connecting models of the interactive system

5.3.1.1 *Usage of the targeting properties of annotations to connect models*

As explained by Uren et al. in [101], Semantic Web annotations are distinct from most common and informal annotations in tools like text editor and “semantic annotations formally identify concepts and relations between concepts in documents”. To illustrate this, they give the example of the term “Paris” located in a text that is connected to an ontology. This ontology associates it to the concept of “City” and link this instance of city to the instance “France” from the concept of “Country”.

Uren et al. note that semantic annotations facilitate the information retrieval and improve the interoperability of the document.

The use of Semantic Web annotations requires to have an ontology upon which the connections will be made. Regarding its application on the artefacts of a project following the UCD process, a main ontology could be defined to get a general structure of the concepts modeled in the artefacts. This general structure can also be built and refined over the iterations of the design process. For instance, this ontology could contain the definition of users, roles, tasks, features and so on. However, each project are different and the instances of each classes should not be mixed from one project to another. While the definition of an ontology for this approach has not been planned, the first version of this approach will aim to connect similar concepts directly. The structuring of the artefacts of the projects and the concepts they represent can be examined in further studies.

One of a problem that will arise when connecting fragments of artefacts together is that design artefacts feature various level of formalization and they rarely share a common notation. This problem will be tackled in the following section dedicated to the formalization of design artefacts to ensure their interoperability.

5.3.1.2 *Necessity to formalize artefacts within the project workspace*

As stated by Uren et al. in [101], the support of heterogeneous documents is a technical challenge, but he also stated that its support is necessary for integrating the semantic annotations into existing work practices. Another requirement identified by Uren et al. is the support of the document evolution

which consists in ensuring the consistency between the annotation and the fragment of artefact it is attached to.

We decided to integrate the annotations in a selected set of type of documents as a first step. These type of documents are listed in the next section. Each of these documents are based on a structured and a formalized notation. Thus, each artefact can be fragmented into unitary elements or group of elements. These fragments can then be used as targets of annotations and thus be connected to other fragments from other artefacts of the project workspace.

This formalization of the artefact is also used for ensuring that the annotations are always attached to the same fragment. Indeed, the interoperability of the models (i.e. artefacts models with the annotation model) is facilitated with the formalization of the notations. In the implementations of the models in the editing tools, the linking mechanism is ensured by an ID system.

However, the different evolution of the fragment (e.g. edition, deletion) can have repercussions on the annotations or on related artefacts. Those repercussions should be taken into account and reflected if necessary on other artefacts as presented in the section 5.2.

5.3.2. Annotations for assisting stakeholders in their activities

Existing annotations can be used as a reference by the stakeholders. Indeed, they contains various information and can be used as an active reading tool for facilitating future analysis of the design artefact it is attached to.

However, the accumulation of annotations over time can become a burden and diminish the readability of a document. Thus, annotations should be managed and indexed in order to facilitate their sorting and usage. In the section 3.4 is listed several uses of annotations. Depending on the task of a user, some annotations might be relevant to the user whereas others should not be displayed.

The type of an annotation is not sufficient for filtering annotations that are relevant for the goal or the task performed by the member of the design team. Even if we assume that every annotation created were relevant to its target at that moment (i.e. thus ignoring “noisy and cumbersome” annotations, the relevance of an annotation toward and artefact is not guaranteed. Indeed, this relationship between an annotation and its target is not static and can evolve over time depending on the evolution of the context, or the evolution of the targeted artefact itself.

This relevance of an annotation can be evaluated manually by the design team after analyzing various attributes (e.g. the content of the annotation, the context, the targets, the author, the date of creation). The evaluation is done after its publication as shown in the diagram (see Figure 5-2). Once an annotation has been evaluated, their relationship with the targeted artefact can be updated as presented in the next section.

5.4. Management of the relationship of an annotation and its targets

5.4.1. Management of the targets based on the relevance of the annotation

The management of the relationship of an annotation to its targets is controlled by its relevance toward the target. If an annotation is relevant to an artefact or to a fragment of an artefact, this element should be integrated in the list of targets of the annotation.

- **Managing existing relationship**

Annotations and design artefacts evolve through the activities and the iterations of the design process. These evolutions can impact the relationship between an annotation and its target.

On the one hand, annotations suggesting modifications on a design artefact or related to an edited fragment tends to be transient. Indeed, after the suggestion of modification has been processed by the design team either by modifying the artefact or by deciding to reject the modification, the annotation can be dismissed by archiving it and marking it as having been processed.

On the other hand, some annotations are persistent over time such as annotations made during the analysis of an artefact and notes carrying information and data. For these annotations, the design team must ensure the consistency between the artefact and the annotation. Indeed, an annotation can be created by referencing a specific state of the design artefact. After the edition of the artefact, this referencing might be questioned. Thus, the annotation should be updated accordingly if necessary to ensure that the information contained in the annotation are still relevant for this state of the artefact.

- **Establishing relationship with new artefacts**

During the design process, new design artefacts can be created during each steps and each activities. These artefacts can be concerned by existing annotations. Thus, these artefacts should be connected to these existing annotations.

- **Disposal of artefacts targeted by annotations**

The presentation of the lifecycle of artefacts showed that artefacts can be disposed if they are no longer useful for the design process. However, there might be annotations that are connected to those artefacts. The disposal of an artefact does not imply the disposal of all of its annotation. Indeed, an annotation can contain precious information that can be relevant for other artefacts or for future uses during the design process. Indeed, an annotation can feature data or results of evaluations that can be reused when designing new artefacts. On top of that, annotations can be targeted to other design artefact that are not disposed. Thus, following the disposal of a design artefact, a decision has to be made on annotations targeting the artefact to decide if the annotation should be kept persistent or disposed as well.

It is also possible to consider the reuse of annotations on other development project of interactive systems featuring similar ideas or problematics

5.4.2. Targeting support of versioned files and versioning of annotations

The implementation of the model of annotation is designed to support the versioning of both files and annotations. This section will be dedicated to the presentation of the support of the versioning in the current implementation of the model which is presented in the section 6.3.

- **File versioning management**

In the current implementation of the annotation model, any artefact of the project workspace can be targeted. This flexibility is exploited for managing the targeting of the different versions of an artefact. Indeed, each version of an artefact can be registered as a different artefact in the annotation file. Then, each annotation can target a selected set of versions of the main artefact.

A future evolution of the implementation of the model is to define a “versions” node within the “file” node which will regroup each versions of the artefact.

- **Annotation versioning**

As briefly mentioned in the presentation of the XSD file in the section 6.3, annotations are grouped into sets of annotation. Each set correspond to a version of the annotations made by a user. A new

version of the annotations and thus a new annotation set is created when the previous set conclude its lifecycle.

The lifecycle of a set of annotation has been defined to support a validation process we designed and is composed of five states: “Annotating”, “Request review”, “Reviewing”, “Reviewed”, and “Rebuttal”.

This validation process is based on the exchange of annotation files and review files, the communication within the design team for the synchronization and is presented in the Figure 5-4 below:

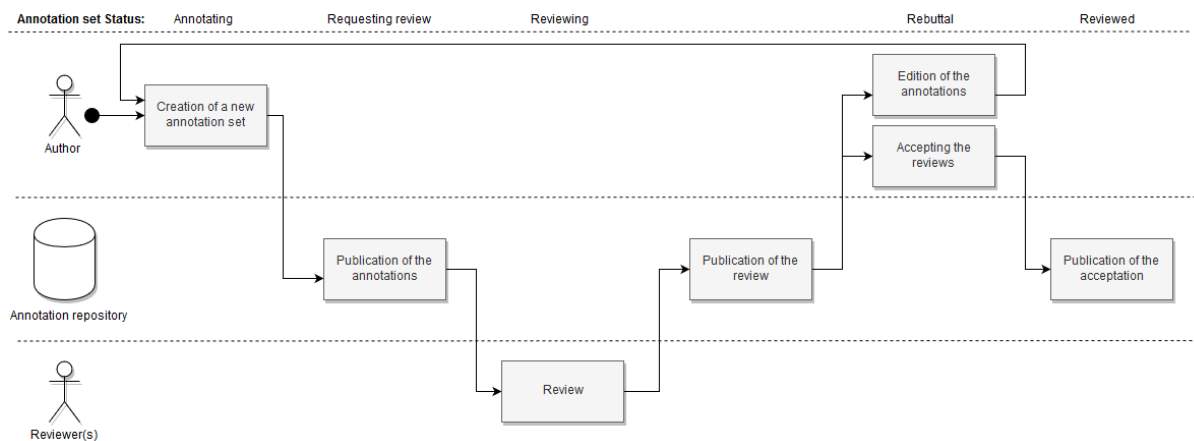


Figure 5-4 Annotation validation process

The validation process starts with the creation of a new annotation set with the “Annotating” status. In this state, the author can create and edit any annotations he wants. However, those annotations are private and only the author can see those annotations.

Once the author is satisfied with his annotations, he can publish his annotation set which will be available to other members of the design team in the state “Requesting review”. In this state, the annotations of the set cannot be edited to ensure that each reviewer will be getting the same version of the annotations.

When a reviewer retrieves the annotation set to review, he can review it. The annotation set is then in the state “Reviewing”, which indicate that at least one person is being reviewing the annotation set. The reviews will be stored in a review file that uses the same targeting mechanisms as the annotations (due to time constraints, the review file model has not been implemented nor the reviewing support system).

After the reviewer has finished his reviews, he can publish it and make it available to the initial author of the annotation set as well as the other members of the design team.

The author of the annotation can then retrieve each review made on his annotation set and consult them. Doing so will set the annotation set to the “Rebuttal” state. In this state, the author can choose to edit his annotations which will be done by creating a new set of annotations and start a new iteration of the validation process or he can accept the reviews and end the process for this annotation set which will have the “Reviewed” status.

When the last annotation set has the reviewed status, the author can initiate a new validation process by creating a new annotation set.

By default, a new annotation set will duplicate existing annotations from the previous annotation set (which can either be in a “Rebuttal” or “Reviewed” state). The duplicated annotations will have their own id but they will also have a “RefId” attribute set to the value of the id of the same annotation from the previous annotation set. This system allows the annotation tool to know that this annotation is a new version of an existing annotation. Thus, this part of implementation of the model can be considered as a versioning system that allow to keep a trace of each versions of the annotations produced.

5.5. Conclusion

In this chapter, we first presented the interactions between artefacts and their annotations during their lifecycles. Part of this work has been accepted and published in the EIMS workshop at EICS 2019.

As shown in the section 5.2.3, these interactions influence the decision activity which determines actions to undertake on the artefacts and in the design process. Thus, the annotations can be playing a significant role in the evolution of design artefact and allow to induce and support decisions related to these artefacts. Annotations can also be used to materialize decisions and connect them to artefacts concerned by these decisions in order to ensure their traceability. Indeed, they contain information related to the interactive system, to the design artefacts representing the interactive system, or to the management of the design process.

After that, we listed a few other example of uses of annotations during the design process. However, we noted that annotations need to be managed to ensure their consistency towards their targets and their relevance for the interactive system through the evolutions of artefacts and of the iterations of the design process. Another management of annotations consists in their update after taking them into account in the evolution of design artefacts.

On top of these managements of annotations, an increasing number of annotations can reduce the readability of the artefact when annotations are placed “in situ” (within the representation of the artefact) and some annotations might not be relevant depending on the intended use of the artefact (ex. Annotations explaining the design might not be relevant for designers who know it when fixing problems on a prototype). Thus, a particular attention should be paid to the indexation and filtering of annotations created during the design process.

Overall, annotations can be a useful tool which can be used for various purpose in the context of a UCD process but they require to be managed in order to avoid any confusion when either the artefact or the annotation evolves and to avoid the increasing number of annotations through the iterations which can impede the activities related to artefacts.

Chapter 6. Model-based approach for describing annotations

Summary

This chapter presents an annotation model connected to artefacts used for the design of interactive systems. This model extends the W3C Web Annotation Data Model and features specific classes for its use within a project following a UCD approach. The first section of this chapter emphasizes the importance of the presence and the independence of annotations within a project of a development process. The second section will list the different concepts retrieved from the W3C Web Annotation Data Model, review the specificities of annotating artefacts used within an iterative design process, and present the annotation model adapted for annotating those artefacts. The third section will be dedicated to the application of this model using XML files to describe the annotations. The fourth will be assessing the extensibility of the annotation model.

6.1. Introduction

During the development process of an interactive system, the design team produces a variety of artefacts that are revised through multiple iterations. Each artefact is likely to receive annotations either as a support for active reading, for communicating, for reviewing, for planning or for editing. Indeed, the annotation activity is a common practice among both contributors and readers of artefacts which allow them to express themselves or to contribute to the creation of the document for instance as mentioned in the section 3.4.2.

In many implementations of annotations in the tools we reviewed (see section 4.3), the annotations were a part of the artefact and was strongly tied to its artefact, thus making annotations as a complement of existing artefacts. Many annotating tools we reviewed were based on finished documents while the artefacts the design team are manipulating evolve through the iterations. Indeed, annotations tends to be disseminated among the different artefacts and their different versions. On top of that, each type of artefacts has their own annotation support which make their management more complicated. This scattering of annotations favors the replication of annotations or the loss of information contained in the annotations when artefacts are evolving or new artefacts relevant to existing annotations are produced. A solution for this issue is to make annotations independents from their targets which can largely differ in term of nature and notation: annotation should not be tied to one version of an artefact as usually implemented in prototyping tools.

This chapter addresses these problems by proposing a model that specify the structure of annotations and somewhat formalize the relationship between annotations and the diverse artefacts used for the development of interactive systems. One of the specificities of the model presented hereafter is the including of attributes that help to follow the life cycle of annotations as presented in the Chapter 5.

Models and formalism allow to describe interactive systems unambiguously. However, stakeholders of the project may need to add additional information to those models that is not necessarily a part of the model itself such as an opinion, an example, a reference or a data sample. This information can be relevant or useful in the context of the model or during the usage of the model. To add this additional information, annotations can be used thanks to their flexibility and their “additional” aspect on artefacts.

In this chapter, we propose an annotation model for dealing with the development process of interactive system. This model is inspired from the Web Annotation Data Model [33] and it provides extensions to cope with the idiosyncrasies of the use of annotations when designing interactive systems.

6.2. Towards an Annotation Models for Artefacts used to build interactive systems

This section present the key features we reused from the W3C Web Annotation Data Model and the features that were specifically added to annotate artefacts along the development process of interactive systems.

6.2.1. Features inherited from the Web Annotation Data Model

The first feature is the ability to connect one annotation to one or several artefacts as well as the ability to specify the state or version of the artefact in which it is relevant to the annotation. Indeed, annotations can contain information that are not limited to only one artefact. For instance, they can feature information on a requirement that is represented or discussed on many artefacts (i.e. prototypes, specification, task model) or feature information that is relevant to every version of an artefact (i.e. feedback on a prototype that should be taken into account on every other prototype produced afterward). This feature is supported by the definition of the “Target” and “Selector” classes that allow to associate an annotation to any number of documents and to specify which parts of the document are relevant for the annotation.

The second feature is the independence of annotations regarding their target as well as the unique format of annotations regardless of the type of the target. This feature is linked to the ability to specify several targets since an annotation should not be dependent on only one of their targets. As presented in the section 5.2, the annotations have their own lifecycle and while an annotation lifecycle is interacting with the lifecycle of an artefact, they should be handled separately: the disposal of an artefact does not always imply the disposal of its annotations which can contain precious information or can be relevant to other artefacts.

The third feature is the support of the lifecycle of an annotation. The Web Annotation Data Model track the events happening on the annotation with the “Lifecycle information” class such as the date of creation of the annotation, of its generation, and of its last modification. By storing information on the evolution of the annotation through its lifecycle will allow us to provide a traceability of the annotation.

The fourth feature is the support of different types of representation. This support is described with the “Body” classes that define both the content and the type of content of an annotation. For instance, it is possible to define annotations through text, images, sound or video.

6.2.2. Specificities of the annotations on artefacts of the design of interactive systems

The Web Annotation Data model specifies a model for unified annotations over web resources. In the context of the design of interactive systems, many different design artefacts are produced within a shared project workspace and used by the design team as well as end-users.

Several group of artefacts can be defined depending on their nature and usage during the UCD process. The first group of artefacts called “Context representation” regroup the artefacts describing the environment and the users targeted by the project (e.g. personas, evaluation results). The second group is the “Documentation and guidelines “ which consists in the generic documentation artefacts that is not restricted to one project and can be used as a reference by several project. The third group is the “Specification and requirements” which gather any documents and models of the interactive system describing how it should be designed and what are the requirements for this project (e.g. prototypes, task models). The fourth group of artefacts is the “Implementation” and it regroup the artefacts tied to the implementation. The last group is the “Communication and organizational” and consists in the artefacts related to the communication and the management of the project (e.g. meeting reports, planning document).

Overall, each artefact within a group identified previously might share a set of the same annotations due to the similarity of what they describe, of their goals and of their usage. However, each of these types of artefact are using a different format. Moreover, one type of artefact might feature several formats (e.g. text documents can be produced using Microsoft Word, or OpenOffice).

The extensions proposed for annotating artefacts for interactive systems are listed as follows:

- **Integration in tools**

The first extension of the annotation model is its integration within tools editing artefacts of different types. Similarly to the Web Annotation Data Model, the model we propose is featuring an interface for implementing specific classes for the selector of fragments of artefact.

Artefacts we would like to annotate includes prototypes, tasks models, system models, specification documents and any other document produced or gathered during the design process.

- **Representation of annotations**

The second extension of the annotation model is the support of custom annotations either for allowing the design team to define a convention in the meaning of the annotation through its representation (Bringay noted in [29] that a convention was defined within a group for the color of a post-it giving the annotation a semantic meaning depending on the color used) or to support annotations structure that are specific to a type of document.

For that, we featured in our model the distinction between “Basic body” and “Structured body”. The basic body is defining the format of the annotation (e.g. text, image). The structured body is defined as a composition of basic body and can includes constraints on the content of each basic body. This structure has been used to define a “Scenario” annotation specific to prototypes that will be presented in the use case in the section 8.3.2. This structure can also be used to define a format of annotations that will give it a semantic similarly to the observation by Bringay in [29].

While we intend to support freestyle annotations (e.g. freehand drawing), they will not be the focus of our studies and represent a special case for annotations in term of content, meaning and targeting.

- **Indexing and management of annotations**

The third extension is dedicated to the management and indexing of annotations. This indexing and management of the annotations is necessary when dealing with an increasing number of annotations in a project workspace.

This extension is first implemented with the integration of the different uses of annotations found in the literature (see 3.4) as categories for sorting annotations. These categories can be represented as a “Tag” on the annotation. Thus, our model distinguishes attentional annotations for future reading of the artefact, organizational annotations for planning tasks and managing actions to take, contributive annotations for enriching the artefact and associative annotations for linking other artefacts. The integration of semantic annotations (represented by “associative” annotations type in the model Figure 6-1) will help for organizing information, building a structure for the information gathered and for describing the interactive system being developed in its entirety instead of only considering localized annotations in artefacts as existing prototyping tools do (more details is given in the Chapter 9). Similarly to the customization of representation of annotations, these categories of annotations could be extended by a design team to make it match to their conventions and naming. Thus, the indexing of these annotations could be assimilated to a folksonomy.

The second implementation related to the management of annotations is the versioning of both the annotations and the artefacts, similarly to the state classes as defined in the W3C Web Annotation Data model. The details of the versioning is be given in the section 5.4.2 and 7.3.

The third implementation for indexing and the management of annotations is the support of the management of the lifecycle of annotations. In the context of the design of an interactive system, among the different uses of annotations, they can be used as a support for suggesting modifications or plan action to take through decisions. Once the annotation has been taken into account, it can be marked as processed and archived. More details on this mechanism linked with the lifecycle of annotations are available in the section 5.2.2 and 7.3.

- **Decision support**

The fourth extension is aimed at using annotations as a tool support for supporting decisions that will allow the traceability of choices made during the design process. Indeed, annotations can contain information that can influence the design process and the interactive system being developed. For instance, annotations can contain feedbacks from an annotator, discussion among the design team, they can attach notes taken during the evaluation or during the study of the context of use of the interactive system to design artefacts. Thus, those annotations can be linked to the decisions either to suggest a decision to make or to provide concrete arguments over a decision to take.

Nonetheless, prior to take a decision, the annotation should be evaluated in order to estimate the relevance of the annotation, its weight, its trustworthiness, its reliability or its popularity of the information it contain (represented as a metadata in the model). For instance, a new requirement requested through annotation by the client or the customer will not have the same weight as if it were requested by a developer. Another example is that more weight can be given to an opinion if it is shared by many (e.g. evaluations with end-users giving the same results). Thus, the weight of an annotation can be checked for instance by consulting the metadata of the annotation (e.g. by checking the author, his role), by manually evaluating the content of the annotation, by providing a voting system on annotations.

On top of these use of annotation for decisions, we also suggest that decisions made through the design process can be traced using annotations. Indeed, annotations could represent the decisions made along the iterations, the options considered, the impact of the decisions, the arguments and resources used for the decision making. More details on the use of annotations as a support for decisions and traceability of the decisions are available in the sections 5.2.3 and 7.3.

6.2.3. Description of the annotation model

The proposed model for annotating artefacts used to build interactive systems is illustrated in the Figure 6-1. Four main entities describe the core elements of our annotation model, as follows: **artefact**, **annotation**, **target** and **creator**.

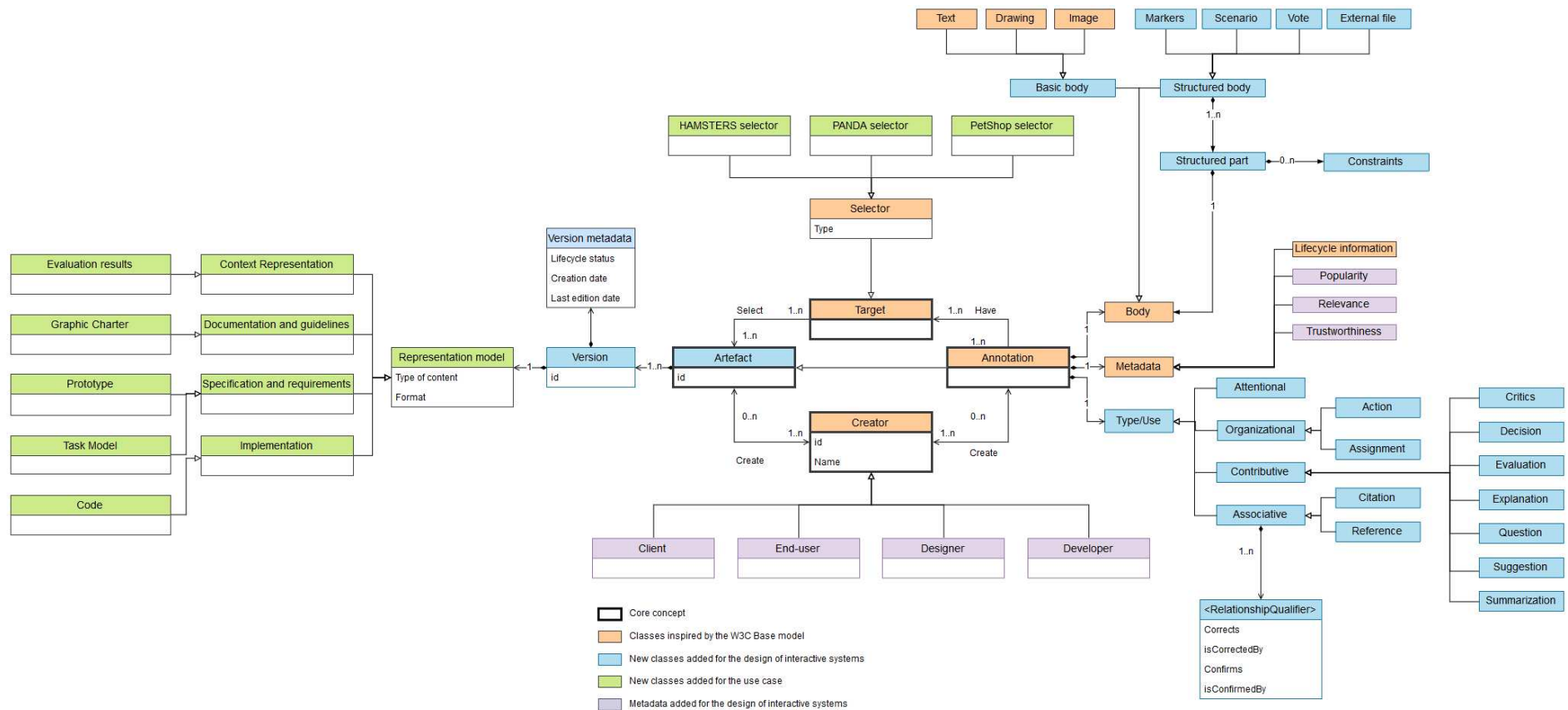


Figure 6-1 Representation of the annotation model inspired from the W3C Web Annotation Data Model

An entity **artefact** represents any file or document that is produced (or gathered) during the design process. It is composed by a set of versions that correspond to a versioned content of the artefact. Each version is also characterized by its metadata which contains various information like the creation date and the lifecycle status of this version that indicate if the artefact is being written, waiting for review, reviewed finished, being updated or archived. The content of the artefact refers to the type of information that is contained in the artefact. Each type of content reflects a different aspect of the interactive system that is being designed and each follow a different syntax. For instance, a task model can be described using a task model notation such as HAMSTERS.

In our model, an **annotation** is considered as a particular type of artefact. This enable the independence of annotations compared to their evolving targets, it enables annotating different artefacts with one annotation and also enable annotating other annotations.

Our model extends the elements body, type and target as defined in the Web Annotation Data Model. Regarding the body, our model includes classes defining the visual aspect of the annotation. A body can have a basic representation (such as text, drawing and images) or a structured representation composed of several basic representations that can features constraints. Examples of annotations featuring a structured body include: a vote (which contains a counter allowing every reader to cast a vote agree/disagree), a scenario (which is presented in the section 8.3.2 and features a list of text elements constrained by a grammar to express a sequence of tasks performed by users), an external file (which in addition to a visible label, has a link allowing to open the file), and markers (associating a glyph/icon next to the label).

As for the type, this was extended to describe the function of the annotation as suggested by Zacklad [52] (i.e. attentional, organizational, contributive and associative annotations).

Like the Web Annotation Data model, a **target** is materialized by the targeted artefact and a **selector** which is used to specify the relevant parts of the artefact that is being annotated. A selector must be specialized to cope with the inner structure of the document being annotated. So far, we have implemented this class to cope with HAMSTERS task models, PETSHOP dialog models and PANDA prototype models. Further details about these models are given in the case study in the Chapter 9.

We also adapted the entity **creator** defined in the W3C. Indeed, we refine the entity creator to include a variety of roles involved in the development process of interactive systems. In our model, these roles are expressed by the means of metadata whose terms must be adapted according to the project. A basic list of roles having access to the annotations would include for example, client, end-users, designer, and developers.

The representation of the annotation model adapted to the design process is presented by the Figure 6-1 which includes: classes inspired from the W3C model (painted in orange), new attributes (metadata classes) that have been added (painted in violet), new classes to the model tied to the context of the design of interactive systems (painted in blue), and specialized classes that extend the use of the overall model for particular types of artefacts used in our case study (painted in green). It is worthy of notice that metadata are used to help actors of the development process to assess the different annotations produced on the artefacts. This assessment will then be useful for the decision process. Each annotation can be created by any user who have access to the annotation system on the artefact and each annotation can express a variety of information on any domain (ex. Use cases, requirements, constraints, data, identification of problems, personal feedback, and personal opinion). The role of the creator of the annotation can help to determine both the expertise and legitimacy of the creator in the information given in his annotations. Other parameters can evaluated (ex. Popularity of an opinion

within an identified group) or be informally assessed such as the relevance and trustworthiness of information (ex. A creator can emit hypothesis or facts based on unreliable sources).

6.3. Operationalization of the annotation model

In this section, we will be introducing a first implementation of the annotation model presented in the previous section. This implementation is still being developed and is supported by a tool framework that will be presented in the next chapter. The implementation of the model consists in the definition of an XSD that is used to structure XML annotation files produced by a tool. The XSD file can be found in the Annex 4.

6.3.1. DTD and XML files

- **Core concepts of the implementation**

In this implementation, we choose to define one annotation file per author (which can be anonymous) and per project. Thus, one annotation file contains every annotation made by one author and the list of each artefacts referenced by the annotations.

Regarding the publication of annotations, this feature has been neglected at the moment since it is out of the scope of this study and thus, its support is limited to an “all or nothing” sharing which is done by exchanging the annotation file. Thus, it is not possible to select a set of annotations to share but given the context of a UCD in which designers collaborate toward the design of an interactive system, personal annotations support should not be mandatory.

- **Presentation of the main nodes of the XSD**

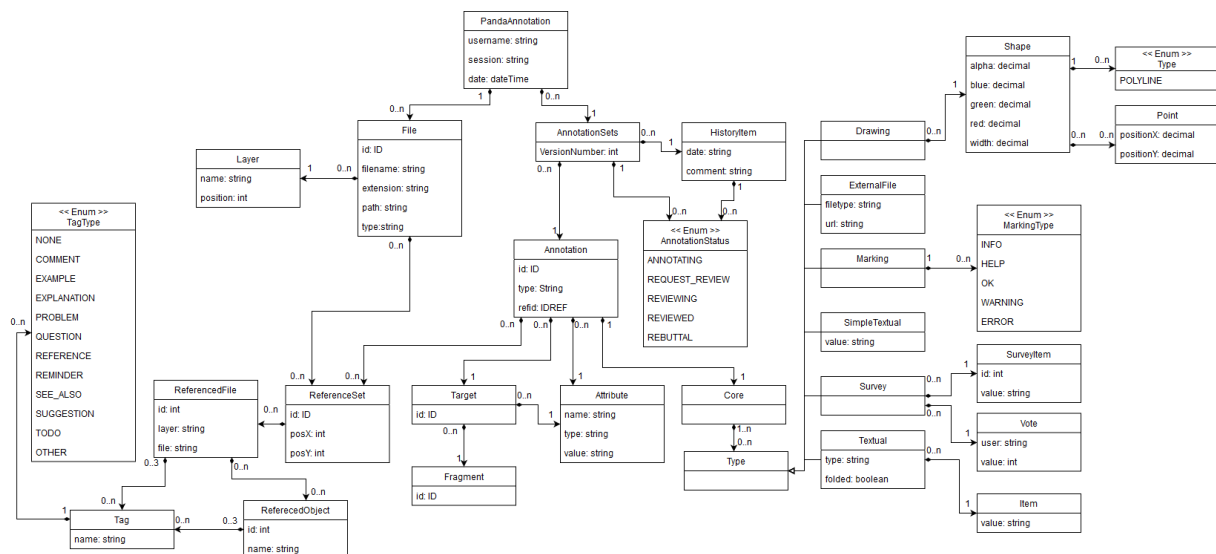


Figure 6-2 Graphical representation of the XSD

In the current implementation of the model represented in the figure above, annotations files are structured on a root node called “pandaannotation”. This root features three attributes: “username”, “session”, and “date”.

```

- <xs:schema targetNamespace="http://www.irit.fr/ICS/PANDAANNOTATION/8.0" elementFormDefault="qualified"
  <!-- *****-->
  <!-- ANNOTATIONS ROOT ELEMENT -->
  <!-- *****-->
  - <xs:annotation>
    - <xs:documentation xml:lang="en">
      This Schema defines an Annotation
    </xs:documentation>
  </xs:annotation>
  - <xs:element name="pandaannotation">
    - <xs:complexType>
      - <xs:sequence>
        + <xs:element name="files"></xs:element>
        + <xs:element name="annotationsets"></xs:element>
      </xs:sequence>
      <xs:attribute name="username" type="xs:string" use="required"/>
      <xs:attribute name="session" type="xs:string"/>
      <xs:attribute name="date" type="xs:dateTime"/>
    </xs:complexType>
  </xs:element>

```

Figure 6-3 XSD describing the main structure of a PandaAnnotation file

The “username” attribute is defining the name of the author which can be identified or anonymized. The session and date attributes are optional and are only used to anonymize participants of experiments on the annotation tools.

The two child nodes of the root are respectively named “files” and “annotationsets”.

```

- <xs:element name="files">
  - <xs:complexType>
    - <xs:sequence>
      - <xs:element name="file" minOccurs="0" maxOccurs="unbounded">
        - <xs:complexType>
          - <xs:sequence>
            + <xs:element name="layers" minOccurs="1"></xs:element>
            <xs:element name="referenceSet" type="referenceSet" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID"/>
          <xs:attribute name="filename" type="xs:string"/>
          <xs:attribute name="extension" type="xs:string"/>
          <xs:attribute name="path" type="xs:string"/>
          <xs:attribute name="type" type="xs:string"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Figure 6-4 XSD of the “files” node

The “files” node is used to list every artefact targeted or referenced by the annotations. A file is defined by the following attributes: “id”, “filename”, “extension”, “path”, and “type”. Each of these attributes are defined as a “string” in order to allow the support of any type of artefact whether there are locally or remotely stored. For instance, this model can be used to reference a Word document or an HTML resource.

```

- <xs:element name="annotationsets">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="annotationset" minOccurs="0" maxOccurs="unbounded">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="historyitems">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="historyitem" type="historyitemElement" minOccurs="0" maxOccurs="unbounded"/>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- <xs:element name="annotations">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="annotation" type="annotationElement" minOccurs="0" maxOccurs="unbounded"/>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:sequence>
- <xs:attribute name="versionnumber" type="xs:int" use="required"/>
- <xs:attribute name="status" type="annotationstatus" use="required"/>
- </xs:complexType>
- </xs:element>
- </xs:sequence>
- </xs:complexType>
- </xs:element>

```

Figure 6-5 XSD of the "annotationsets" node

The "annotationsets" node gather every annotations and sort them into sets. A set of annotation is defined by a version number, a status and its annotations. These sets can be considered as a versioning system of the annotations in which an annotation will be duplicated in each set. This versioning system is also used in our implementation as a way to control the editability of annotations for its validation by other designers and thus ensuring that once an annotation set has been shared for a validation, it can't be edited.

```

<!-- *****-->
<!-- ANNOTATION ELEMENT -->
<!-- *****-->
<xs:complexType name="annotationElement">
  <xs:sequence>
    <xs:element name="targets" type="targetsElement"/>
    <xs:element name="core">
      <xs:complexType>
        <xs:sequence>
          <xs:choice>
            <xs:element name="drawing" type="drawingElement"/>
            <xs:element name="externalfile" type="externalfileElement"/>
            <xs:element name="marking" type="markingElement"/>
            <xs:element name="simpletextual" type="xs:string"/>
            <xs:element name="survey" type="surveyElement"/>
            <xs:element name="textual" type="textualElement"/>
            <xs:any namespace="##other" processContents="lax"/>
          </xs:choice>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="attributes" type="attributesElement"/>
    <xs:element name="referenceSetIds" type="xs:int" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID"/>
  <xs:attribute name="type" type="xs:string"/>
  <xs:attribute name="refid" type="xs:IDREF"/>
</xs:complexType>

```

Figure 6-6 XSD of the "annotationElement" type

Each "annotationsets" contains a set of child nodes called "annotation". The "annotation" node is composed of a "targets" node containing the different targets of the annotation, a "core" node defining its representation and its content depending on the type of representation, a set of attributes stored in the "attributes" node which specify parameters such as the default position of the annotation, and a "referenceSetIds" which point towards files acting as references of the annotation.

An annotation has three direct attributes: an "id", a "type" and a "refid". The "type" is used as a readable name for the tool parsing the annotation file. The "refid" is used to reference another version of the same annotation contained in a previous annotation set in order to connect them to each other.

```

<!-- *****-->
<!-- TARGETS ELEMENT -->
<!-- *****-->
<xs:complexType name="targetsElement">
  <xs:sequence>
    <xs:element name="target" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="fragments">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="fragment" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:attribute name="id" type="xs:ID"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="attributes" type="attributesElement"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

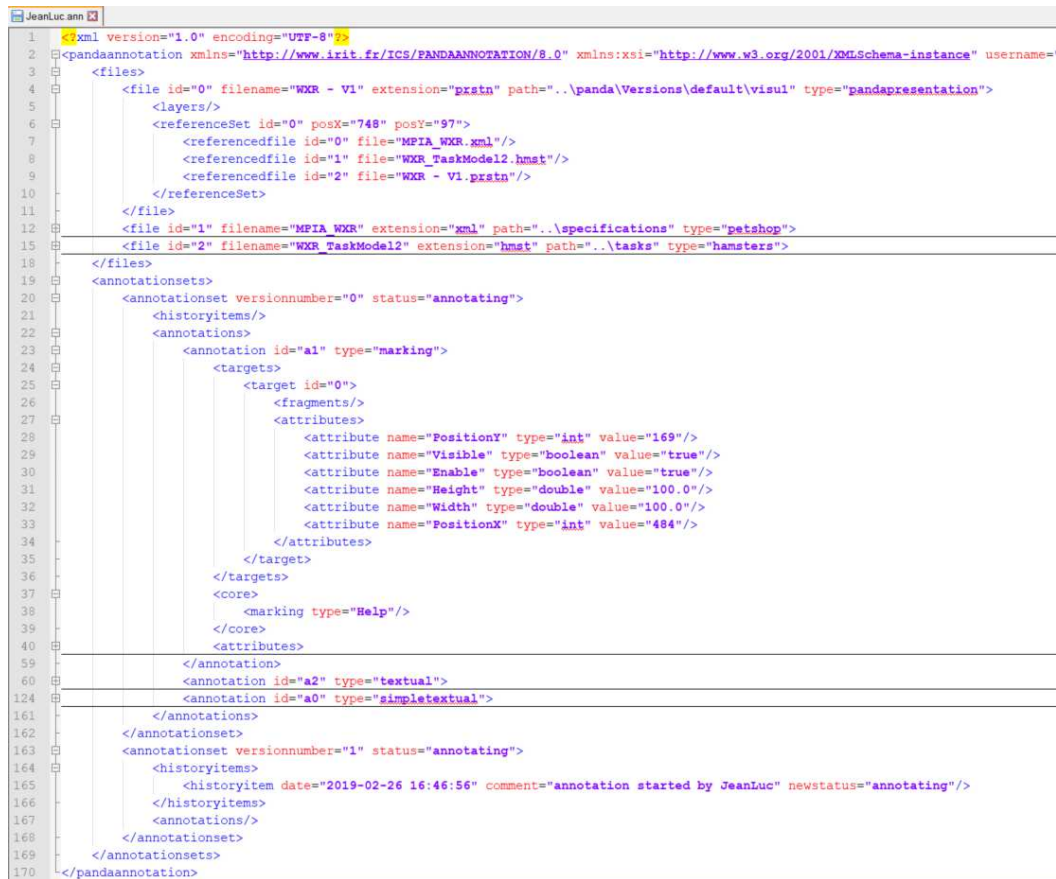
Figure 6-7 XSD of the "targetsElement" type

The child node "targets" of an annotation is defined by a set of targets referencing files listed in the "files" node (i.e. one child of the root). A target can have fragments that identify parts of the target. Details on the targeting system can be found in the section 5.4.2 of the Chapter 5.

A target has also a set of attributes. These attributes are defining properties of the annotations in the context of the target. Thus, each attribute defined in this node will override the properties defined in the "annotation" node when the annotation tool will display the annotation on the targeted artefact. An example of this overriding is given in the case study of the section 9.3.3

- **Example of annotation file**

Annotations files are stored in an XML format and are structured using the XSD file presented in the previous section. The figure below shows an example of a collapsed annotation file that target 3 files and contains 3 annotations within one annotation set.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <pandaannotation xmlns="http://www.irit.fr/ICS/PANDAANNOTATION/8.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" username="
3
4   <files>
5     <file id="0" filename="WXR - V1" extension="prstn" path="..\panda\Versions\default\visul" type="pandapresentation">
6       <layers>
7         <referenceSet id="0" posX="748" posY="97">
8           <referencedfile id="0" file="MPIA_WXR.xml"/>
9           <referencedfile id="1" file="WXR_TaskModel2.hmat"/>
10          <referencedfile id="2" file="WXR - V1.prstn"/>
11        </referenceSet>
12      </file>
13      <file id="1" filename="MPIA_WXR" extension="xml" path="..\specifications" type="petshop">
14      <file id="2" filename="WXR_TaskModel2" extension="hmat" path="..\tasks" type="hamsters">
15    </files>
16    <annotationsets>
17      <annotationset versionnumber="0" status="annotating">
18        <historyitems/>
19        <annotations>
20          <annotation id="a1" type="marking">
21            <targets>
22              <target id="0">
23                <fragments/>
24                <attributes>
25                  <attribute name="PositionY" type="int" value="169"/>
26                  <attribute name="Visible" type="boolean" value="true"/>
27                  <attribute name="Enable" type="boolean" value="true"/>
28                  <attribute name="Height" type="double" value="100.0"/>
29                  <attribute name="Width" type="double" value="100.0"/>
30                  <attribute name="PositionX" type="int" value="484"/>
31                </attributes>
32              </target>
33            </targets>
34            <core>
35              <marking type="Help"/>
36            </core>
37            <attributes/>
38          </annotation>
39          <annotation id="a2" type="textual">
40          <annotation id="a0" type="simpletextual">
41        </annotations>
42      </annotationset>
43      <annotationset versionnumber="1" status="annotating">
44        <historyitems>
45          <historyitem date="2019-02-26 16:46:56" comment="annotation started by JeanLuc" newstatus="annotating"/>
46        </historyitems>
47        <annotations/>
48      </annotationset>
49    </annotationsets>
50  </pandaannotation>

```

Figure 6-8 Example of annotation file

6.3.2. Mapping between annotations and artefacts

One of the key aspects of the model of annotations proposed is the support of targeting several heterogeneous artefacts with one annotation while keeping its integration on the representation of the artefact. The targeting system is facilitated with the independence of the annotations. Indeed, this independence consisting in considering annotations as a distinct type of artefact detach the lifecycle of the targeted artefact from the lifecycle of the annotation artefact. Thus, the lifecycle of the annotation artefact is not dependent of one artefact and can exist on its own.

Regarding the implementation of this targeting system, a modular approach has been chosen for supporting the variety of design artefacts. Indeed, each design artefact may require the usage of a tool dedicated to its creation. Thus, the annotation tool support can be attached as a plugin to these tool to support a unified management of annotations in the different tools for the other design artefact. A more detailed explanation on this modular aspect of the annotation tool support is given in the Chapter 7.

In this section, a presentation of the targeting mechanism from the point of view of the annotation file will be given. Then, a preview of the current implementation of the representation of annotations and its targets will be presented. After that, we will present two strategies for associating the annotation and its targets that were implemented. The first strategy is the integration of the annotation in the artefact editor as an implicit targeting mechanism. The second strategy is the integration of a preview of the artefact in the annotation editor.

- **Formal specification of the target in the annotation file**

The targeting system is based on the referencing of the set of the targets for each annotations. In the current implementation of the model, the targeting has two granularity levels.

The first level of targeting is the “Artefact level” which consists in specifying which artefact the annotation is related to. When adding a new target to an annotation, the implementation of the model will store information on the artefact such as its name, its extension, its path and the type of artefact if it is recognized by the annotation tool. Once the targeted artefact has been referenced in the annotation file, it is referenced in the annotation as a new target.

The second level of targeting is the “Fragment level” which consists in specifying a part of the targeted artefact as the target of the annotation. This level of targeting allows to define precisely the relevant fragment of the artefact which is relevant to the annotation. However, this targeting has a few technical challenges. Indeed, the targeting system has to be generic in order to be compatible with the different type of design artefacts but it has to be adapted for each model to support the selection of the fragment.

To tackle this technical challenge, the modular aspect of the implementation of the model and the extensibility of the model has been exploited. Indeed, in order to support this level of targeting, the editor managing the design artefacts has to implement the dedicated plugin of the annotation system by overriding the default placeholder code. This implementation will have three roles: specify a format in which the fragment can be serialized in the annotation model, allow the user to select a fragment to target, manage the reading and writing of the fragment on the annotation file, and manage the display of annotations in the editor. Thus, the tool managing the targeted artefact can identify the targeted fragment by the annotation.

- **Representation of the annotation and its targets within an annotation editor**

To facilitate the use the annotations compliant with this model of annotation, a tool support is required. The details of this annotation tool support is presented in the Chapter 7 and through the cases studies in the Chapter 8 and Chapter 9.

The Figure 6-9 below is a screenshot of the representation of an annotations with its target in the annotation tool support we developed for this PhD thesis.

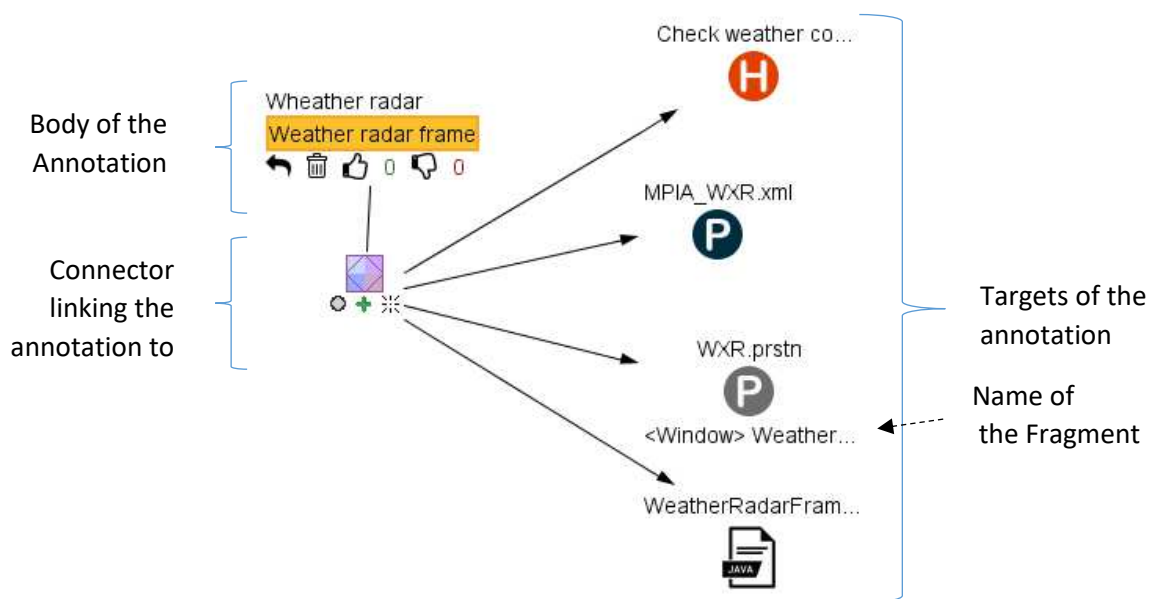


Figure 6-9 Example of annotation targeting several targets

This representation is composed of three parts: the body of the annotation which is represented by the widget on the upper-left side of the figure, the connector which is represented by the widget on the left side of the figure and the targets represented in the right side of the figure. Those elements are then linked together using lines and arrows.

In this representation of the annotation, we featured interactive icons that can be used to interact with the annotation. For instance, the “thumb” icons allow to give an opinion on the annotation. The “+” on the connector allows to fold or unfold the list of targets associated with the annotation.

Each target is represented by a first label indicating the name given to the artefact, an icon to give a representation of the type of artefact annotated, and a last label to indicate a name for the fragment targeted.

It should be noted that this representation of the annotation and its targets is a prototype and their final representation are still being developed.

- **First strategy: Integration of the annotations in the artefact representation**

This strategy of representing the relationship between an annotation and its targets consists in displaying the annotation in the view of the artefact within the tool allowing to edit the artefact. This strategy rely on the use of the geographical proximity of the annotation to its target.

This strategy is illustrated in the Figure 8-2 which present a prototyping tool supporting the annotation model presented in this chapter. This prototyping tool will be presented with more detail through the case study in the Chapter 8. In this figure, each annotations are highlighted with a green bullet. For instance, the annotation “A” and “B” are a combination of two annotations, a warning icon followed by a brief text underlined in yellow. These annotations are placed within the prototype. The annotation “C” is connected to a yellow token indicating that the annotation is related to the object under the token.

When parsing the annotation file, the implementation of the annotation plugin (more details are presented in the Chapter 7) checks if there is an annotation on the artefact edited by the main tool. Each annotation found by the plugin will be displayed in an overlay of the view of the artefact using position attributes specific for this representation as illustrated in the presentation of XSD file and in the case study in the Chapter 9.

The main drawback of this solution is that it does not allow to see the other artefacts targeted by the annotation directly.

- **Second strategy: Integration of the artefact representation in the annotation**

The second strategy for making explicit the relationship between an annotation and its target is the integration of a preview of the target in the annotation editor. The preview can be considered as a “citation” that is extracted from the artefact. This feature consists in clicking on a target of an annotation to display a screenshot of the representation of the artefact or to display an extract of the content of the target. This feature has been prototyped for images and text artefacts in the annotation tool support presented in the section 7.5.

Due to the variety of artefacts that can be targeted, this feature represents another technical challenge especially for its scalability. Indeed, like integration of the annotation in the artefact editor, this feature requires a specific implementation for providing a preview of a fragment for each artefact type.

6.4. Extensibility of the model

One of the goals of this model of annotation is to support the annotation activities during the design process. This include the editing of annotation on any design artefacts that can be used during the design process. The variability of the artefact being annotated imply that both the model and its implementation have to be sufficiently flexible to support their integration on the different artefacts while keeping the centralization of the annotations within the project workspace.

Thus, the current implementation of the model for annotations feature several extension points that can be used to extends the annotation support within the design process while staying compliant with the annotation model. This section will not go into the details of the extensibility of the tool support of the annotations but rather on the extensibility of the model. The extensibility of the tool support will be detailed in the section 7.4.1.

The first extension point of the implementation of the model is the targeting system as explained in the previous section. Indeed, each new implementation of the targeting system will allow the annotation system to support a new type of artefact and thus, allow the selection of a fragment of this type of artefact. The selected fragment will then be persisted according to the flexible notation given by the XSD file of the annotations. Indeed, as shown in the Figure 6-10 below, a target can feature any custom attribute that can be used to either encode or decode the parameters of a custom fragment selector for one artefact.

```

<!-- *****-->
<!-- TARGETS ELEMENT -->
<!-- *****-->
- <xs:complexType name="targetsElement">
  - <xs:sequence>
    - <xs:element name="target" minOccurs="0" maxOccurs="unbounded">
      - <xs:complexType>
        - <xs:sequence>
          - <xs:element name="fragments">
            - <xs:complexType>
              - <xs:sequence>
                - <xs:element name="fragment" minOccurs="0" maxOccurs="unbounded">
                  - <xs:complexType>
                    <xs:attribute name="id" type="xs:ID"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="attributes" type="attributesElement"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

Figure 6-10 XSD of the "targetsElement"

```

<!-- *****-->
<!-- ATTRIBUTES TYPE ELEMENT -->
<!-- *****-->
- <xs:complexType name="attributesElement">
  - <xs:sequence>
    <xs:element name="attribute" type="attributeElement" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

```

Figure 6-11 XSD of the "attributesElement" type

The second extension point is the customizable type of the annotation. This customization allows to create a new structure of data for annotations by assembling existing types (e.g. text, marking), by defining new types and by adding custom attributes that will be manipulated by a tool support. Annotations can be customized on their models and on their representation on the editor. This customization can be used for defining tailored annotations for a specific type of artefact or for a specific use by the design team, thus giving a convention for annotations through their representation. For instance, the example given by Bringay in [29] in which the color of a post-it is giving a meaning within the working group can be applied to this model by creating two types of annotation which will feature a specific rendering.

Overall, the flexibility of the annotation model is supported by the modular approach adopted for the annotation system. This modular approach allows to run custom code on annotations depending on the plugin used (i.e. the default implementation or a custom code overriding the plugin) at different extension points such as the parsing of annotations or the display of parsed annotations within an editor. The technical aspects of this modular approach will be detailed in the Chapter 7.

6.5. Conclusion

In this chapter, we defined an annotation model based on the W3C Web Annotation Data Model recommendation. This model has been adapted for tackling the specificities of a UCD design process which features various artefacts that evolve along the iterations and which are related to the same interactive system. Thus, this model promotes the independence of annotations toward their targets while insuring its genericity for adapting the annotations and its support on various artefacts and tools.

The independence of the annotations will facilitate their management within the design artefacts of a project and facilitate their centralization. Another aspect of the annotations that can be supported with the independence of annotations is that they can be attached to any targets in the scope of the project workspace regardless of the type of the artefact or their version. Thus, it is possible to annotate several versions of an artefacts, several representations of the interactive system with one annotation since it can be relevant to each of those targets.

However, the centralization of annotations within the project workspace raise two main technical challenges.

The first technical challenge is that making annotations independent from the artefact can have an impact on their readability and understanding. Indeed, pervasive annotations have the advantage of being in a context promoting their understanding through implicit meaning. For instance, the usage of brackets in a margin and the proximity of a written note give a clear visual representation of the association of the note with the area defined by the brackets. This pervasive visual representation is important since it notify the user that there is an annotation attached to the artefact even if it might be at the expense of the readability of the artefact. Thus, these implicit meanings given on annotations using their properties such as their positioning within the artefact should somehow be permitted in this new architecture of annotations in order to keep the advantages of pervasive annotations.

The second technical challenge is related to the specification of the targets of an annotation. Indeed, annotations in a project workspace should be targetable. However, the targeting system should be sufficiently generic to target any type of artefact but also extensible to allow an accurate targeting of a fragment within an artefact.

To overcome these two technical challenges, we propose an architecture for a tool support that will enable the use of annotations compliant with the model presented in this chapter. This architecture consists in a plugin handling annotations that can be implemented in various tools for managing both

the reading and display of annotations within any compatible editor as well as the creation of annotations and the targeting of artefacts of any type. This architecture will is presented in the Chapter 7. A presentation of the use of the annotation tool support is given in the case studies in the Chapter 8, Chapter 9 and Chapter 10.

Chapter 7. An architecture for integrating annotations made on diverse artefacts

Summary

This chapter is dedicated to the presentation of the tool support designed and developed within the ICS team from the IRIT for the annotations based on the model presented in the previous chapter. The section 7.2 gives an overview of the architecture defined for the support annotations referencing diverse artefacts and for managing them. The section 7.3 will be dedicated to the presentation of the necessity to use a common file repository for the tools used during a project. The section 7.4 details the different modules developed for the implementation of this architecture. The section 7.5 is used to present the tool developed to support the management of annotations of a project workspace.

7.1. Introduction

The two previous chapters were dedicated to the presentation of an annotation model aimed at the support of the annotations within the design process which include the management of the targeting of multiple artefact, the acknowledgement of the evolution of the targets, the versioning of the annotations and the different types of annotations. They also presented the different uses of annotations during the activities of the UCD process as well as the management of the annotation status and the targets of the annotation.

A tool support has been specified for supporting the annotations within the UCD process. This tool support consists in two parts. The first part is a module dedicated to the support of annotations that can be integrated to compatible artefact editors. The second part of the tool support is a standalone tool to manage annotations. This standalone tool is called ARMADILLO which stands for “Annotating by Referencing Models, Artefacts, Documents to Identify Logically Linked Objects”.

Overall, this tool support is aimed to help the development team to accomplish two main tasks:

- i) Annotate artefacts in the context where they are created;
- ii) Allow cross-referencing of annotations and multiple artifacts.

To do so, we propose a modular approach for the integration of the annotation management in existing tools editing design artefact as well as the usage of a common project workspace shared by these different tools.

7.2. Inner of the architecture

The independence of annotations regarding their targets is essential since they can be related to any set of artefacts or set of versions of artefacts: an annotation should not depend on one of these artefacts. Thus, a dedicated tool is necessary to assist users to create and edit annotations on their artefacts.

Due to the variety of annotatable artefacts (see 6.2.2), the development of the architecture for the tool support of annotations was done in two distinct parts.

The first part consists in a plugin that can be added to other editors of design artefact using the NetBeans Platform framework. This plugin has been developed by using the NetBeans Platform framework for the UI and the NetBeans Visual Library for the display and representation of the annotations on a canvas. The purpose of this plugin is to enable the integration of annotations within the artefact editor and make the creation, the edition and the display of annotations possible in the editor while viewing the artefact edited. So far, this plugin has been integrated in three different editors which will be presented in the use case in the Chapter 9.

The second part of the tool support is dedicated to the management of annotation artefacts using a standalone tool called ARMADILLO built as a Java application over the NetBeans Platform framework for the main architecture and the UI. The representation of the annotations is built using the Netbeans Visual Library which allow to create, edit and display widgets on a canvas. A more detailed presentation of ARMADILLO is given in the section 7.5.

Overall, ARMADILLO works as a project management system as shown by Figure 7-1 below:

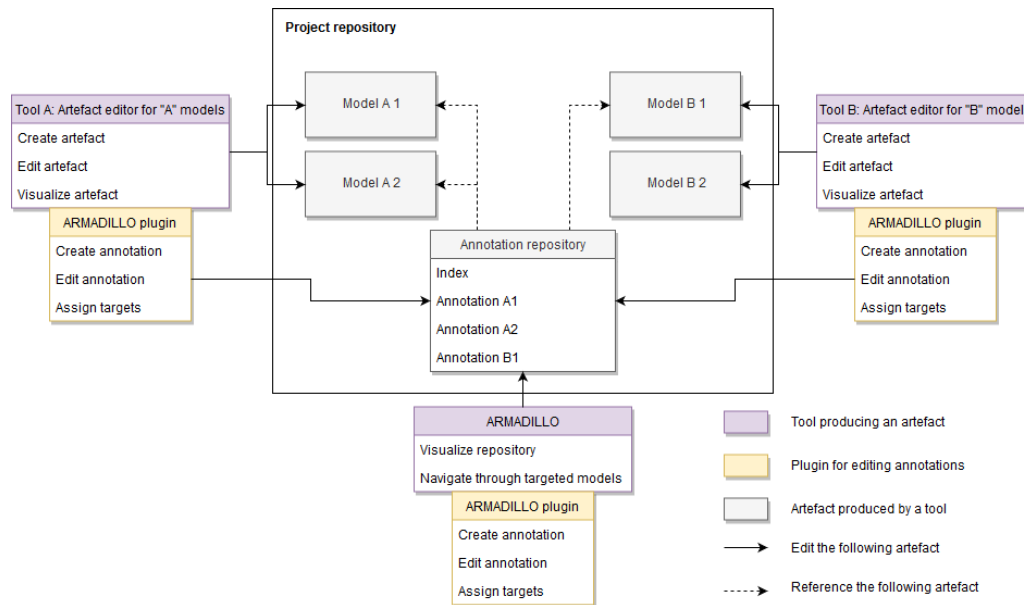


Figure 7-1 Overview of the tool and the files repository used by ARMADILLO

The central part of the Figure 7-1 shows the ARMADILLO repository which contains the list of artefacts which contain models describing the interactive system (models A1, A2, B1 and B2) the list of individual annotation files (one per artefact and per version of the artefact created, respectively annotation A1, annotation A2, and annotation B2) and an index file for regrouping all files in a project.

Even though the annotation plugin is restricted to tools built with the NetBeans Platform framework, the model and its implementation (i.e. the XSD and the XML) can be reused in other tools based on other technology after some development. Thus, the effort to integrate this model in other tools depends on targeted the environment.

For instance, if the targeted environment is using the JAVA language, the ARMADILLO plugin which manipulate the annotation models could implement an API to better support the interoperability with other tools. Then, this API would be usable in any tool after the development of features for displaying and manipulating annotations in the targeted tool.

However, for the integration of the annotation model in tools using other language, the annotation support would also require the implementation of a parser for reading and writing annotation files matching the XSD.

Furthermore, as will be presented in the third case study “Industrial transfer over eazly™”, it is also possible to only consider the key concepts of the annotation model proposed in the Chapter 6 such as the independence of annotations over their targets and to fully implement a new annotation support in another environment for commercial purposes.

7.3. Presentation of the file repository for managing annotations of the project

The tool support of the annotation model is heavily based on the management of a file repository for the project of the design of an interactive system. This repository is collecting the different design artefacts produced and gathered during the design process which include all files that contains the specification of the interactive system and all the annotations created over these artefacts.

This project repository is representing a valuable material for our studies which are listed below:

- **A repository for gathering the different versions of design artefacts to trace the evolution of the interactive system**

The project repository is supposed to contain the various design artefacts created along the design process and features their different versions for ensuring a sort of traceability of the design workspace. This traceability of the design workspace will then allow to follow the evolution of each representation of the interactive system. While we did not focus our studies on the traceability of the evolution of the design, we suggest that gathering these artefacts might be useful for defining an accurate representation of the evolution of the artefacts and listing the different milestones of a project.

- **A common workspace shared by editors of a tool suite**

The current tool support for annotations has been designed as a plugin for applications developed with the NetBeans Platform framework. This choice has been motivated by the usage of this framework in the CIRCUS modeling tool suite which is being developed by the ICS team at the IRIT. Indeed, this framework gather several tools built with the NetBeans Platform framework. Each of these tools are developed for supporting the design of a different model of an interactive system and the CIRCUS modeling tool suite aims at editing, running and inter-connecting those models

While each tool can be used as a standalone with a distinct project workspace, the CIRCUS modeling tool suite allow to create a common workspace that is used for the inter-connection of the models. An example of a CIRCUS project workspace is showed in the Figure 7-2 below.

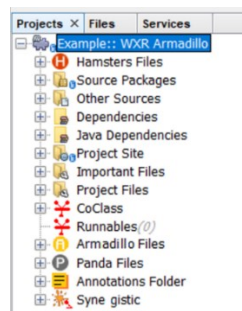


Figure 7-2 Example of a CIRCUS project workspace

In the CIRCUS framework, each standalone tool can be used as a module managing a type of model. When opening an artefact in the project workspace, the CIRCUS framework automatically loads the module managing this artefact. This mechanism has also been replicated for the management of annotations by the plugins attached to each standalone tool. Indeed, when creating an annotation on an artefact of the CIRCUS framework, a reference to the right artefact is automatically added in the annotation file. For instance, when creating an annotation on a HAMSTERS task model, the annotation added in the annotation file will feature a reference to the task model annotated. However, it is still possible to create annotations without an initial target in the ARMADILLO editor (see section 7.5) and then add each relevant target manually.

Thus, the annotation plugin can be seen as a contribution to CIRCUS which feature its own models that can be connected to other models of the interactive system.

- **A common repository for referencing artefacts**

As stated in the previous section 5.3, the targeting of artefacts is made by referencing the corresponding file in the annotation model and with a custom identifier for fragments of this artefact.

This referencing requires that the artefact should be somehow accessible in order to retrace the context of the annotations for a better understanding of this annotation. Moreover, to ensure the consistency between the content of an annotation and its targets, the versions of both artefacts should be memorized. Indeed, the evolution of a target may question the content of the annotation.

We also suggest that the context of the annotation should be saved in the project repository when possible in order to be able to understand annotations and especially tacit annotations. This context could be rebuilt by doing a snapshot of the targeted artefacts for keeping a trace of its current state when an annotation is targeting it with the condition that this snapshot is both possible and allowed. Indeed, there can be some restrictions on artefacts such as the confidentiality of the artefact or the size of the artefact.

In the case where the copy of the artefact is possible, the snapshot could be a copy of the targeted artefact, a screenshot of the relevant fragment or an extract of the artefact. In other cases, the current implementation of the targeting system allows to reference the name or the URL of the file with the date of the attachment of the annotation.

However, this solution might have some limits such as the overload of artefacts on the long term requiring to sort the repository and to manage the duplicated artefacts.

7.4. Architecture of the annotation plug-in

The annotation plugin is composed of three distinct modules. Each of the module is designed to support a different aspect of the support of annotations within the NetBeans Platform environment.

- **The “Annotation” module**

The “Annotation” module is the core of the plug-in. Indeed, this module contains the different classes that implement the annotation model and the interfaces to implements in order to customize and adapt the annotations to a type of artefact.

- **The “AnnotationUI” module**

The “AnnotationUI” module is dedicated to the definition of the UI of annotations. This module is composed of two distinct parts. The first part relies on the NetBeans Platform framework and consists in the definition of the menus and user interfaces that allow users to access to the features related to annotations. This part includes a menu located in the toolbar that allow to toggle the visibility of annotations (Figure 7-3.a), a palette of annotations for creating annotations using a drag & drop (Figure 7-3.b), two windows used to customize the stroke of drawing for free-hand annotations and the windows for setting targets and references on annotations (Figure 7-3.c).



a) NetBeans Platform toolbar including annotations menus and icon

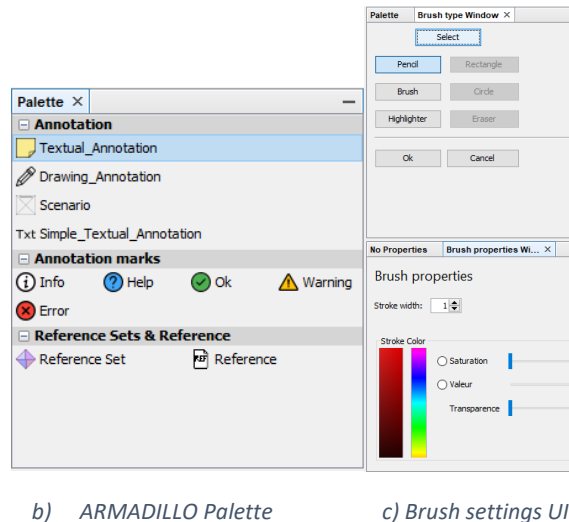


Figure 7-3 Overview of the UI of the ARMADILLO plugin

The second part of this module rely on the NetBeans Visual Library and is dedicated to the representation and interactivity of annotations using widgets on a canvas. Indeed, this part of the module is defining the UI of each type of annotation within the canvas, their registration in the palette presented in the previous paragraph, the interactivity of the UI of the annotation, and the impact of users' action on the model of annotations (e.g. moving a widget representing an annotation will edit its positions in the model of annotation).

An example of the UI defined in this module is presented in the Figure 7-4 below. This illustration corresponds to the representation of a textual annotation. This annotation is composed of two parts. The upper parts includes a title and a content that can be edited. The lower part is made of buttons that can be clicked for adding a reply, for deleting the annotation and for voting.



Figure 7-4 Representation of a textual annotation

Thus, this module is dependent from the “Annotation” module and is used as an interface for the user to manipulate their annotations files.

- **The “AnnotationProject” module**

The last module of the annotation plug-in is the “AnnotationProject” module. This module is used to recognize annotation files within the project workspace of applications built with the NetBeans Platform framework. Projects modules are essential in the CIRCUS framework for presenting a structured tree of a project. Indeed, each editor included in the CIRCUS framework define their project structure which list the different artefacts related to the editor. The Figure 7-5 below show an example of a CIRCUS project which include Hamsters files from the HAMSTERS task modeling tool (i.e. “Hamsters Files” node), PetShop files from the PetShop modeling tool (i.e. “CoClass” and “Runnable” nodes), Panda files from the PANDA prototyping tool (i.e. “Panda Files” node), the annotations files from the Annotation plug-in (i.e. “Annotations Folder” node), and lastly the Synergistic files from the TOUCAN tool (i.e. “Synergistic” node).

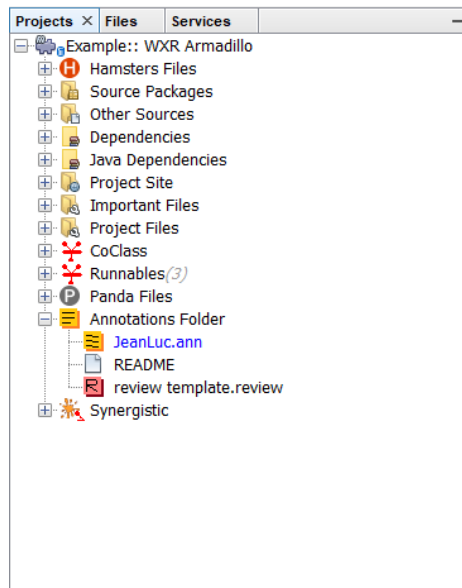


Figure 7-5 Example of a CIRCUS project

7.4.1. Integration of the plugin for its support in a new editor and its artefacts

The three modules of the annotation plug-in presented in the previous section are defining the main implementation for the support of annotations within the NetBeans Platform framework. However, the support of annotation within an artefact editor require the implementation of a new module that will act as an adapter for the editor. This adapter will gather every dependency to the annotation plugin to limit its adherence with the core modules of the artefact editor.

The adapter has three distinct objectives:

- **Integrate the editing and display of annotations within the artefact editor**

The first objective of the adapter is to integrate annotations features to the artefact editor as a standalone tool. This integration is done by allowing the display of the right annotations within the artefact display of the editor. Thus, only the annotations targeting the artefact being viewed in the editor should be displayed and this display should be adapted to the artefact viewer integrated in the editor.

The Figure 7-6 below shows the integration of textual annotations within the PetShop editor. In this figure, annotations are represented in yellow in the center (highlighted with a green rectangle), and the palette is represented in the right side which is highlighted with a red rectangle.

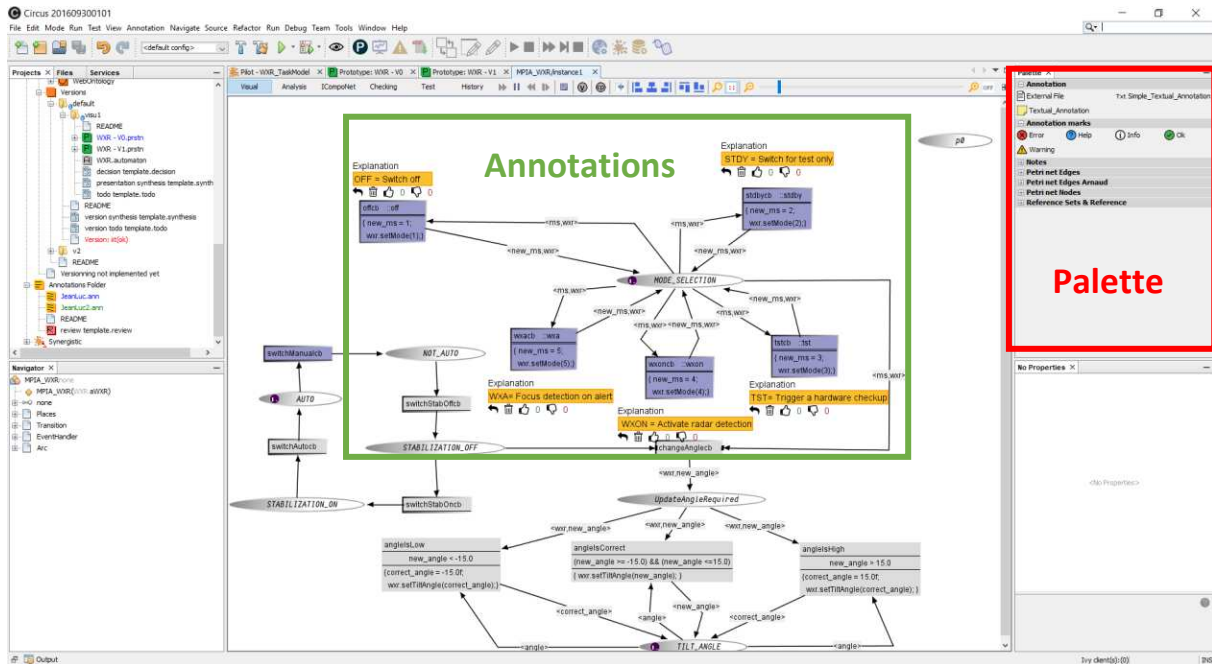


Figure 7-6 Screenshot of the PetShop editor

- **Define an adapter for targeting fragments of the artefact**

The second objective of the adapter is to define the fragment selector for the type of artefact manipulated by the editor. When annotations are created and an artefact is assigned as a target, the level of granularity of the target is a reference to the file that describes the artefact. For a fine selection of elements in an artefact, the custom fragment selector should define the element that can be targeted within the artefact (ex. zone, objects, etc.) and a UI to specify the fragment to connect to the annotation.

The fragment selector of the adapter will be used in three different contexts. The first context corresponds to the execution of the artefact editor as a standalone tool for selecting targets of the artefact being edited. The second context of use of this custom fragment selector is within the execution of the CIRCUS framework when a user wants to create an annotation and target different artefacts of different type. Depending on the structure of the artefact and of the architecture of the editor for this artefact, it is possible to consider the third context which correspond to the use of the fragment selector in the ARMADILLO tool presented in the next section. This tool is dedicated to the edition of annotation artefacts and allow to connect an annotation to any artefacts regardless its type. In order to support this third context, the editor should provide an API to interact with the artefact. This API should be sufficiently self-contained regarding the dependencies in order to avoid the necessity to retrieve every modules of the artefact editor within the ARMADILLO tool.

The Figure 7-7 below is a screenshot of the targeting system implemented in the annotation plugin. The left side of the window lists the different artefacts targeted by the annotation selected. This selection is only based on the referencing of the artefact as a file. The information of the file is listed in the upper right side of the window. The bottom right side of the window is used to list the fragments targeted by the annotation.

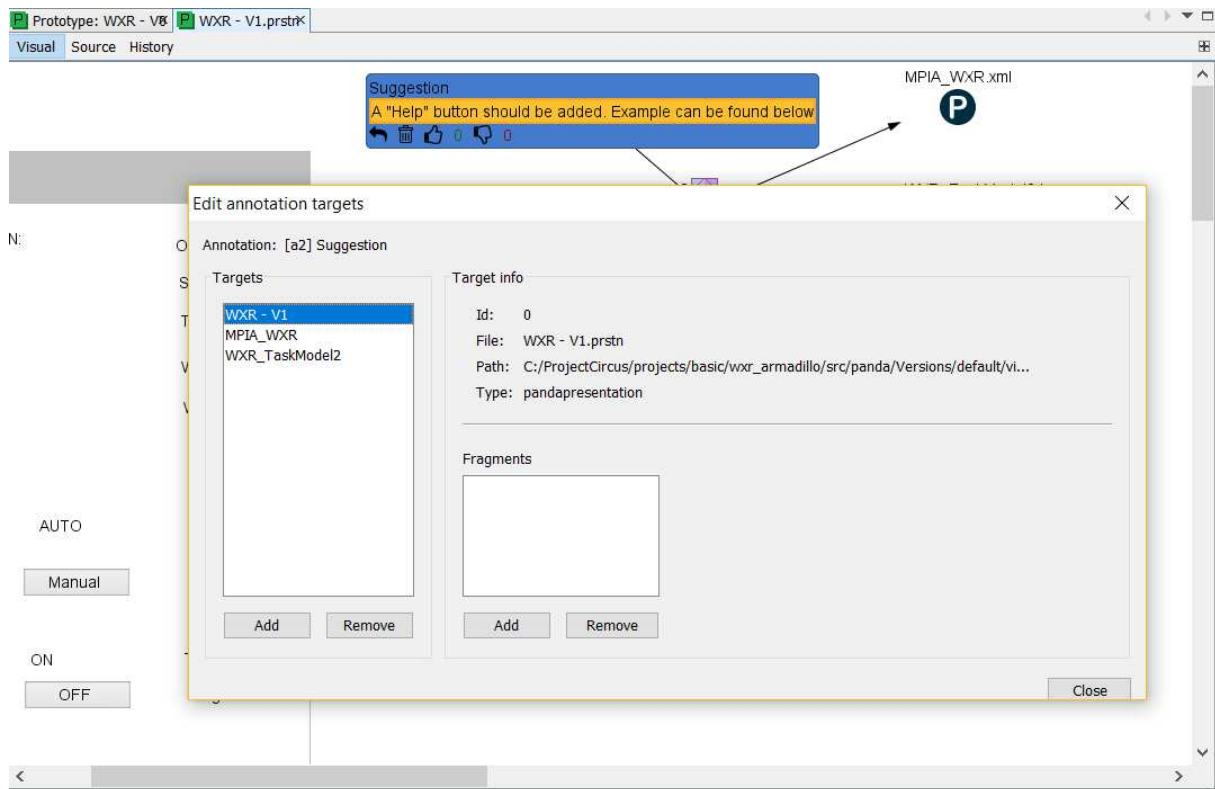


Figure 7-7 Screenshot of the targeting system of the annotation plugin

The “Add” button associated to the fragments is used to select a fragment of the selected target. This button is only enabled if a custom selector has been defined for the type of artefact selected (in this example, the type of the artefact is “pandapresentation”). By clicking on this “Add” button, the UI defined in the custom selector for “pandapresentation” files will be displayed as illustrated in the Figure 7-8 below:

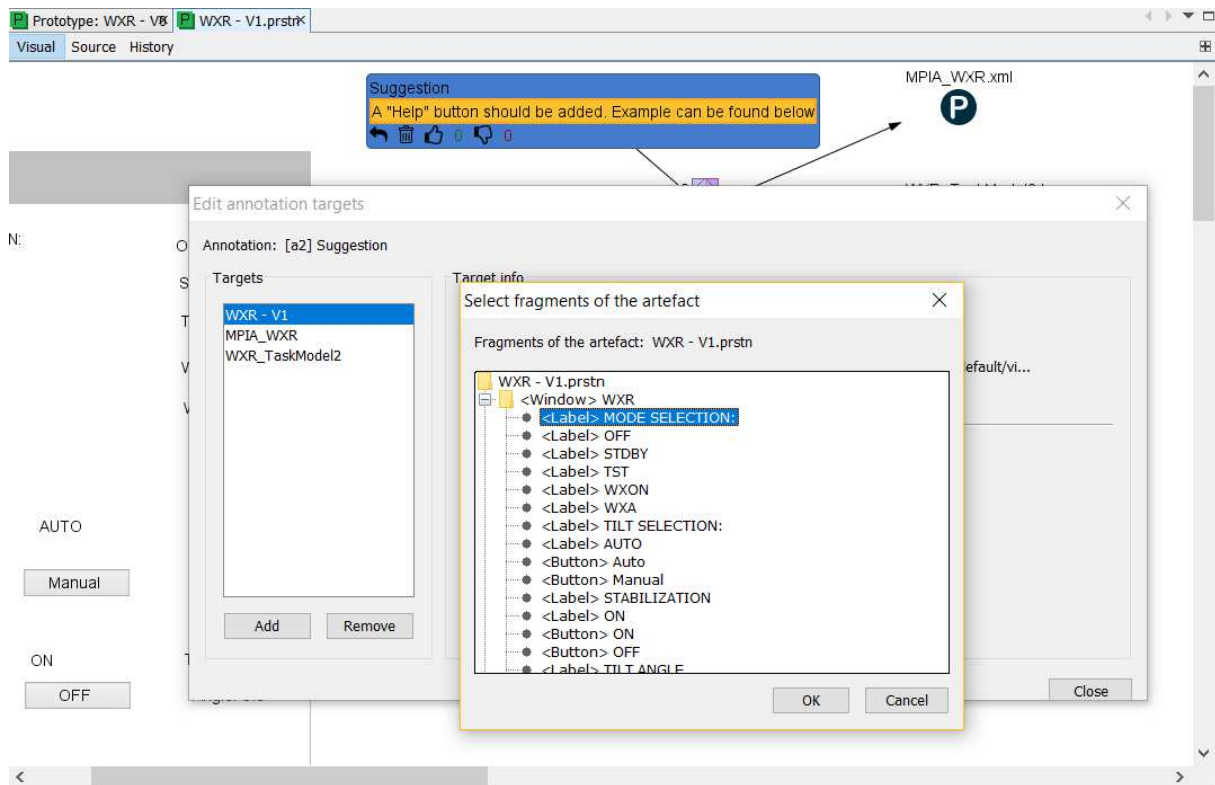


Figure 7-8 Screenshot of the fragment selection window for PANDA prototypes

In this implementation of the fragment selector, the PANDA adapter provides a selector based on a tree describing the different components of the prototype “WXR - V1”. The notation used in this prototype is as follow: “<Type of component> Name of the component”. In the figure, each components (e.g. labels, buttons) displayed are sorted into the container “WXR”.

- **Provide an extension point that can be used for customizing the annotations plugin**

The third objective of this adapter is to provide an extension point that can be used to customize the annotations support.

Using these extension points, custom code can be added to add a new type of annotation specific to the artefact, to support the targeting of fragments as detailed in the previous paragraph or to implement new features related to annotations.

For instance, a new type of annotation called “scenario” has been defined for PANDA prototypes as illustrated in the Figure 7-9 below:

Find flight

```

[V] Given I go to "Find flight"
[V] And I type "Paris" choose "CDG - Paris Ch De Gaulle, France" in the field "From"
[V] And I type "Dallas" choose "DFW - Dallas Fort Worth International, TX" in the field "To"
[V] And I set "valid date" in the field "Depart"
[V] And I set "valid date" in the field "Return"
[V] And I click on "Search"
[V] Then will be displayed "Choose Flights"
  
```

Figure 7-9 Example of scenario annotation

Scenario annotations are defined as a set of structured “steps” with a defined syntax and keywords. A step is defining either an action made by the user or an action made by the interactive system. A

feature has been developed in the PANDA prototyping tool that allow to check if the scenario can be run in the prototype. More details on the annotations types is given in the section 8.2.2.

7.4.2. Current status of the integration of the plugin in editors and their artefacts

So far we have developed ARMADILLO plugins for editors of PetShop, HAMSTERS and PANDA artefacts, all of which were built upon the NetBeans Platform framework. The ARMADILLO plugin is deployed in these artefacts editor as a palette which contains a set of basic types of annotation (such as textual annotation, drawing annotation, etc.) and structured annotations (such as scenarios and annotation marks, references, etc.) as shown by the Figure 7-3.b. The creation of annotations is then made by simple drag & drop interaction from the palette to the main editor area where annotations can be customized and connected to specific targets as shown by the Figure 7-10 below:

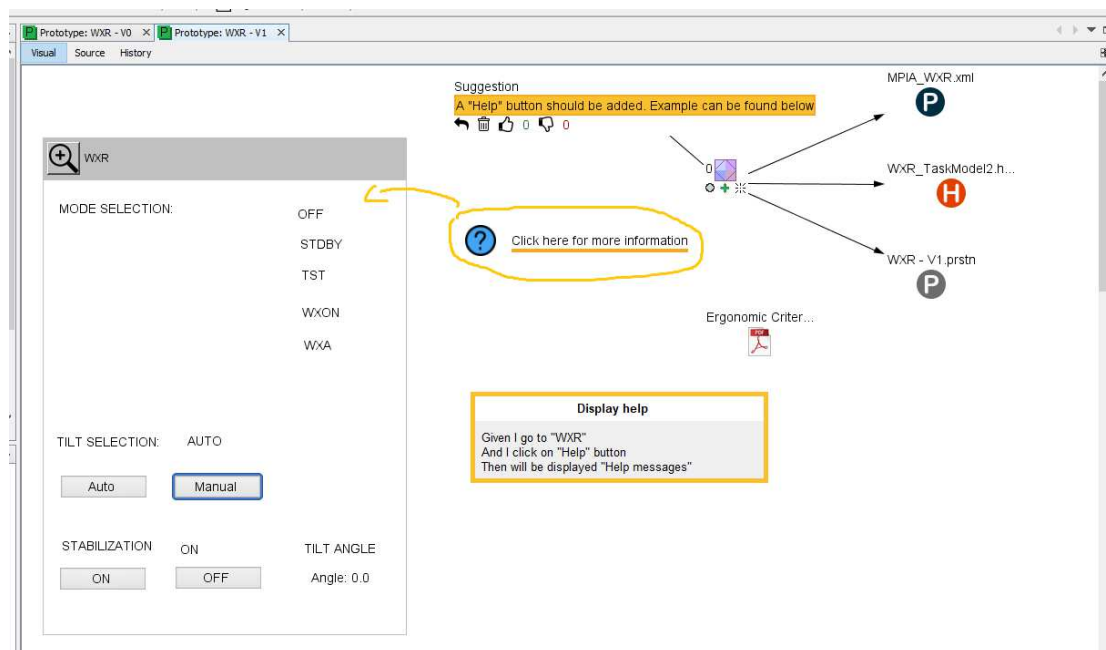


Figure 7-10 Annotations created over an artefact in the PANDA editor

7.5. ARMADILLO: An annotation tool support for managing annotations of a project

This section presents the ARMADILLO annotation tool designed for managing annotations which is the second part of the annotation tool support as explained in the section 7.2. This framework has been developed within the ICS team at the IRIT and is dedicated to the management of annotation artefacts within a design process project.

7.5.1. Description of the ARMADILLO tool support

While the module of annotation can be used to fulfill the first task identified (i.e. to annotate artefacts in the context where they are created), the ARMADILLO tool support is dedicated to the second task, namely to allow cross-referencing of annotations and multiple artifacts.

To do so, ARMADILLO has been designed as an annotation editor allowing users to visualize every annotations of a project repository, create independent annotations and to assign artefacts as targets of these annotations. Another feature of the ARMADILLO tool support is to provide users a way to navigate through artefacts of the project repository using annotations as hyperlink. To ensure the compatibility of the annotations made with others editors, ARMADILLO also integrate the annotation plugin described in the previous section.

ARMADILLO uses the index file briefly presented in the section 7.2 to parse individual annotations files of the project workspace and build a graph for visualizing the annotations in a project as it is shown at Figure 7-11. This graph is composed of several trees. Each root corresponds to an annotation created within the project workspace. The leaf of the trees are representations of the artefacts attached to the annotation. The icon of the leaf is a representation of the type of artefact if it has been recognized by ARMADILLO. Thus, the Figure 7-11 shows 5 annotations represented as yellow post-it. Each of these annotations are attached to two artefacts: an ICO model called “MPIA_WXR.xml” produced with PetShop and a PANDA prototype called “WXR - V0.prstn”.

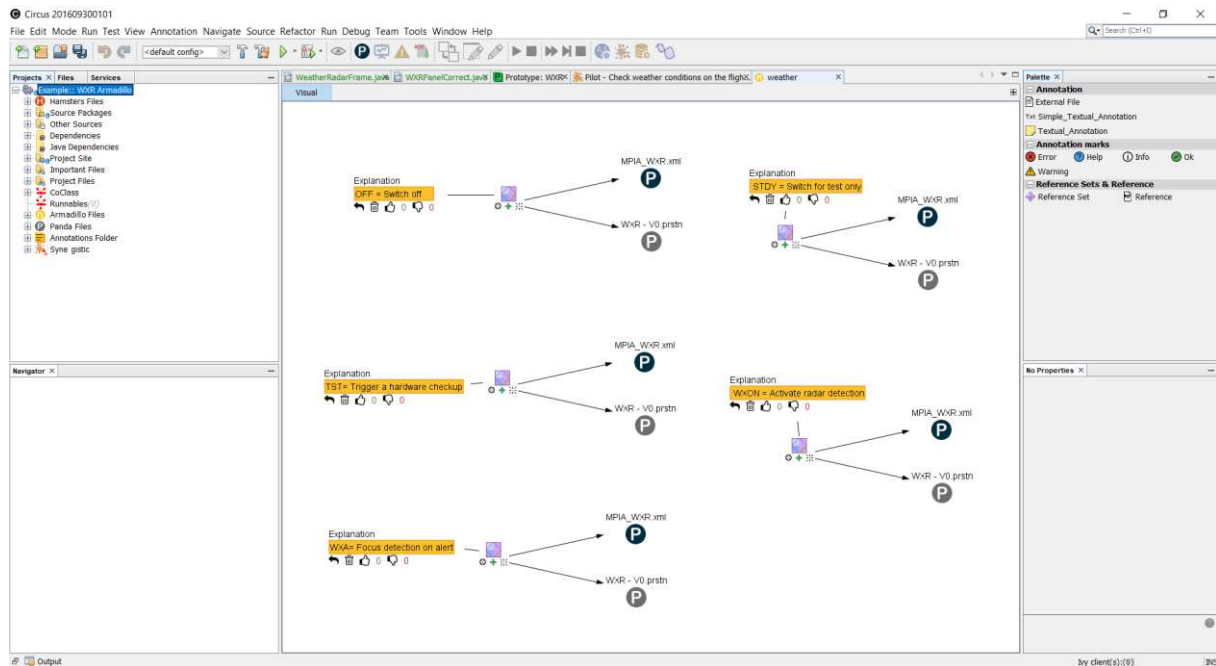


Figure 7-11 Overview of the annotations using ARMADILLO

As each annotation file contains references for the artefact it annotates, it is possible to navigate from a particular annotation to the corresponding the artefact. The index files records information about the artefact editor, so that we can launch the corresponding editor to see the annotations in the context where artefacts were built.

7.5.2. Architecture and extensibility of the ARMADILLO tool support

As stated in the section 7.2, ARMADILLO has been developed using the NetBeans Platform framework and the NetBeans Visual Library, making it compatible with the CIRCUS tool suite.

- **Main architecture of ARMADILLO**

The architecture of ARMADILLO is a bit different from the other editors of the CIRCUS too suite. Indeed, the particularity of this tool is that annotation file is the artefact edited by ARMADILLO. Thus, the modules of the annotation plugin are defining the core of ARMADILLO, namely the module “Annotation” defining the model, the UI module “AnnotationUI” and the module “AnnotationProject”.

- **Required dependencies of the ARMADILLO tool support**

One of the technical challenges of the ARMADILLO tool support is that this annotation tool should support the most possible number of artefact type from the design artefact used during the UCD process while being available as a standalone tool. Indeed, some features require a specific

implementation depending on the type of targeted artefact: the targeting, the preview of the target, custom annotation types. While a minimal and generic support is available in the core module of ARMADILLO, it features dependencies on a new module for each type of artefact supported. Thus, there is currently a dependency to a PANDA annotation module, a PetShop annotation module and a HAMSTERS annotation module.

- **Extensions of the ARMADILLO tool support**

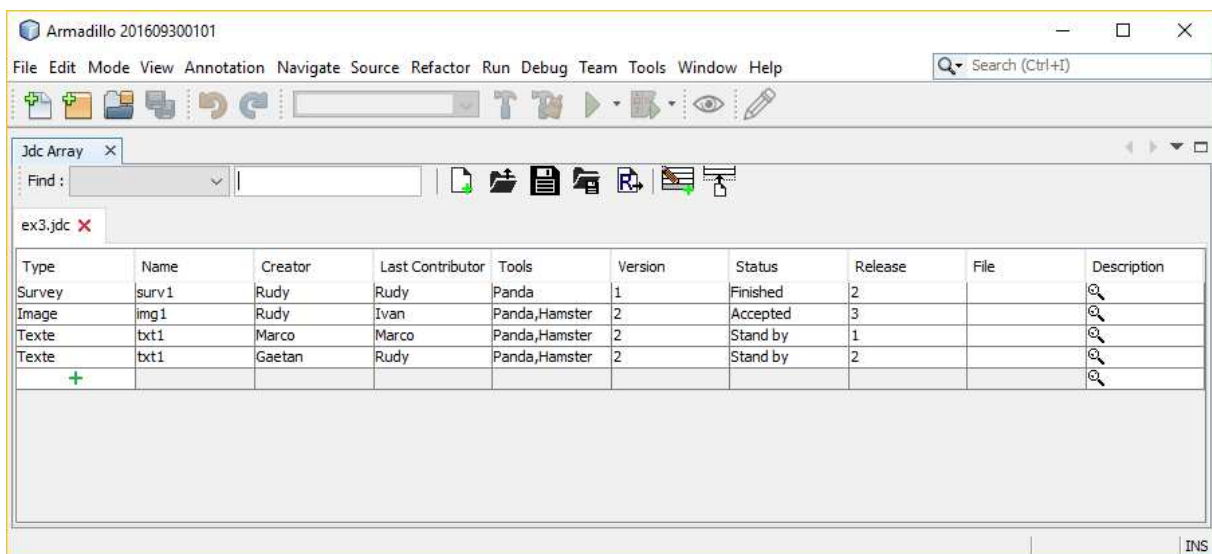
The ARMADILLO tool support can be extended in three different ways. These extensions methods have been mentioned in the section 7.4.1.

The first extension method is the support of a new type of artefact which consists in implementing the fragment selector for the artefact.

The second extension method is the creation of a new annotation type which can be specific to one type of artefact (ex. scenario for PANDA prototypes).

The third extension method is the adding of a feature using annotations which can defined new UI and processing on annotations.

For instance, a new component called “Design journal” has been developed by a group of students we supervised. This component is an early prototype integrated in ARMADILLO that has been designed within the time constraints of the students and consists in providing an alternative representation of the annotations of a design project using a table as illustrated in the Figure 7-12 below:



The screenshot shows the ARMADILLO 201609300101 application window. The 'Jdc Array' window is open, displaying a table of annotations. The table has columns: Type, Name, Creator, Last Contributor, Tools, Version, Status, Release, File, and Description. The data rows are as follows:

Type	Name	Creator	Last Contributor	Tools	Version	Status	Release	File	Description
Survey	surv1	Rudy	Rudy	Panda	1	Finished	2		
Image	img1	Rudy	Ivan	Panda,Hamster	2	Accepted	3		
Texte	txt1	Marco	Marco	Panda,Hamster	2	Stand by	1		
Texte	txt1	Gaetan	Rudy	Panda,Hamster	2	Stand by	2		

Below the table, there is a search bar with a 'Find:' label and a dropdown menu. The table also has a '+' icon at the bottom left and a search icon at the bottom right.

Figure 7-12 "Design journal" extension

Each annotations of the repository are listed in this table that can be sorted and give a set of selected attributes (i.e. Type of annotation, name, creator etc.). A search feature based on these attributes is also provided in the upper part of the JDC window. The user can specify which attribute he want to use as a search criterion using a combo box and then specify the value to search in the following input field.

[illegible]

This window is composed of 3 main sections. The upper part of the windows summarizes the information on the annotation already available in the table. The middle part of the window is dedicated to the display of a preview of the targets of the annotation. Each target is listed in the dropdown menu which can be edited using the “+” and “-” buttons. The bottom part of the window can be used to add a comment to the annotation.

7.6. Conclusion

99

ARMADILLO can be considered as an evolving platform based on a modular approach in which each module is providing additional features for the support of the annotation activities.

The modular architecture of ARMADILLO has been designed for ensuring the flexibility of the tool and for ensuring a generic tool support for managing annotations within the targeted environment, namely the NetBeans Platform framework. Thus, a module can be adapted to any tool using this framework in order to export features to other editors if the feature is relevant in the context of the editor to enrich.

Several features have been identified for the evolution of ARMADILLO to help the design team during activities of the UCD such as the filtering of annotations, a decision system which would list design choices connected to the different fragments of artefacts using annotations, providing a timeline displaying the different events on artefacts during the design process. These features are related to the need for the traceability of the design process, and to the need to manage annotations of the project workspace.

So far, the ARMADILLO plugin has been implemented in three different tools which will be presented through the two cases studies in the Chapter 8 and the Chapter 9.

Chapter 8. The PANDA ecosystem

Summary

This chapter presents the integration of the annotations modules in an editor through a case study for prototyping the UI of a flight booking system.

The first part of this chapter will give an overview of the PANDA prototyping tool. In the second part, will be presented the features of the PANDA prototyping tool over a case study called “Flight Booking application”.

8.1. Introduction

During the course of this PhD thesis, we designed a framework called the “PANDA ecosystem” based on a prototyping tool we developed called PANDA which stands for Prototyping using ANnotation and Decision Analysis. A more detailed presentation of the PANDA ecosystem and its architecture can be found in the Annex 7. Parts of the work presented in this chapter and the annex have been published in [105; 106; 107; 108; 111].

This framework allows us to illustrate the integration of the ARMADILLO plugin within a new editor for featuring annotations over new models and artefacts as well as to illustrate how annotations can be processed for providing new features over the artefacts.

8.2. The PANDA prototyping tool

8.2.1. Overview of PANDA

As the other tools of the CIRCUS tool suite [104] which encompasses other tools for engineering interactive systems, PANDA has been designed using the NetBeans Platform framework. PANDA is a prototyping tool which allows to create medium-fidelity and interactive prototypes by specifying both the dialog and the presentation of an interactive system.

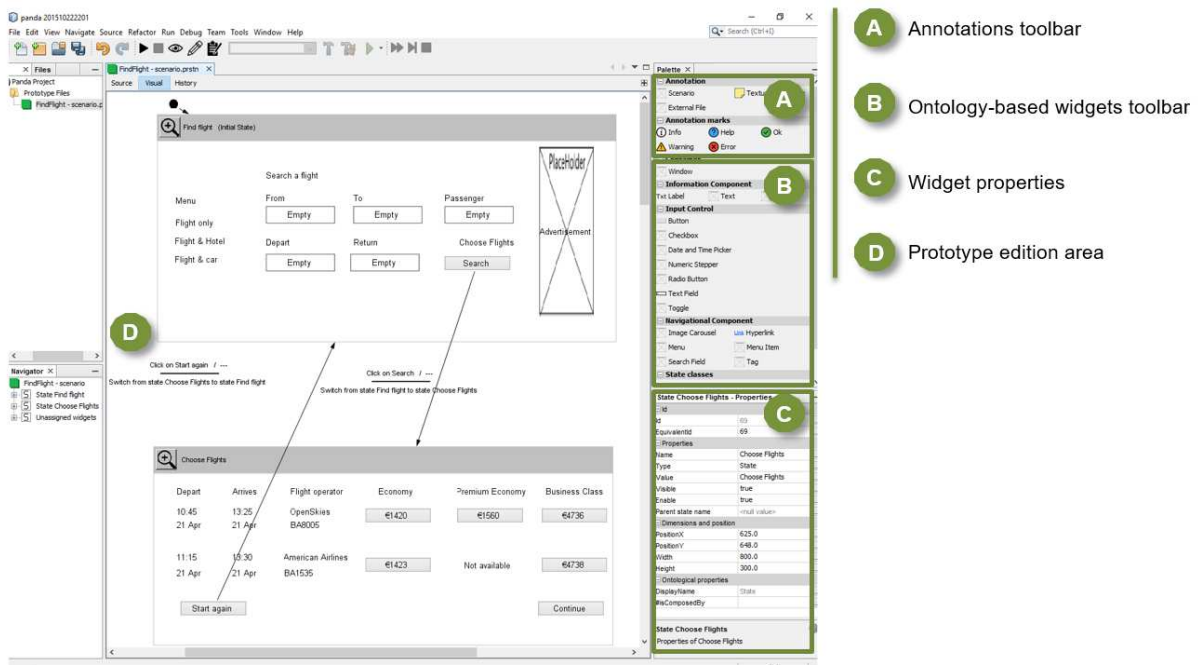


Figure 8-1 Overview of the interface of PANDA

The PANDA prototyping tool has been designed to provide a prototyping support in the CIRCUS tool suite. This tool support focuses on the modeling of prototypes to formally represent the interactive system. Thus, this prototyping tool allows to model the dialog of the interactive system using automata

and the presentation associated to the states of the automata. A generic presentation model has been defined to support the prototyping of the UI of an application.

Prototypes produced with this editor will be used within the PANDA ecosystem as the main artefact for ensuring the traceability of the design process. Indeed, prototypes are partial representation of interactive system, thus, they convey a representative view of the current goal of the design process. This goal can be refined or altered during the course of the design process. Thus, we suggest that tracing the evolution of this artefact and the reasons of the evolutions would be feasible by structuring the different versions produced within the project workspace of the prototype and by using annotations to connect those evolutions of the prototypes with the other artefacts of the project.

8.2.2. Integration of the annotations plugin in PANDA

Once the prototype has been created within PANDA, it can be presented to users or during meetings for discussions and evaluation. It can be evaluated on both the representation of the dialog and the user interface regarding the users' needs, the client's constraints and the requirements. The persons who participate to the evaluation and discussions can edit the prototype to express themselves through annotations or modifications of the prototype. The communication around the prototype can be worth noting since valuable information can emerge. For instance, we can note data on users' tasks, feedbacks, suggestion of modifications, ideas or problem reports. Thus annotations can be effectively used as a support of communication and collaboration since they add information and refers to a concrete artefact (e.g. a widget, a window), they support the memorization, the planning and can be used as a support to draw the attention for future analysis [52]. PANDA proposes six types of annotation that are illustrated in Figure 8-2, as follows:

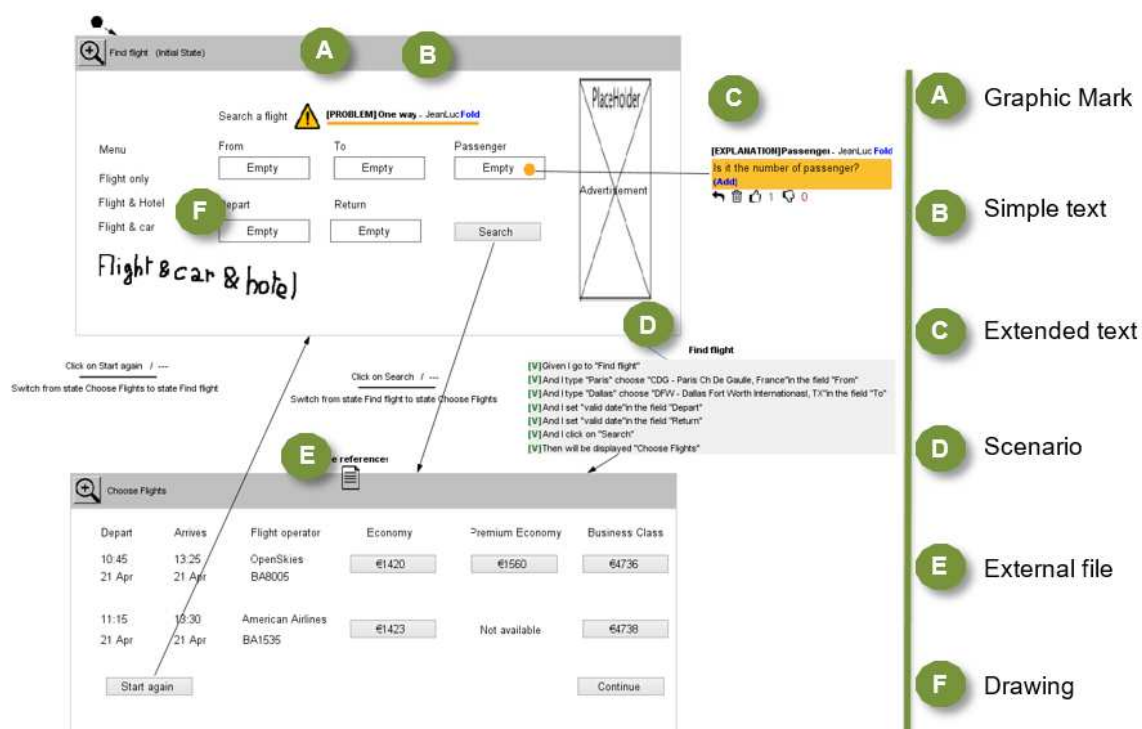


Figure 8-2 Overview of annotations supported by PANDA

Graphic Marks (Figure 8-2.A) feature icons used to place hotspots over a prototype. The meaning of the iconic marks is arbitrary and they are meant to support quick recognition of areas that needs further attention and/or development. So far, we propose 5 types of marks: Info, Help, Ok, Warning

and Error. They can be self-sufficient or they can be completed with another annotation to explicit the details.

Simple Texts (Figure 8-2.B) are, as the name indicates, textual annotations that can contain any kind of comment, question or suggestion of improvement.

Extended Texts (Figure 8-2.C) are similar to Simple texts but they can be used to create discussion threads around a subject. Indeed, doing so might promote collaboration and argumentation about a matter on the prototype. Extended Text annotations include a voting system so that we can associate a weight for comments that corresponds to the results of a discussions, for example with the users and/or clients. Indeed, those factors might have an influence on the decisions that could be made based on the annotations in later iterations of prototyping. We suggest that extended text can also be used as a planning tool featuring a checklist. However, other features should be included to fully support the planning like the description of the task, the deadline, the person in charge of the task [51] and the status of the advancement of the task.

Scenarios (Figure 8-2.D) are a type of annotation that refers to a list of tasks that can be performed with a prototype. A first use for this annotation is in replacement for the formal description of the behavior. Scenarios in PANDA are structured text. A scenario starts with an initial state and the keyword "Given". Then a set of steps are described in each following line. A step defines an action to perform. This action is described with a verb (e.g. "I click on") and a target (e.g. "Search"). An action can be completed with data (e.g. "I type "Paris" in the field "From""). Lastly, the scenario ends with a final action made by the system indicating a final state.

External Files (Figure 8-2.E) are used to enclose any kind of document located on the computer directly on the prototype and put it where it is relevant. For instance, it can be an audio file, a video from a recorded usage of the prototype or a specification document. By clicking on the annotation, the Operating System opens the file associated.

Drawing annotations (Figure 8-2.F) are the more polyvalent since it allows the annotator to create any freehand drawing to represent anything. On top of that, annotating with a pen is less cognitively demanding than with a mouse and a keyboard [112] and those annotations stand out from the underlying document making them easier to find. A drawing can be used to show, illustrate an idea, write something or emphasize elements by underlining, circling similarly to annotation on paper.

All of these annotations can be placed on the prototype or next to it. The proximity of the annotation is one way to establish its connection with its target [28] like we have done for the graphic mark annotation and the simple text in Figure 6. However, we also developed an anchoring system that allows to create bullets to represent explicitly the target of the annotation as illustrated in Figure 6 with the extended text annotation.

Once annotations have been created, their management should not be neglected. Indeed, over time, the number of annotations can grow significantly and thus, making their exploitation harder. Moreover, transient annotations [28] are not relevant during the whole design process (e.g. a modification request handled or a planned task done) but it might be interesting to keep a record of it to understand the progress of the design process. As a consequence, annotations should be updated to take into account their status of advancement

8.3. Case study

In this section, a case study will be presented to demonstrate the use in practice of the PANDA prototyping tool and of its operationalization of the model combination. This prototyping tool is representing the fragment already developed for the PANDA ecosystem.

The details of the building of the prototype using the PANDA ecosystem can be found in the section 7.4.1 of the Annex 7.

8.3.1. Informal presentation of the case study

The “Flight Booking application” is a common web application that allows its users to book a flight after finding it among the available flights. This booking of a flight consists in three main tasks: specify the flight to book, select a suitable flight among the available flight by comparing them, and validating the booking by paying.

In the first task, the user should specify the information of the flight he want to book. This include the area of departure and arrival, the date of the travel, the one-way or return travel, the selection of direct flight only, the number of passengers and the flight class. Other optional criteria can be specified by the user in this step such as the booking of a hotel or the car hiring.

In the second task, the user has to select the most suitable flight matching his needs and requirements among the flight proposed by the application. Indeed, during this task, the application is displaying every flight matching the criteria input by the user in the previous task. Each flight is characterized by several parameters: the hour of departure, the hour of arrival, the duration of the flight, the price, the company, and the number of connecting flight.

In the last task, the user has to validate his choice by inputting his personal information prior to proceed to the payment of the flight. In this step, the user can add to his purchase several options such as additional baggage, services, and seat selection. At the end of this task, a receipt is produced and the ticket can be printed by the user.

8.3.2. Automated test of prototypes with a scenario

Regarding the validation of the prototype created with the PANDA prototyping tool, an automated test feature using scenarios has been developed and integrated in the Annotation plugin as a custom annotation type for PANDA prototypes [105].

The implementation of scenarios annotations is an adaptation of scenarios as defined within User Stories by North [109] and Cohn [110]. Scenarios give a step-by-step description of task performed by users using an interactive system. The template of these User Stories is presented in the figure below:

```

Title (one line describing the story)
Narrative:
As a [role]
I want [feature]
So that [benefit]
Acceptance Criteria: (presented as Scenarios)
Scenario 1: Title
Given [context]
  And [some more context]...
When [event]
Then [outcome]
  And [another outcome]...
Scenario 2: ...

```

Figure 8-3 Template for specifying User Stories as defined by North [109] and Cohn [110]

Each scenario is defined by a title and is composed of three distinct parts: a context in which the scenario can occur introduced by the keyword “Given”, an event triggering the scenario introduced by the keyword “When”, and the outcomes of the scenario introduced by the keyword “Then”. Each of those parts can be enriched using the “And” keyword to either add more context, more conditions on the triggering of the event or more outcomes. Each statement of a scenario is called “Step”.

The details on the methodology to create a scenario is described in a previous paper [105].

In the current implementation, scenario are structured annotations which uses the keyword presented in the previous paragraph and describe tasks (either from the user or the system) to perform sequentially. A task consists in a verb and can feature an associated value introduced by quotation marks as illustrated in the Figure 8-4 below:

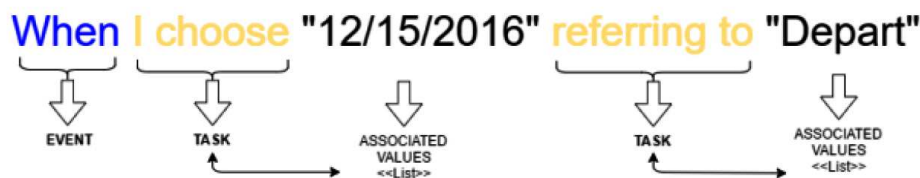


Figure 8-4 Structure of a step from a scenario specified in the PANDA annotation tool support

In the PANDA prototyping tool, each scenario is attached to two state of the prototype to define its context introduced by the keyword “Given” and to define the outcome introduced by the keyword “Then” as illustrated in the Figure 8-5 below:

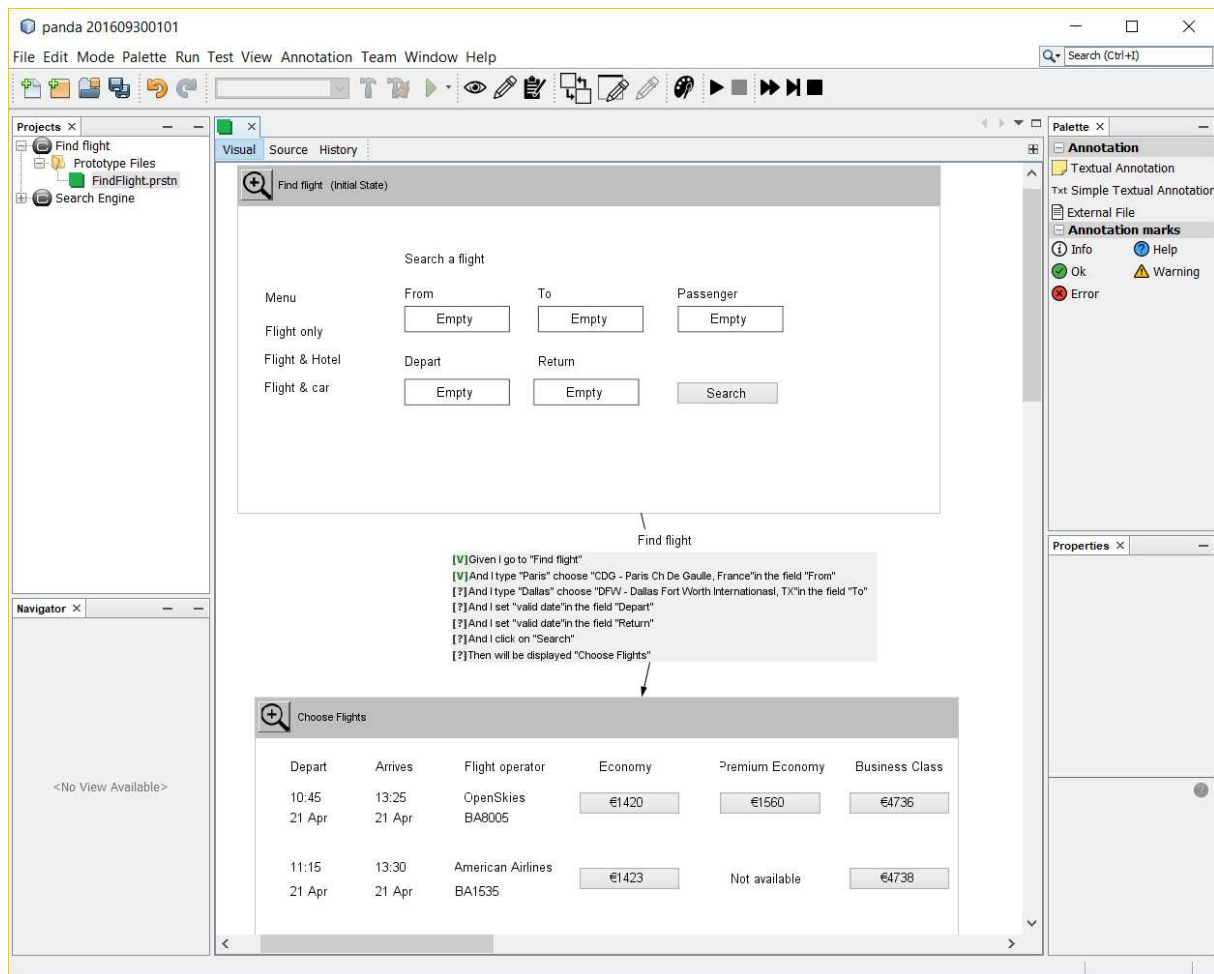


Figure 8-5 "Scenario" annotation attached to two states in PANDA

A scenario is tested step by step and is always related to a state of a prototype. The test of a step consists in an analysis of the state of the PANDA prototype (presentation, dialog and the ontology) to check if the task described in the step being tested can be performed using the prototype in the state tested.

To do so, the testing engine starts by parsing the step described in the scenario to try to identify the following elements when they are present: the interaction made by the user, the data input by the user and a widget targeted by the interaction. In the example of the Figure 8-6, the action "type" is identified based on the library of verbs known by the testing engine and it is associated with the value "Paris" identified by quotation marks. The "And" keyword indicates that there is a second interaction made by the user identified as "choose" with the value "CDG - Paris Ch De Gaulle, France". The locator "in the field" is used to identify the target of the interaction of this step. The value "From" is thus identified as the name of the widget targeted.

And	I type	"Paris"	and	choose	"CDG - Paris <u>Ch</u> De Gaulle, France"	in the field	"From"
-----	--------	---------	-----	--------	---	--------------	--------

Figure 8-6 Example of parsing of one step

After the parsing of the step, the testing engine tries to map the step to the prototype and check their compatibility. To do so, the testing engine looks for a widget whose name corresponds to the identifier parsed. After finding a matching widget, an analysis is done for each interaction and data input parsed. Indeed, the properties defined in the ontology used when designing the prototype are

analyzed to check if the widget support the interaction described by the step of the scenario. Only the current ontology loaded is checked since a widget can feature several properties from several ontology, but only one ontology can be active at the same time in the PANDA prototyping tool.

For instance, in the step showed in the Figure 8-6, the testing engine will look for a field called “From” and check if it supports the interaction “Type”. The Figure 8-7 below list the properties of the widget named “From” as stated by the first property visible in the table. The highlighted property shows that there is a property “#typeInTheField” defined in the ontology used when the widget has been created. Thus, the testing engine is recognizing the verb “type” within this property. However, the “choose” interaction has not been recognized in the list of properties of the widget. Thus, the testing engine assume that the step is not supported by the prototype and thus mark this step as failed as indicated by the red “X” on the left side of the step as shown in the Figure 8-7.

Text_Field From - Properties

+ Id		
- Properties		
Name	From	...
Type	Text_Field	...
Value	Empty	...
Visible	true	...
Enable	true	...
Parent state name	Find flight	...
+ Dimensions and position		
- Ontological properties		
#resetTheValueOfTheF		...
#typeInTheField		...
#setInTheField		...
DisplayName	Text Field	...
#typeInTheField		
Displays the property #typeInTheField from the ontology		

Button Search - Properties

+ Id		
- Properties		
Name	Search	...
Type	Button	...
Value	Search	...
Visible	true	...
Enable	true	...
Parent state name	Find flight	...
+ Dimensions and position		
- Ontological properties		
#clickOnReferringTo		...
#clickOn		...
#clickOn		
Displays the property #clickOn from the ontology		

Scenario steps:

- [V] Given I go to "Find flight"
- [X] And I type "Paris" choose "CDG - Paris Ch De Gaulle, France" in the field "From"
- [X] And I type "Dallas" choose "DFW - Dallas Fort Worth International, TX" in the field "To"
- [V] And I set "valid date" in the field "Depart"
- [V] And I set "valid date" in the field "Return"
- [V] And I click on "Search"
- [?] Then will be displayed "Choose Flights"

Figure 8-7 Properties of “From” and “Search” widgets in the PANDA prototype and the result of the scenario tested

On the other hand, the step “And I click on ‘Search’” is marked as valid with a green “V” since a “Search” button supporting the “#clickOn” property has been found within the state “Find flight”. As for steps that have not been tested yet, a black “?” is displayed on the left side as illustrated in the last step of the Figure 8-7.

Thus, if any of the tasks of one step is not possible, the step is considered incompatible with the current state of the prototype and thus, is marked as failed with a red “X”. Otherwise, if the step is considered as compatible, the step is marked with a green “V”. The support of a scenario by a state is defined by the support of each steps of the scenario: if at least one step is not supported by the state being tested, the scenario is considered as incompatible with the prototype.

Overall, while the testing feature introduced by scenario is a development toward the testing of multi-artefact for Behavior-Driven Development as explained in [111], it is one of the possible extensions of

the PANDA ecosystem made possible thanks to the formalization of the models manipulated within the framework. Moreover, due to its modular architecture, several extension points can be added to improve the support of automated test. For instance, a data model can be integrated within the list of models manipulated within the PANDA ecosystem. This data model could then be analyzed with the prototype and check if the UI specified support the display and the input of the data manipulated as detailed in a specification document. This analysis could be processed similarly to the scenario testing feature.

8.4. Conclusions

This chapter presented the PANDA ecosystem which include the integration of the ARMADILLO plugin for the management of annotation. This integration allows to create six different types of annotations based on the annotation model presented in the Chapter 6 over PANDA prototypes. On top of that, this framework shows that annotations can be processed for providing various features such as the automatic testing as presented during the case study.

Parts of this ecosystem has been presented at the EICS 2016² conference [108] and has also been used as a support for running automated test in various publications [105; 106; 107; 111].

While only a few modules have been implemented with some limitations on their current implementation, this framework has already been integrated within the CIRCUS tool suite. Further development can be planned for the support of the other modules of the ecosystem, namely the versioning and the decision modules.

Other evolutions have also been identified within this ecosystem such as the integration of more artefacts representing the interactive system such as the data model or a specification of the backend. Indeed, the presentation and the dialog of the interactive system are not the only representations that can evolve during the iterations of the design process.

The next chapter will be dedicated to the presentation of an integrated view of annotations within the CIRCUS framework for featuring various models and connect those using annotations through a case study called the “Weather Radar application (WXR).

² <http://eics.acm.org/2016/program.html>

Chapter 9. Integrated view of annotations on CIRCUS

Summary

This chapter presents the use of annotations created over design artefacts to provide an integrated view of a project through the case study of an application called “Weather Radar” (WXR).

This chapter starts with the presentation of a few models that can be used during the UCD to represent and to specify an interactive system. These models will be included in the study of the framework. The second part of this chapter presents the case study showcasing the model-based approach. This approach is supported by a framework for gluing together models featured in artefacts of the design process and consists in an integrated view of annotations within the CIRCUS tool suite.

9.1. Introduction

Prototypes are the most common type of artefacts used to specify design solutions for interactive systems during the UCD process. Prototypes feature a concrete yet partial representation of some specific aspects of an interactive system and they can be used to explore many design alternatives before implementing the final product [95].

It is interesting to notice that prototypes are useful and necessary but they are not sufficient to fully specify an interactive system. On one hand, prototypes are not self-explicative, which is illustrated by the fact that annotations have to be used to explain for instance the use of icons in a design. On the other hand, prototypes cannot directly inform other important aspects of the interactive system such as what goals can be achieved with the help of the tool and how the system will process the information provided by users. To capture and make such information explicit, other artefacts such as task models, dialog models and interaction models must be used.

Many intermediate artefacts are created, tested, revised and improved until the development team produces a validated version of the full fledge system. Each of those artefacts are answering different needs of the design process and each of them are specifying or representing a dedicated aspect of the interactive system. However, it should be noted that many artefacts can share the same type and share similar goals (such as low-fidelity prototypes and high-fidelity prototypes, or two versions of the same artefact).

Thus, many design artefacts are produced during the design process and each of those artefacts gives a different representation of the interactive system. In this chapter, we propose a generic framework for structuring those artefacts and for allowing designers to attach the same piece of relevant information on different artefacts using annotations.

This end-goals of this framework are the following:

- i) To enrich artefacts used to specify an interactive systems with any kind of complementary information that would be used to support decision making along the development process ;
- ii) To cross-reference design decisions to all artefacts at a concern;
- iii) To follow the evolution of the design decisions over the evolution of artefacts along the development of interactive systems.

This framework is based on establishing a relationship between artefacts or fragments of artefacts by using annotations. We suggest that by manipulating formalized artefacts and by using a model-based approach, it should be possible to provide a generic and an interoperable annotation system that feature an accurate targeting system. This annotation system can then be used to provide an integrated view of the artefacts of the project workspace.

9.2. The different models and design artefacts representing the interactive system

It is interesting to notice that model-based artefacts (such as task models and dialog models) provide complimentary information for the interactive systems and they might co-evolve with prototypes along the development process [116]. This relationship between artefacts implies that related artefacts should be updated together through the iterations in order to keep a consistent view of the interactive system and its context during the design process [104]. So that if a design decision is made causing an update in an artefact (for example adding a new button to a prototype for supporting of new requirements such as navigating among windows), it will ultimately affect other artefacts (for example, update the task model and the dialog model describing the inner system behavior).

In the section 6.3.2, we presented the two level of targeting supported by our annotation model: the artefact level and the fragment level. The specification of a target for an annotation at the artefact level is compatible with any type of artefacts. However, this level of targeting is not accurate enough for providing a connection of models that can be used efficiently by the design team. Thus, it is preferable to use the fragment level of targeting in order to provide connections that can give a good insight of the repercussions of design decisions and of complementary information to the models.

However, the targeting at the fragment level requires a specific implementation for specifying what fragment can be annotated and how it should be specified. Thus, this section will review the different models for representing and for specifying the interactive system which are currently supported by this framework at the fragment level of targeting. The details on the support of this level of targeting is explained through the presentation of the “adapter” in the section 7.4.1.

9.2.1. Annotation using ARMADILLO

The first model included within the framework is the annotation model which is the basis for establishing connections between artefacts using the targeting system. The ARMADILLO annotation model is dedicated to the description of annotations made on design artefacts of interactive systems. This model is detailed in the Chapter 6 and its tool support is presented in the Chapter 7.

The ARMADILLO annotation model is used to describe annotations that can be made on design artefacts. This model is generic so that it can be used for annotating various type of artefacts and extensible so that custom annotations can be specified for certain types of artefact.

This model is also focused on the independence of annotations. Indeed, annotations are defined as autonomous entity which are linked to related artefacts or to related fragments of supported artefacts. These links are defined by specifying references to those artefacts and fragments.

By specifying several references to many artefacts, annotations can be considered as a linking system for models. These connections can then be explained or completed with information by specifying the body of the annotation.

9.2.2. Prototypes using PANDA

As explained in the introduction, prototypes are a representation of the interactive system. Different representations can be created by designers depending on their needs and the aspect they want to evaluate. Prototypes can be used during evaluation activities with end-users but also as a specification for the implementation of the adopted solution.

For this framework, we will be using prototypes based on the PANDA prototyping tool which is a tool dedicated to the creation of medium-fidelity and interactive prototypes [119]. The PANDA prototypes are used to give a partial representation of the interactive system from the end-user point of view.

Those prototypes are made of two separated models: the presentation model which describe the look of the UI and the dialog model which define the interactivity layer of the interactive system.

The prototyping tool and the details on the models are presented in the Chapter 8. The integration of other prototypes model within the framework could be considered provided that a matching adapter for the targeting of fragments is developed.

9.2.3. Task models using HAMSTERS

Task models [113] are useful artefacts aimed at supporting the analysis of user tasks and the specification of the logical activities that have to be carried out in order to reach the user's goals.

HAMSTERS (which stands for Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems) is a tool dedicated to the creation of task models using the HAMSTERS notation [117].

HAMSTERS models are based on an oriented tree structure and can be used to define and decompose the different activities of the users and of the interactive system. This notation includes an extensive list of type of tasks and operators that can be used to accurately describe a task by defining what the user is doing, what the system is doing and what is the temporal relationship between those tasks. This notation also includes ways to specify the information manipulated during the use of the interactive system by allowing designers to specify where are the information coming from, when are they used etc.

9.2.4. Dialog models using ICO

Dialog models [114] play a major role in the design of interactive system by capturing the dynamic aspects of the user interaction with the system. These models include the specification of the relationship between presentation units (e.g. transitions between windows) as well as widgets (e.g. activate/deactivate buttons). The ICO formalism is used to formally describe the behavior of interactive system [118].

ICO models are based on high-level Petri nets and are used to model the behavior of event-driven interface. The places of the Petri nets are used to represent states variables that will affect the user interface and transitions are associated to actions and allow the change of state. Thus, the token system of Petri nets allows to define the dynamic behavior of the system. Indeed, transition from a given state are enabled if there is enough token since each transition consumes a given number of tokens.

Beyond this specification of the different states of an interactive system, the ICO formalism also features linking mechanisms to connect user events on a presentation to event handlers and state changes.

9.3. Case study

This section aims at demonstrating the use in practice of both the annotation model and the ARMADILLO tool supporting their management. For this case study, we will focus on the cross-referencing of models using annotations. Thus, this case study will be limited to a snapshot of an existing project at a given point in time. Thus, the evolution of artefacts following the production of annotations will not be featured in this case study.

9.3.1. Informal presentation of the case study

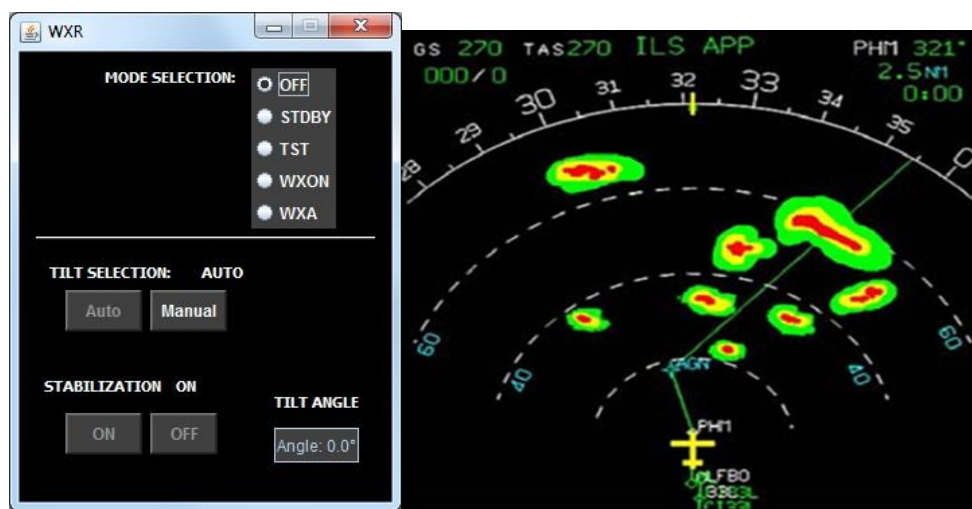
Weather radar (WXR) is an application currently deployed in many cockpits of commercial aircrafts. It provides support to pilots' activities by increasing their awareness of meteorological phenomena

during the flight journey, allowing them to determine if they may have to request a trajectory change, in order to avoid storms or precipitations for example.

Figure 9-1.a presents a screenshot of the weather radar control panel, used to operate the weather radar application. This panel provides two functionalities to the crew. The first one is dedicated to the mode selection of weather radar and provides information about status of the radar, in order to ensure that the weather radar can be set up correctly. The operation of changing from one mode to another can be performed in the upper part of the panel (named MODE SELECTION).

The second functionality, available in the lower part of the window, is dedicated to the adjustment of the weather radar orientation (Tilt angle). This can be done in an automatic way or manually (Auto/manual buttons). Additionally, a stabilization function aims to keep the radar beam stable even in case of turbulences.

Figure 9-1.b shows a screenshot of the weather radar display produced by the weather radar aircraft system. Spots in the middle of the images show the current position, importance and size of the clouds (importance is capture by color coding).



a) Weather radar control panel

b) Screenshot of the weather radar display

Figure 9-1 Image of the WXR application

9.3.2. The various artefacts of the WXR project

In this case study, we consider four artefacts which were produced during the duration of the project. Each of them gives a different representation of the interactive system:

- Task models using the HAMSTERS language [116] that describe the different tasks of the pilot (Figure 9-2) as well as their goals while using the Weather Radar Application
- System models using the ICO formalism [118] that formally describe the behavior of the interactive system (Figure 9-3)
- A medium-fidelity prototype designed with PANDA [119] which is a prototyping tool (Figure 9-4) dedicated to the early phases of interactive systems design
- A High-fidelity prototype of the panel implemented in Java (Figure 9-1a)

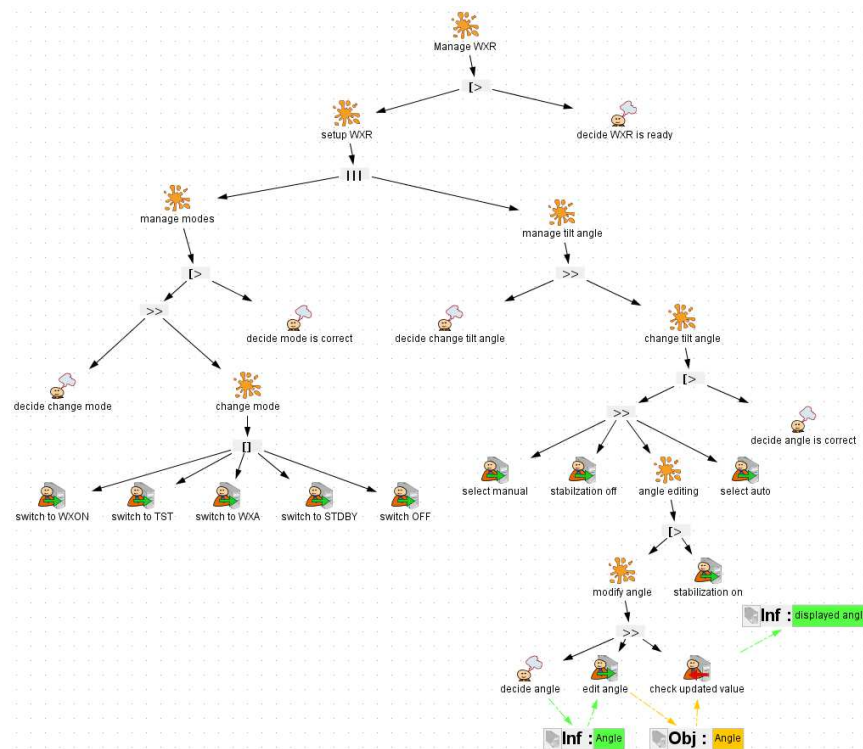


Figure 9-2 The task model artefact of the WXR project in HAMSTERS notation using the HAMSTERS task model editor and simulator

The task model in the Figure 9-2 represents crew activities performed in order to check weather conditions. At the higher level of the tree, there is an iterative activity (circular arrow symbol) to “detect weather targets” that is interrupted (operator [$>$]) by a cognitive task “mental model of current weather map is built”.

Other human tasks include perception (task “Perceive image”) and motor (task “Turn knob”). Connection between crew’s activities and cockpit functions is made through interactive tasks (as input “Turn knob” and output “Rendering of radar information”). The time required for performing the latter heavily depends on the radar type. Such behavioral aspects of systems can be modeled using ICO notation and PetShop tool as detailed in section below. The task “Manage WXR” is a subtask of a more global activity of flying crew understanding weather conditions ahead of the aircraft. This activity would include analyzing the image produced by the weather radar (shown in Figure 9-1.b)). This task model corresponds to the manipulation of the user interface presented in Figure 9-1.a). From these models we can see that the tasks to be performed in order to check weather conditions in a given direction are rather complex. The time required to perform them depends on 3 elements: the operator’s performance in terms of motor movements, perception and cognitive processing.

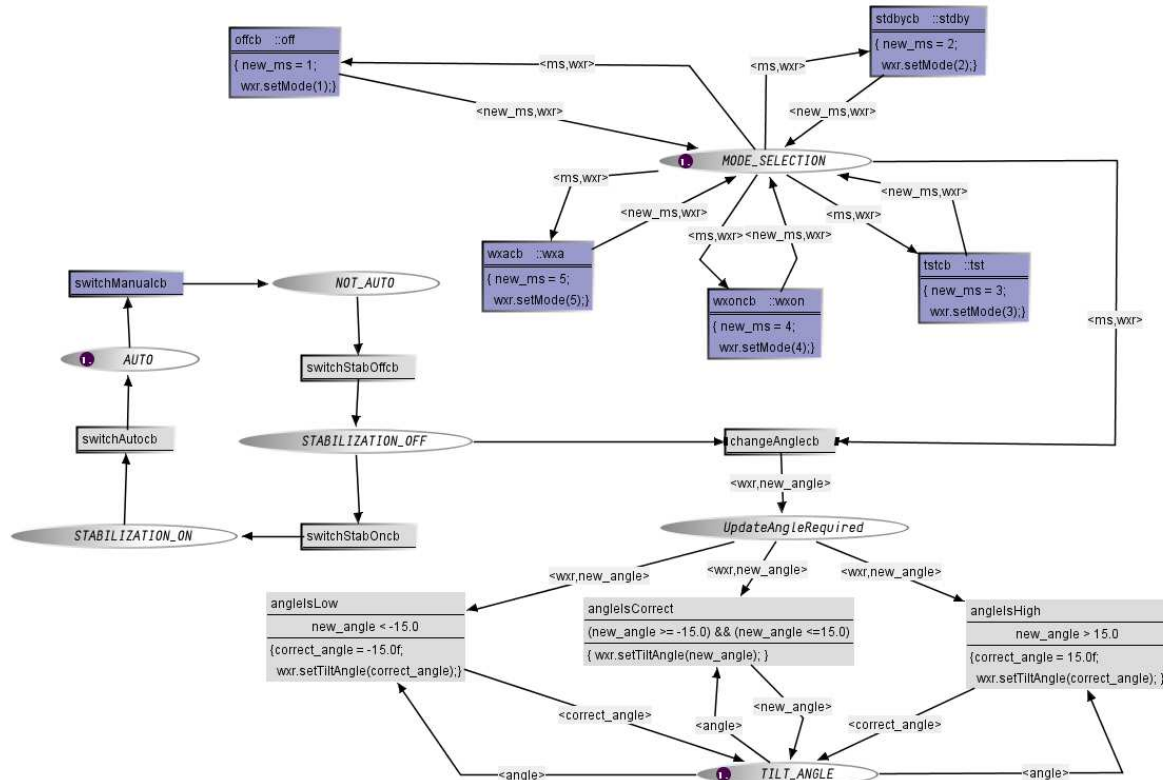


Figure 9-3 The system model artefact of the WXR project using the ICO formalism within PetShop Tool

The ICO model presented in the Figure 9-3 describes how it is possible to handle the weather radar configuration of both its mode and its tilt angle. The Figure 9-1.a shows the user interface provided to the user to:

- Switch between the five available modes (upper part of the figure) using radio buttons (the five modes being WXON to activate the weather radar detection, OFF to switch it off, TST to trigger a hardware checkup, STDBY to switch it on for test only and WXA to focus detection on alerts).
- Select the tilt angle control mode (lower part of the figure) amongst three modes (fully automatic, manual with automatic stabilization and manual selection of the tilt angle).

The Figure 9-3 presents the description of the behavior of this part of the interactive cockpit using the ICO formal description technique and may be divided into two parts.

- The Petri net in the upper part handles events received from the 5 radio buttons. The current selection (an integer value from 1 to 5) is carried by the token stored in MODE_SELECTION place and corresponds to one of the possible radio buttons (OFF, STDBY, TST, WXON, WXA). The token is modified by the transitions (new_ms = 3 for instance) using variables on the incoming and outgoing arcs as formal parameters of the transitions. Each time the mode value is changed, the equipment part (represented by the variable wxr within the token) is set up accordingly.
- The Petri net in the lower part handles events from the four buttons and the text field (modify tilt angle). Interacting with these buttons changes the state of the application. In the current state, this part of the application is in the state fully automatic (a token is in AUTO place). To reach the state where the text field is available for the angle modification, it is necessary to bring the token to the place STABILIZATION_OFF by successively fire the two transitions

switchManual_T1 and switchStabOff_T1 (by using the two buttons MANUAL and OFF represented in Figure 7), making transition change_Angle_T1 available. The selected angle must belong to the correct range (-15 to 15), controlled by the three transitions angleIsLow, angleIsCorrect and angleIsHigh. When checked, the wxr equipment tilt angle is modified, represented by the method called wxr.setTiltangle.



Figure 9-4 The prototype artefact of WXR project using PANDA tool

The Figure 9-4 presents an early prototype representing the UI of the WXR produced using the PANDA prototyping tool. This prototype was designed to evaluate one organization of widgets and information to be displayed in the panel. As the interactive application presented in Figure 9-1.a, this prototype is composed of one window called “WXR” and features several labels and buttons. The upper part of the prototype lists the different mode that can be selected: OFF, STDBY, TST, WXON, and WXA. The lower part is used to change the stabilization and the tilt of the radar as well as display the angle of the radar. It is important to note that, with respect to the application in Figure 9-1.a some user interface elements are not identified yet e.g. the radio button group on the upper part of the window.

9.3.3. Annotation of the prototype artefact of the case study

Figure 9-5 presents the how the PANDA tool enhanced with the plugin functionalities presented in the tool section. In that Figure, we created five annotations that give further information of the different modes of the WXR panel on the prototype and attached a reference to a PDF file of the Ergonomic Criteria for the Evaluation of Human-Computer Interfaces by Christian Bastien and Scapin [120].

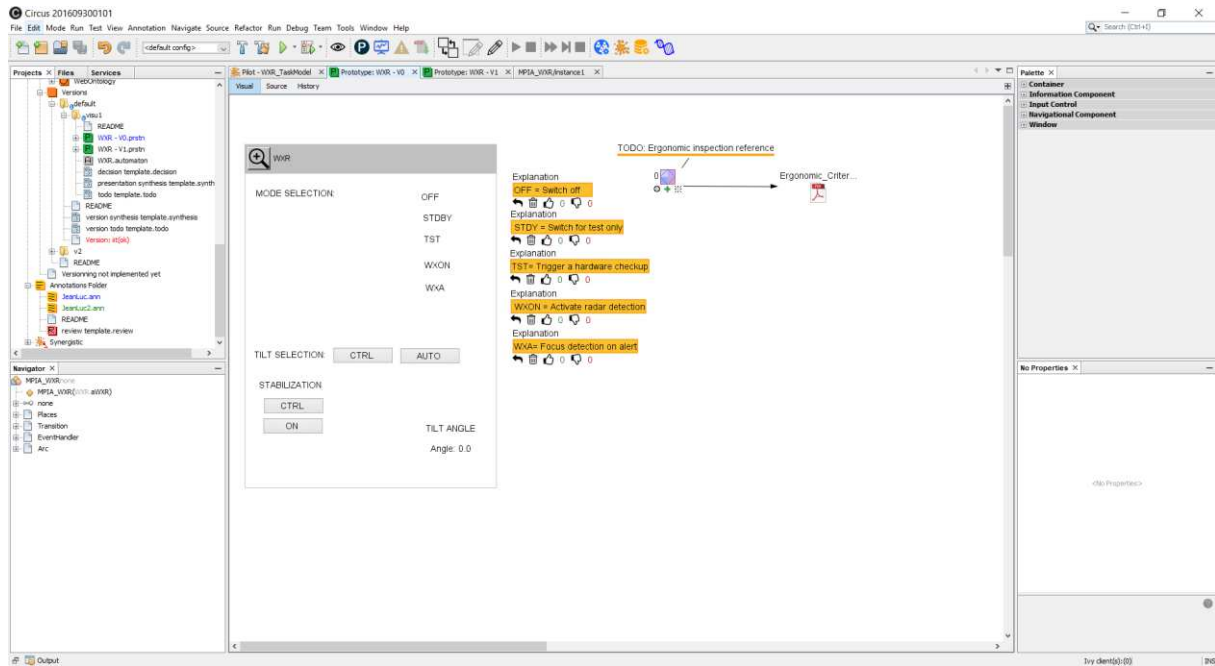


Figure 9-5 Annotations on the medium-fidelity prototype

The five annotations that describe each mode can be relevant in the other representations in which the abbreviation of the modes is used. However, the reference to the PDF document is only relevant to artefacts that represent the presentation of the interactive system. For the remaining of this section, we will only focus on the five annotations that explain the abbreviations.

At this moment, the annotations are only connected to the medium-fidelity prototypes. How annotations can be managed at WXR project level is presented in the next section.

9.3.4. Use of ARMADILLO tool to manage annotations at project level

Using ARMADILLO feature presented in the tool section, it is possible to associate those annotations (on the medium-fidelity prototype) to other artefacts. We present here how these annotations can be connected to the ICO model as they are also relevant for system behavior. This association means that each annotation is now targeting both the low-fidelity prototype and the ICO system model and this annotation will be displayed at the same location and will feature the same size in both representations.

Once we add the ICO system model to the target of all four others annotations, we can open the ICO system model using the PetShop editor and observe that the same annotations are displayed (Figure 9-6).

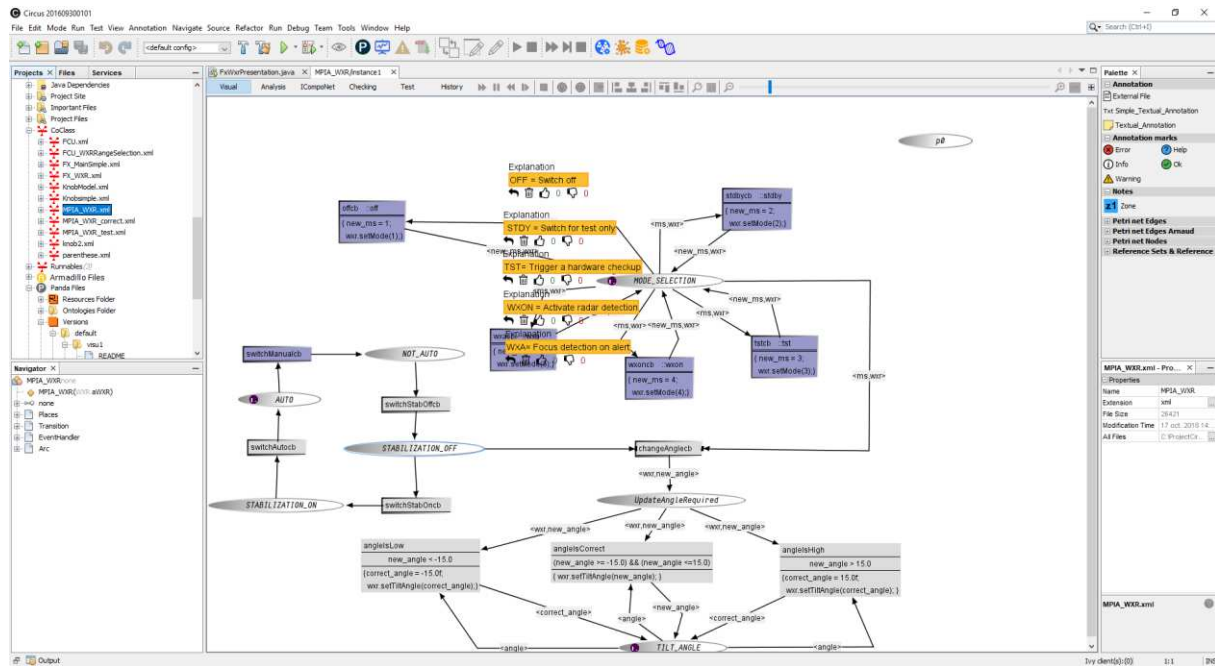


Figure 9-6 Annotations displayed in the PetShop editor using the same location properties as the low-fidelity prototype

At the moment, the annotations are displayed at the same coordinates as they used to be in the low-fidelity prototype since they are identical. However, it is clearly apparent that the different representations of the interactive system have their own layout and the annotations are unlikely to be located at the same place for each artefact. Thus, it is necessary to move the annotation to an appropriate location. Since we cannot use the selection of fragment for the annotations, we will rely on the location of the annotation to associate them with the relevant fragment of the model.

The location and the size of an annotation are properties that are specific to one target while the other properties like the content are shared. Thus, moving the annotations in this model will not affect their display on the low-fidelity prototype and vice-versa.

Once the annotations are targeting two different representations of the interactive system, we can open the ARMADILLO viewer to get a view of the annotations we created and their targets. The screenshot in the Figure 9-7 below is showing the annotation file we got after we moved the annotations in this view.

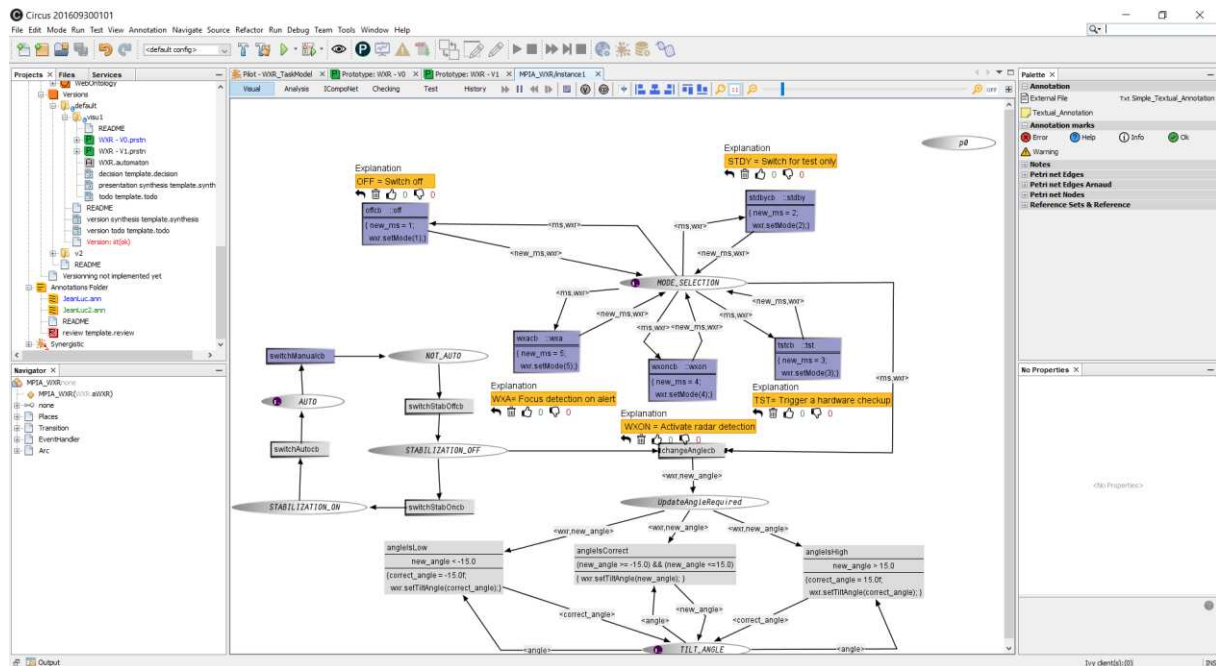


Figure 9-7 Annotations displayed in the PetShop editor after moving the annotations for the ICO model

In the Figure 9-8 below is displayed the five annotations contained in the annotation file. Each annotation is connected to the ICO system model file “MPIA_WXR.xml” and the PANDA prototyping file “WXR – V0.prstn”.

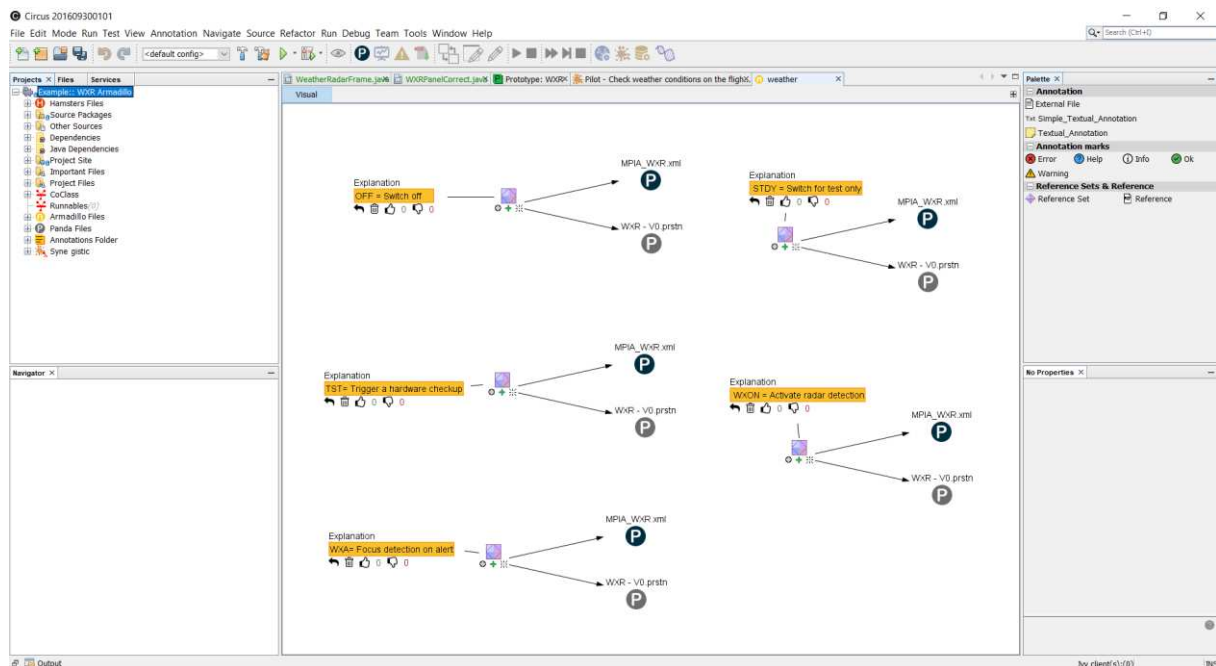


Figure 9-8 Overview of the annotations using ARMADILLO

9.4. Conclusions

This chapter presented a tool that allows in a systematic way to annotate design and development artefacts. This is done by a generic plugin that has been instantiated to every tool editor of artefacts. So far the plugin has been deployed in a few editors which are built on the same framework, namely NetBeans Platform and NetBeans Visual Library. Thus, the development for this case study has been facilitated due to the development over a single framework. Tools have different interaction workflow

when building artefacts and the ARMADILLO plugin allows this flexibility. For instance, the management of the palette of annotations is different from one tool to another: PANDA provide a different palette of annotations with the use of different modes for the editor (i.e. “prototyping mode”, “previewing mode”, “annotation mode”) while PetShop appends the different annotations within its main palette. Overall, the concepts for annotations and its flexible plugin is generic enough to be deployable in other tools and artefacts.

In order to support design and development activities, the plugin support the entire life cycle of annotations including their connection to the different artefacts produced by different stakeholders involved in the design process of interactive systems.

The use of the annotation models and of the ARMADILLO tool were shown in a case study on the aerospace domain involving four different artefacts. The case study demonstrates that it is possible to collect several information that might be useful for supporting design decision along the development process. These open several perspectives for investigating the use of information collected and design decisions along the process. In the future work, we will focus on traceability of the decisions and tools for supporting the visualization and rational design. Now that the concept is operational, we have the tools for a large case study including the collection of data describing the results of usability testing.

Chapter 10. Industrial transfer over eazly™

Summary

This chapter presents the industrialization of some of my contributions on the usage annotations within the design process. These contributions are applied and integrated in the design of a commercial tool called “eazly™” which allows its users to edit e-service applications after their deployments on production servers.

The first section will give a brief presentation of the context of eazly™ which is detailed in the Annex 8.

The second section will be dedicated to the presentation eazly™. The first part of this section will give an overview of the implication of the adding of “eazly™” in the e-Citiz™ framework through the presentation of the updated design process, the actors involved in this new version of the process and the presentation of the models manipulated with “eazly™”. The remaining parts of the third section will be dedicated to the presentation of the applications of the contributions of this PhD thesis in the “eazly™” project, namely the annotations, the versioning and the traceability of the design.

10.1. Introduction

This chapter presents the integration of the contributions of this PhD thesis about annotations within an industrial context. Thus, this chapter includes a presentation of the work done for the company “Softeam” in the “Softeam Software” business unit as a part of my CIFRE PhD scholarship. The Softeam Software business unit is providing a dematerialization of business process to governments and private companies. This service consists in producing, deploying and maintaining e-service applications. The production and the maintenance of these applications is ensured by the usage of the e-Citiz™ framework which is devoted to the modeling and the development of e-service based applications.

This work consisted in the development of the “eazly project” which is part of the evolution of the e-Citiz™ framework. As such, the tool developed in this project called “eazly™” is an extension of the “e-Citiz™” framework. In the “eazly project”, many of the concept developed in this PhD thesis has been applied: Annotations, Versioning, and Traceability of the design. Though the associated features are not fully integrated yet to date.

In order to better understand the core of our contributions materialized by the extension to the base framework with eazly™, we provide in the Annex 8 a view at glance of the e-Citiz™ framework, its environment, its limitations as well as the conclusions of the use of the e-Citiz™ framework. Overall, this annex hints towards the start of the “eazly project”.

The section 10.2 recaps the detailed presentation of the e-Citiz™ framework given in the Annex 8. The section 10.3 gives a presentation of the “eazly™ project” with a focus on the features developed from the industrial transfer of the contributions of this PhD thesis.

10.2. Overview of the e-Citiz™ framework

The e-Citiz™ framework is a proprietary framework for designing and generating e-service applications. For that, the framework include an editor called “e-Citiz Studio™” which allows its user to generate an application from different models. Among these models, we can note the “Business Process Model and Notation” (BPMN) which defines each services of the application as a sequence of tasks and the actors involved in each of these tasks as well as the Data model which describes data manipulated within the tasks.

Overall, the use of the e-Citiz Studio™ requires both analytical skills and knowledge of the business processes for modeling an efficient and effective e-Citiz™ application. Thus, the usage of this editor is limited to users called “Business Analyst” (BA) who will enquire about the business processes from the “Functional Expert” (FE).

The use of the e-Citiz™ framework revealed over time that while the application generated following the design process associated to the e-Citiz Studio™ is both suitable and satisfactory regarding the robustness and the suitability of the application, this framework lacks of flexibility for quickly updating and maintaining an application and lacks of accessibility for novice users.

These observations lead to two successive projects: the “Process 2.0” project and the “eazly™” project. The goal of these projects is to empower Functional Experts by allowing them to participate directly and become more active during the design of their applications. For that, a new editor will be provided to customize e-Citiz™ applications.

10.3. The “eazly™” project

As mentioned in the section 8.4.3 of the Annex 8, the eazly™ project is a continuation of the Process 2.0 project and consists in the design and implementation of an editor for overcoming the lack of flexibility of the e-Citiz™ design process after the deployment of the e-Citiz™ application.

The study performed during the “Process 2.0” project showed that the Functional Experts are not sufficiently familiar with modeling and the concepts of the BPMN for them to use the e-Citiz Studio™ (see 8.4.2 of the Annex 8). Thus, the project eazly™ will be offering the Functional Experts a lightweight editor that will be easy to use.

The perimeter of this project is as follow:

- The editor is designed toward the Functional Experts which include people not familiar with conceptual design and computer science but who are knowledgeable about their BPMN
- Editable e-service applications are limited to the one developed and provided using the e-Citiz Studio™ and therefore limited to e-Citiz™ applications
- The customization features are limited for now to the edition of the tasks defined in the Business Model represented in the e-Citiz™ application
- The customization features do not include the adjustment of the graphic charter of the application nor the editing of the CSS used by the application

10.3.1. Overview of eazly™

In this section we present the current results of the eazly™ project which consists in the integration of an editor called eazly Studio™ within the e-Citiz™ framework. The adding of this new studio has an impact on this framework on several aspects: the design process of an e-Citiz™ application, the models of the applications and the models of end-users’ data. A particular attention to the contribution of the PhD thesis will be given in this section, namely the **versioning** of the models, **traceability** of the evolution of the design and the integration of **annotation** within the design process.

10.3.1.1 The evolution of the design process

This section will first focus on the evolution of the design process implied by the adding of eazly™ to the e-Citiz™ framework. The design problems that arise with this new collaborative design process on the application as well as the implication of this collaborative design on the data manipulated by application will be examined in the section 10.3.4.

During the old design process with the e-Citiz™ framework, the main contributor for the design of the application was the Business Analyst who specified, developed, migrated and deployed the

application. The role of the Functional Expert was limited to give his requirements, the description of his Business Process at the beginning and to validate the prototypes produced by the Business Analyst.

In this design process, the modeling of the application and the migration procedure were considered too complicated and technical for the users as pointed by the study performed for the “Process 2.0” project (see 8.4.2 of the Annex 8). Thus, in order to empower Functional Expert for the design of their application, it has been decided to integrate their active contributions after a first iteration of the development cycle and the deployment process illustrated in the Figure Annex 8-2. At this state of the design process, a validated version of the application is deployed on a server (either a test server or a validation server). This version of the application benefits from the expertise of Business Analysts in the modeling of the models of the applications. Thus, this versions can be considered as a basis for Functional Experts who will be able to edit autonomously their application seamlessly without having to go through all the activities of the maintenance process.

The new version of the design process with the integration of eazly™ is illustrated in the Figure 10-1 and features the adding of a third phase in the design process. This part of the design process is performed by the Functional Expert. This third phase is colored in green and corresponds to the customization of the application with eazly™.

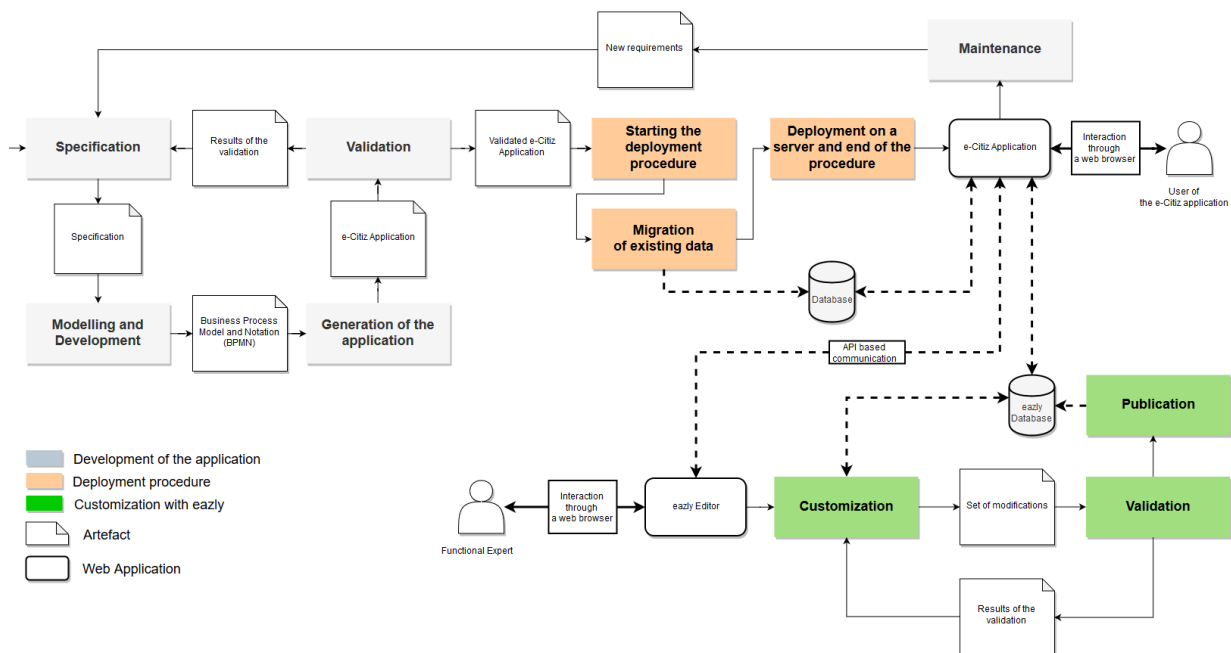


Figure 10-1 e-Citiz™ design process with eazly™

This part of the process can be done in parallel to the development by the Business Analyst as long as there is a version of the e-Citizen™ application available and reachable by the eazly Studio™. The details of the fusion of the modifications done will be discussed in the section 10.3.4.

The customization of the application with eazly™ can be divided into three activities: Customization, Validation and Publication.

- **The customization**

This activity is dedicated to the creation of a set of “modifications” by the Functional Expert called “Designer”. These modifications will be applied to the application, thus reflecting the customization by the Functional Expert. In the current version of the eazly Studio™, the modifications consist of adding

web elements to the different pages of the e-Citiz™ applications. Future iterations of this editor will include other customizations options such as the adding business rules.

Similarly to the “eazly™” prototype created for the “Process 2.0” project, this new studio starts with the management of Appeaz as shown in the Figure 10-2.

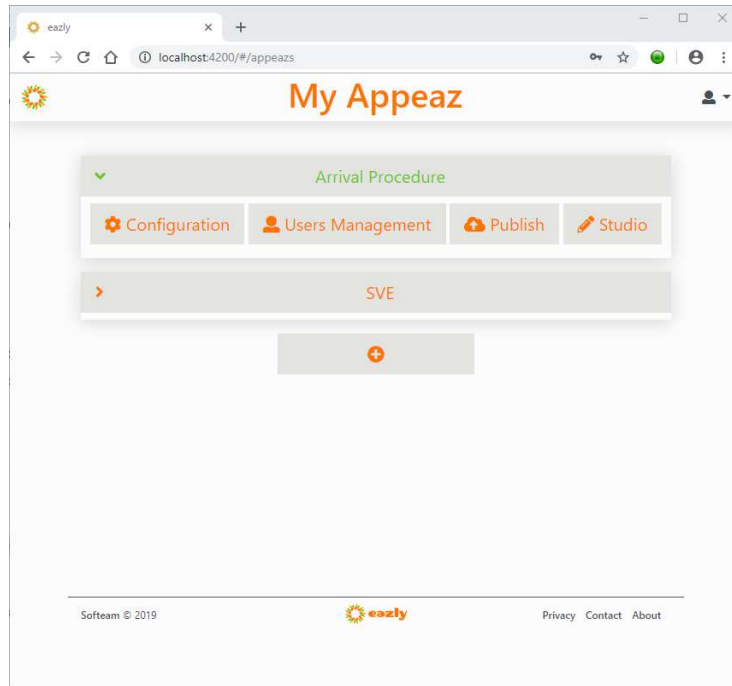


Figure 10-2 "My Appeaz" page

However, the management of Appeaz in this version of eazly™ differs from the previous prototype. Indeed, in this version, Appeaz are referring to existing e-Citiz™ applications that must be registered in eazly™ prior to proceeding with the modifications. To do so, the user can either configure an existing Appeaz by clicking on the “Configuration” button or register a new Appeaz by clicking on the “(+)” button. Doing so will open the configuration page of the Appeaz in which the user can specify the name of the Appeaz, the URL of the e-Citiz™ application and an API key to communicate securely with the application as illustrated in the Figure 10-3.

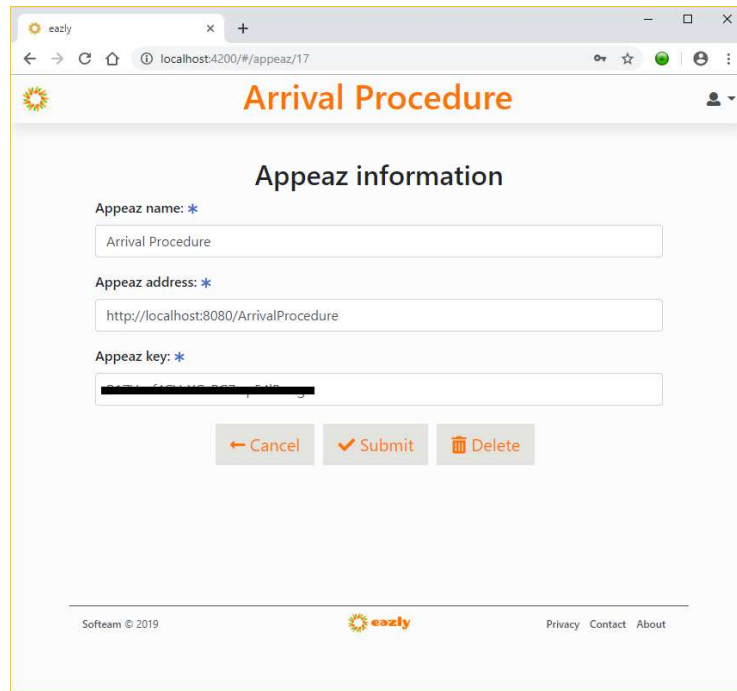


Figure 10-3 Appeaz configuration

Once the Appeaz has been configured, it can be accessed through the eazly Studio™ by clicking on the “Studio” button in the main page. This studio presented in the Figure 10-4 is divided into three sections: the navigator, the preview, and the palette. The navigator located on the left side of the studio and lists the different pages of the e-Citiz™ application that can be edited with eazly™. A preview of the selected page is displayed in a Wysiwyg editor at the center of the page. At the right side of the studio is located the palette of pre-made modifications that can be dragged and dropped on the preview for creating the modifications. Those pre-made modifications are represented in a grey rectangle to distinguish them from the base application and can be edited afterwards by the Functional Expert (i.e. labels, required attribute, values of a combo-box). For instance, the field “Age” in the Figure 10-4 is a modification made on the page “Declare my arrival”.

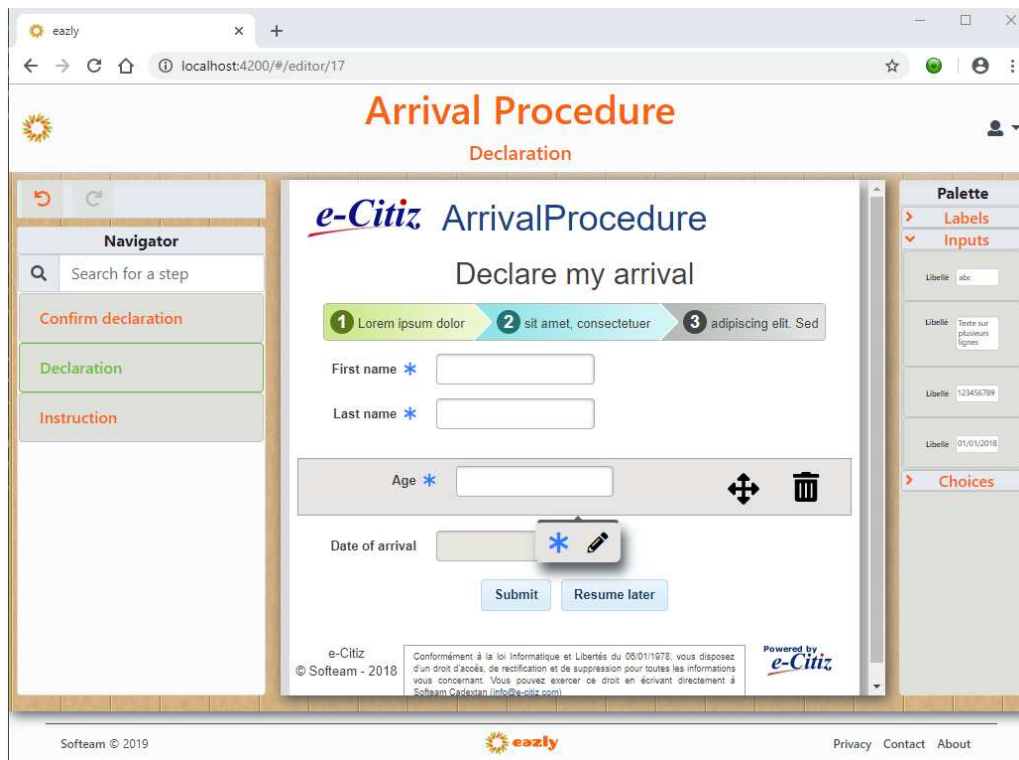


Figure 10-4 Screenshot of the eazly Studio™

The set of modifications created with the eazly Studio™ forms a Model of modifications. From this model, a structure of data is deduced seamlessly from the different fields added by the Functional Expert. This structure of data called “User Data Model” (more details in the section 10.3.2.2) will be populated by data input by end-users of the application in the different fields during their usages of the application. The type of the data is defined by the field added and is identified by the label of the field. For instance, adding a text input with the eazly Studio™ will define a data with the type “text” in the User Data Model while adding a date input will define a data with the type “date”.

The User Data Model can be considered as an extension of the existing data model defined in the e-Citiz Studio™. Nevertheless, the data entered by a user in fields of the extension are attached to the same instance of process of the BPMN as the data of the main structure of data. Thus, an adequate database query can display both the data from the main structure and the data from the extension.

The linking of fields through the different pages of the application are made by using the User Data Model. If two different fields share the same type and the same label, they will be pointing to the same data whose value is defined within an instance of process of the BPMN.

- **The validation of the modifications**

After the customization of the application, the modifications can be validated by any participant of the design process. This validation consists in checking that no error were made during the customization activity and that they match to the initial needs of customization. If the modifications are not validated, they must be edited until they satisfy the criteria used for the validation.

- **The publication of the modifications**

After the validation of the modifications, the modifications can be published to the application by a “Publisher” in the “Publishing” page shown in Figure 10-5. This page is only visible by the “Publisher” and displays various information on the status of the Appeaz such as the “Active version” which is

identified by the timestamp of the last modification made, the “Publish date” which indicates when the last publishing has been made and the version being edited.

The publication consists in activating the set of modifications that has been validated for application that is run by the server whether it is a test server or a production server. Once the modifications are activated, they are visible by any user who use the application.

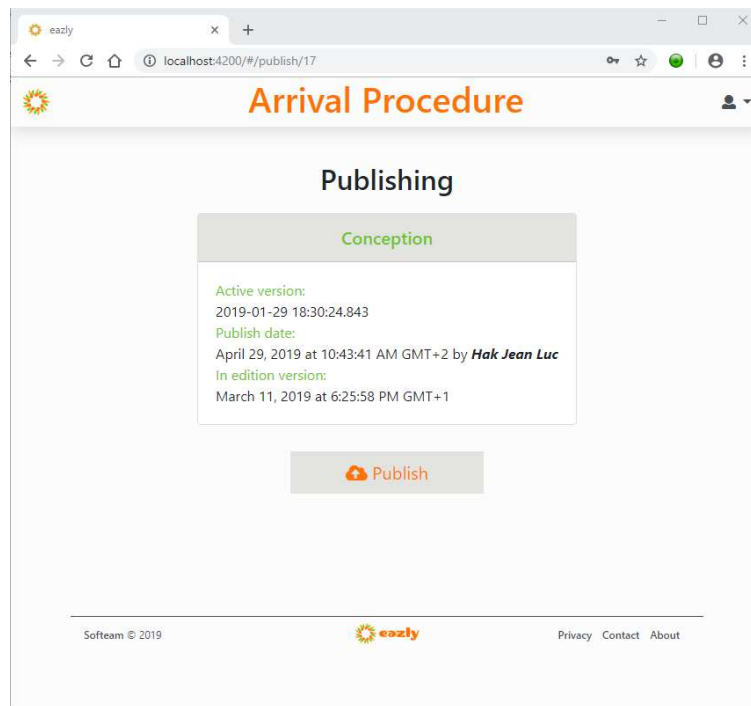


Figure 10-5 Publishing page

The publication process is done instantly, does not interrupt the service and is transparent to any user of the application. Thus, after each publication, the application combines with the new set of modifications can be considered as an evolution of the previous version. Indeed, when a user is using the application, the application displays a new and working UI with updated pages that interact with an updated model of the application.

The impact of this publication on the different models of the application as well as the management of the versions of the application will be discussed in the section 10.3.4.

10.3.2. Using of eazly™ as a prototyping tool

The eazly Studio™ has been designed to be able to produce viable updates of an application that can be deployed on a production server and can be maintained in the long term as well as to produce a complete application from an empty template without having to use the e-Citiz Studio.

However, the modifications that can be made with eazly Studio™ are limited compared to the modifications that can be achieved by coding and editing the BPMN or database queries with the e-Citiz Studio™.

While this design process is valid for designing an application, it is also possible to consider that the modifications done with the eazly Studio™ are applied to a prototype being currently developed iteratively and deployed on a test server. Thus, the modifications can be considered as a suggestion of modifications done by Functional Experts that need to be modeled and natively integrated in the BPMN with the e-Citiz Studio by the Business Analyst.

By using it this way, eazly™ is used as a prototyping tool for runnable and high-fidelity that can be used to discuss the evolution of the application.

10.3.2.1 Users of the eazly Studio™

The targeted users of the eazly Studio™ are the Functional Experts who are attached to the organization of the customer of the e-Citiz™ application. In the section 10.3.1.1, we identified two roles who intervenes at different steps of the design process: the “Designer” and the “Publisher”.

The “Designer” is responsible for editing the application by creating modifications on the application using the Wysiwyg editor.

The “Publisher” is responsible for publishing the set of modifications made by the Designers once they have been validated.

A third role can be added for the use of the eazly Studio™: the “Manager”. The role of the manager is to give access to the e-Citiz™ application through the eazly Studio™ by sending invitations to create an account if necessary and by assigning one role to the user for the application. Indeed, a customer can possess several e-Citiz™ applications and thus, he should be able to manage the different people who will be working on each application as illustrated in the Figure 10-6.

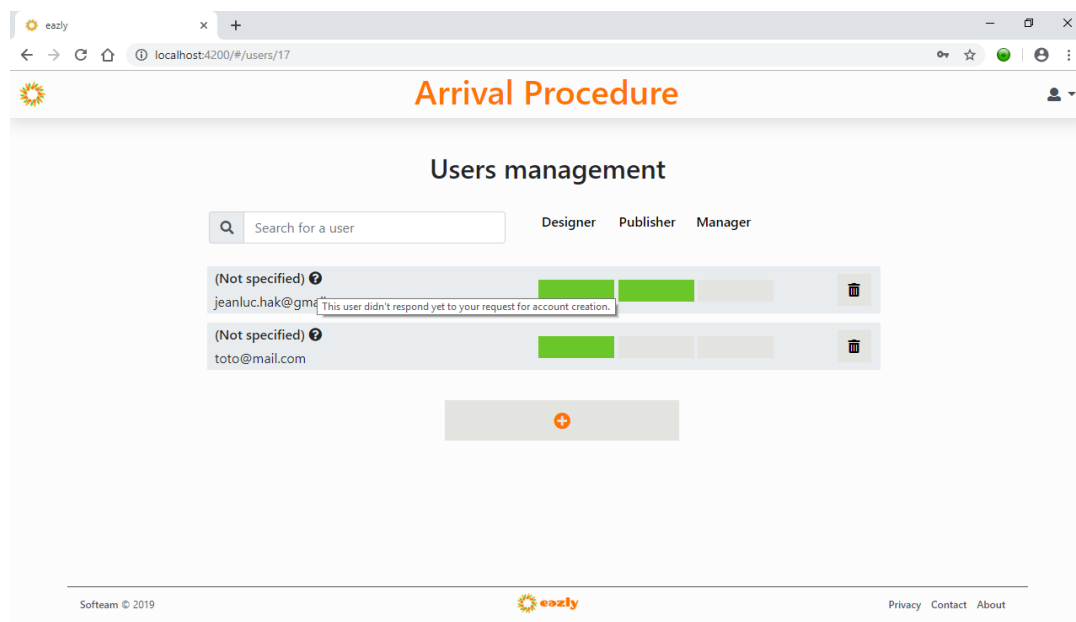


Figure 10-6 User management page

10.3.2.2 The different models of the eazly™ framework

Similarly to the e-Citiz™ framework, the eazly™ framework contains two different models: the “eazly™ model” and the “User Data Model”

- **The eazly™ model**

The eazly™ model is built automatically when the Designer creates his modifications. These modifications are attached to a reference to the different pages of the applications which were generated by the e-Citiz model. This imply that the eazly™ model owns a dependence to the e-Citiz™ model. Thus, the eazly™ model can be considered as an extension that adds itself on top of the e-Citiz™ model.

In the first version of eazly™, this model is only composed of modifications which correspond to web elements to display in each page. Each modification is attached to a set that can be published. The eazly™ model can contain several sets of modification. The publication consists in specifying in the application that there is a specific set of modification to use when navigating in the application. Thus, the update of the application through eazly™ can be done seamlessly without having to proceed with the deployment process of the application.

- **The User Data Model**

The User Data Model is built seamlessly as the Functional Expert designs his modifications in the eazly Studio™. This structure of data is typed and is based on the type of field dropped (e.g. a text input field will be associated to a “text” data while a number input field will be associated to a “number” data). This structure is persisted in the database along with the modifications to apply on the e-Citiz™ application.

The data of the fields added with eazly™ are handled by the e-Citiz™ application in the same ways as the data corresponding to the e-Citiz™ user data model: the data from eazly™ fields are also attached to the instance of the business process along the data from e-Citiz™ fields.

10.3.3. Annotation support for collaborating over the design

Annotations could be used in the eazly Studio™ as a part of the design process. Indeed, the design process used for the design of e-Citiz™ applications involves the collaboration of several actors (i.e. designers, publisher). As presented in the Chapter 3, annotations can be used for various purposes such as explaining the design, expressing an opinion, or simply communicate.

10.3.3.1 Motivations for adding annotations support in the eazly Studio™

As illustrated in the Figure 10-1, the design process of the creation and update of an e-Citiz™ application involves many actors with different skills in activities where many types of artefacts are used.

For this collaborative process, the actors must communicate. This communication can be achieved through the use of annotation. Indeed, as noted in review of the literature in the section 3.4.2, annotations are used for many purpose such as contributing to the elaboration of artefacts. For instance, a Functional Expert can create an annotation aimed at Business Analysts to specify modifications he would like to apply to the application that cannot be done using the eazly Studio™, thus requiring an update of the application through the e-Citiz Studio™.

Thus, annotations can be used for the validation and publication process added with the eazly Studio™. The section 10.3.3.2 will briefly present this use through the presentation of the prototypes of the eazly Studio™.

On top of that, the e-Citiz™ environment is including various models and artefacts that are tied, especially the web application which is generated from the BPMN models. Similarly to the case study presented in the Chapter 9, annotations could be used for gluing the different representations of the interactive system, thus favoring the communication of issues over the different artefacts.

For instance, any actors can create an annotation on the application and attach it to any fragment of the application. With the appropriate implementation, the annotation could then be automatically referenced over the different parts of the BPMN models.

10.3.3.2 Early prototypes of the annotation support in the eazly Studio™

- **Presentation of the annotations for commenting the design**

The first step in the integration of annotations in the eazly Studio™ consisted in the design of prototypes of their integration within the different pages of the eazly Studio™. This integration is dedicated towards the use of annotation for communicating about the e-Citiz™ application and the modifications created with eazly™. The Figure 10-7 illustrate an example featuring five annotations created on the Wysiyg preview. By selecting an annotation, the targets of the annotations are displayed with yellow arrows.

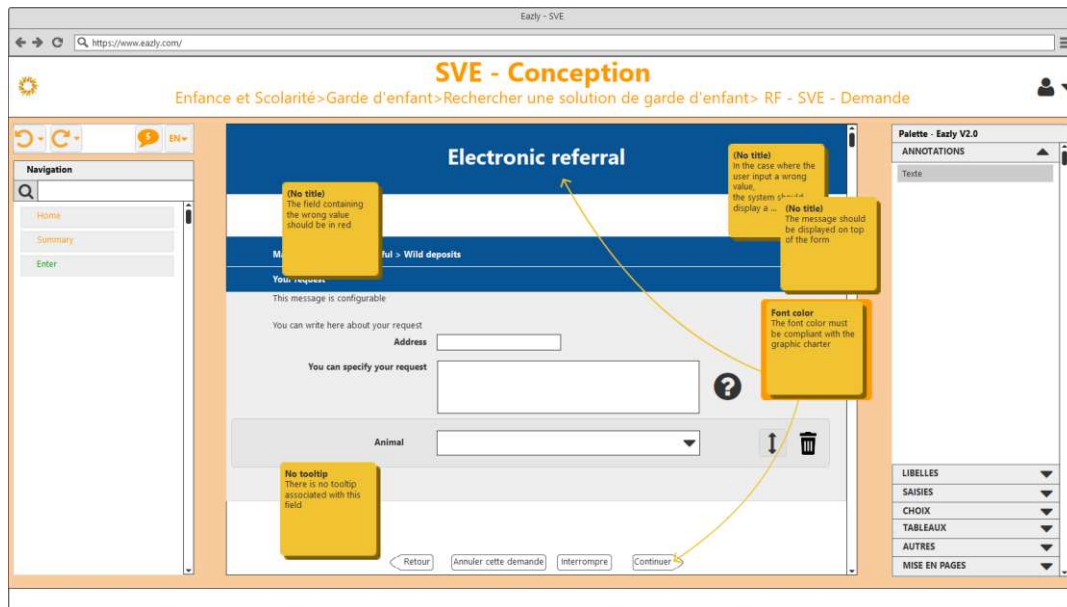


Figure 10-7 Display of the targets when selecting an annotation

In this early prototype, only the title and the content of the annotation are displayed. However, it is possible to display the details of the annotation as presented in the Figure 10-8.

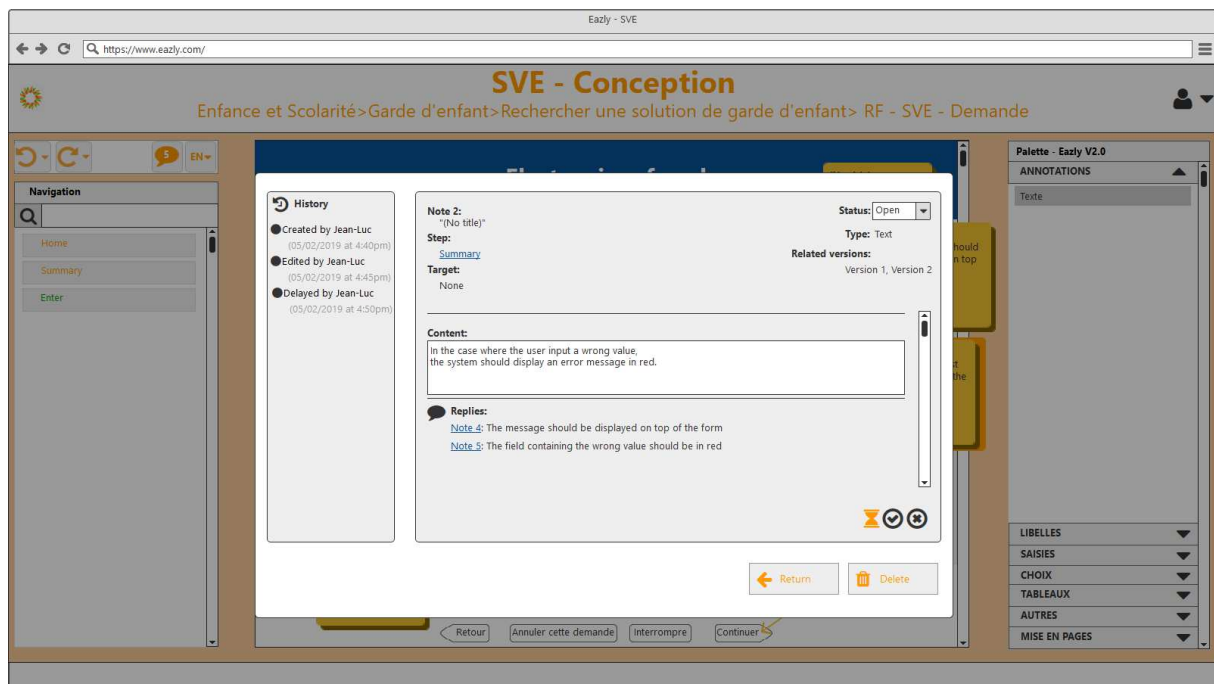


Figure 10-8 Details of an annotation

In this figure various information on the annotation is shown. On the left-hand side is displayed the history of events that occurred on the annotation (e.g. creation and edition date). The upper part of the window displays detailed information of the annotation such as the title, the pages in which it is attached to (e.g. “Summary” indicated using a hyperlink to redirect to this page), the targets of the annotation, the status of the annotation, its type, and the versions of the application in which the annotation is valid. The bottom part of the window displays the content of the annotation with the eventual replies to this annotation. These replies are described using hyperlink to allow the navigation through the different annotations. Icons located on the bottom right side of the panel can be used to make a decision on the annotation: to delay the processing of the annotation, to accept the decision or to reject it. These decisions are only used for ordering the annotations and to facilitate their management.

- **Overview of the annotations created over the design**

After the creation of the annotations over the application, a prototype of interface for managing the annotation has been created and illustrated in the Figure 10-9. This figure can be divided into three main parts. The first part is the versioning system located on the left-hand side of the prototype. This versioning system allows to display annotations that are related to a specific version of the modifications made with the eazly Studio™. For instance, this prototype show annotations that would be created in the first version of the modifications that have been published.

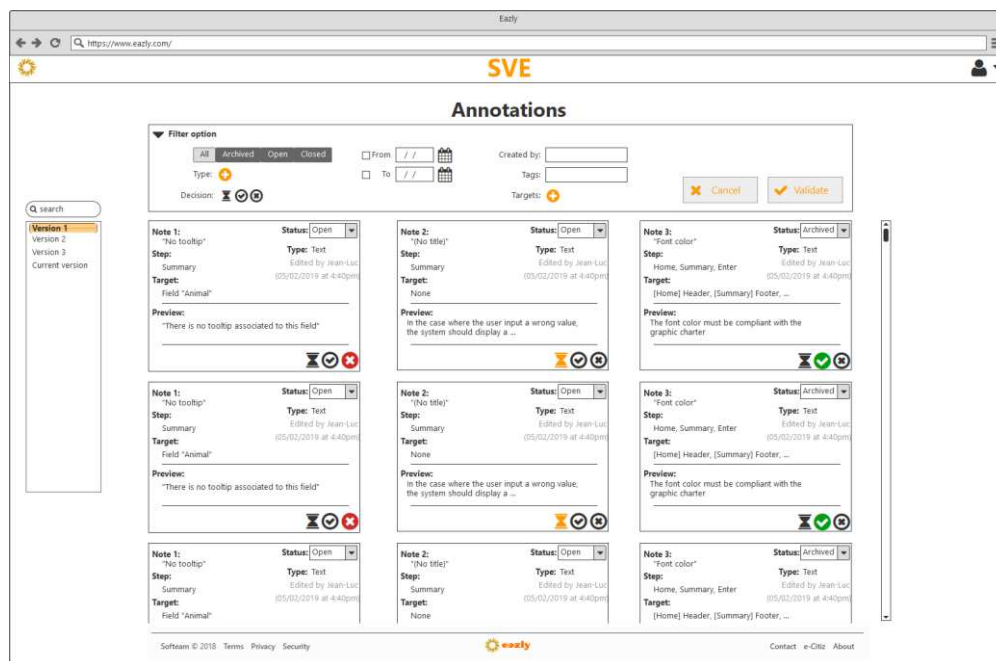


Figure 10-9 Prototype for the management of annotations created for each versions of the prototype

The second part of this prototype is the filtering system which allows to specify different parameters such as the status of the annotation, its type, the date of creation, and so on.

The third part of this prototype is the display of annotations that match the criteria specified in both the versioning and filtering system. Each annotation is represented by a box summarizing the information on the annotation. Similarly to the annotations displayed in the preview of the application, the details of the annotations can be displayed as shown in the Figure 10-10.



Figure 10-10 Details of the annotation in the annotation page

- **Integrating a decision-making process using annotations for the validation of the modifications**

The decision-making process explored in the Chapter 8 can be linked to the publication procedure of the modifications made with eazly™. Indeed, the publication of a set of modifications is done by a publisher who is responsible of the final validation of the modifications proposed with the eazly Studio™ by the Functional Experts. In the current implementation of the publication of modifications, there is no procedure defined for validating a version prior to its publication and there are no extra features associated to the publication. An extension to eazly™ could be considered for managing the decisions on the evolution of the application. The Figure 10-11 illustrate a prototype of the publication page showing the different servers in which the application has been deployed such as the Pre-Production and the Production.

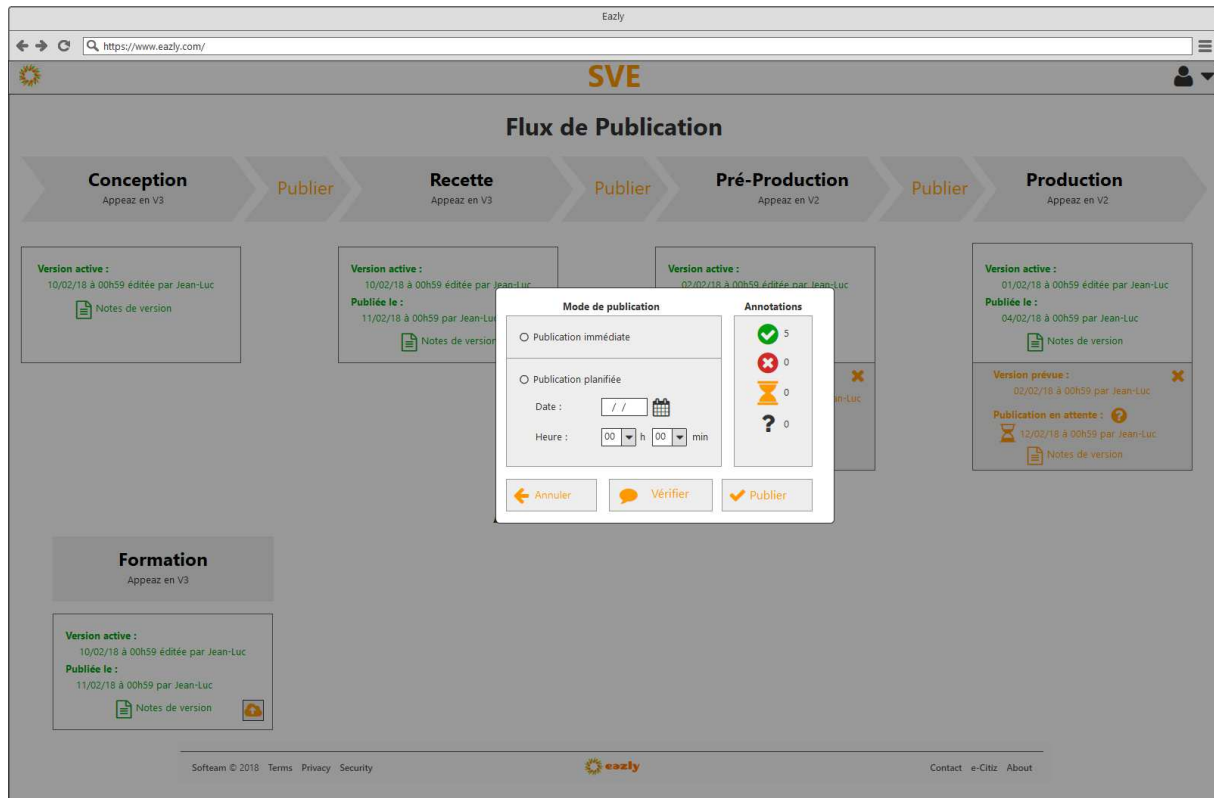


Figure 10-11 Integrating a recap of the annotations associated with the current version in the publication process

In this prototype, the modal window used for validating the publication features adding related to the annotations. On the right-hand side of the modal windows are listed the different annotations created for this versions and they are sorted by the decision made on each of these annotations. In this example, there are five annotations that were validated and no annotations that were rejected, delayed or lacking of decision. A button “Verify” has also been added to check the different annotations associated with this version of the modifications. This button will redirect the user to the page dedicated to the display of annotations as illustrated in the Figure 10-9.

10.3.3.3 Integration of the annotations in the eazly Studio™

In the first version of the integration of annotations in the eazly Studio™, the annotations can be used to create textual notes on the different pages of the preview as presented in the Figure 10-12.

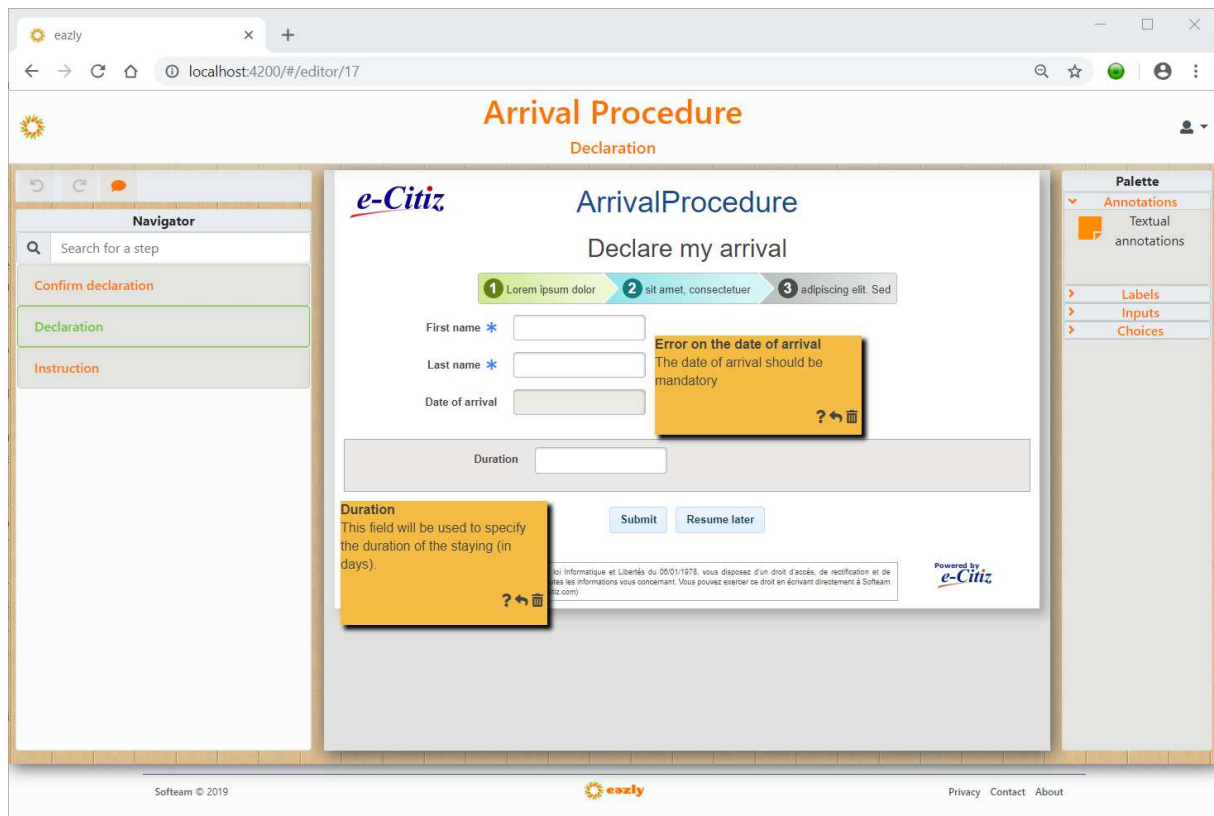


Figure 10-12 Integration of annotations within the editor

The annotations can be hidden using the button next to the undo/redo button as illustrated in the Figure 10-13.

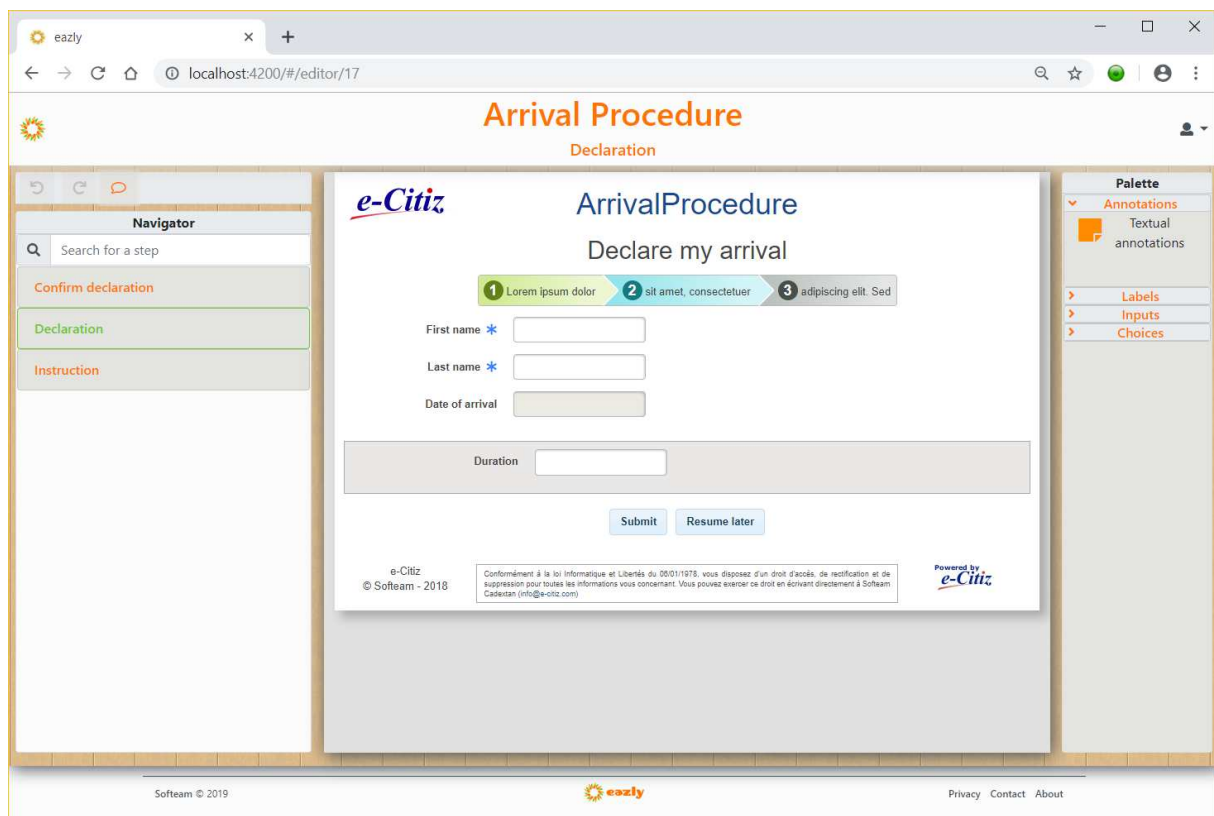


Figure 10-13 Integration of an "Annotation" button for displaying them

- **Architecture used for the annotations**

While a first prototype for creating annotations over the e-Citiz™ application in the eazly Studio™ has been implemented as a proof of concept, the final architecture of annotations has not been decided yet.

The prototypes presented in this section are only dedicated to the creation of annotations within the eazly Studio™. However, this studio is a part of a larger framework which includes other models and artefacts that could be annotated as well. First, the specifications used for the design of the application can be annotated. These specifications are then used for the modelling of the application in the e-Citiz Studio™ which represents another environment in which annotation support could be added. Then, the studio produces a web application that will be deployed and used by end-users. This application represents another design artefact that can be annotated. Finally, this application is used by the eazly Studio™ for creating and editing modifications as well as creating annotations on the representation of the application or the modifications created.

Overall, annotations can be created on the various artefacts produced and used at each step of the design process of the e-Citiz™ framework. Thus, the integration of annotations within this framework should be carefully planned since it involves different environments and different artefacts to annotate. In this industrial context, an analysis of the feasibility of such implementation should be done as well as an estimation of cost for the development of the architecture of annotations. These problems however represent more of a technical challenge requiring the implementation of various plugins for each environment rather than of a scientific one.

This aspect of this case study is very close to the concepts explored with the ARMADILLO tool presented in the Chapter 9. Thus, a similar architecture of annotations (presented in the Chapter 7) could be developed for integrating annotations in the e-Citiz™ and eazly™ framework.

10.3.4. Versioning of the evolutions to ensure the integrity of the application

10.3.4.1 Challenges imposed by the e-Citiz™ design process with eazly™ and its double iterations loop

The e-Citiz™ design process with eazly™ is a collaborative process in which contributors can iterate autonomously on their respective models. This section will expose the issues raised by the eazly™ project. Some of those problems are due to the fact that there is no interruption of service and no migration of data in the eazly™ design process which is a design choice made to simplify the tasks of the Functional Expert, to reassure the Functional Expert by hiding the complexity of the process of evolution and give a more immediate feedback through a seamless and stable evolution.

Those problems can be sorted into two categories: the model compatibility and the data consistency.

- **Model compatibility issues**

The model compatibility issues are due to the fact that the eazly™ model is built on top of artefacts generated from the e-Citiz™ model which are the references of the webpages of the application. Indeed, the eazly™ models specify modifications to display on the different pages of the application.

This dependency of models is giving the priority to the e-Citiz™ model since it is the basis of the application and the eazly™ model is an extension of this basis. Thus, the modifications made by the Business Analyst can provoke errors on the eazly™ model, but the modifications on the eazly™ model cannot disturb the e-Citiz™ model. Those errors can be due to the concurrency of the contributions (e.g. an input field defined both in the e-Citiz™ model and the eazly™ model for the same data which is redundant) or validity related (e.g. modifications were made with eazly™ on a page that is deleted by the Business Analyst).

Moreover, the update of the eazly™ model can occur when a user is being connected to the application. This update can provoke a constancy issue of the UI of the application and the data manipulated during the navigation which can disturb the user during his usage of the application.

Thus, the eazly™ model should be robust to minimize the impact of compatibility issues, it should detect those compatibility issues and the update of one of the two models needs to be validated to ensure the consistency of the application.

- **Data consistency issues**

The data consistency issues concern the instances of processes created by end-users of the e-Citiz™ application when the Functional Expert publish his modifications using the eazly Studio™. Indeed, the edits made by the Functional Expert on the pages of the e-service application does not interrupt the services provided by it and the instances of process of end-users are not migrated. Thus, a constancy issue can occur between in the instance of process of the end-user since the data structure and the pages are evolving.

The migration of the instances of process of the BPMN cannot be automated since it requires that the end-users enter their information for filling the new data.

On top of that, modifications of the BPMN can include the adding of new data to input, adding of new values that can be accepted, the mandatory aspect of a data, or the deletion of a data defined with eazly™ that is no longer required. Those data can then be used later in the BPMN by others actors or by the application itself to compute and process data (e.g. filtering, sorting, searching).

10.3.4.2 The proposed solutions to tackle the issues of the design choices made for eazly™

This section will give the adopted solutions for the issues raised by the design choices made for the integration of eazly™ in the e-Citiz™ design process.

- **An integration server for the validation of the application for the model compatibility issues**

The update of the e-Citiz™ model can induce a compatibility issue with the eazly™ model. However, the eazly™ model is an extension of the e-Citiz™ model and the application is validated before its deployment. Thus, the base application without the extension brought with eazly™ is guaranteed to work and the remaining task to complete the update is to validate the eazly™ model with the new e-Citiz™ model.

This validation can be done on an integration server with both the Business Analyst and the Functional Expert during the validation step of the e-Citiz™ design process. Once the eazly™ model has been validated with the new e-Citiz™ model, both models can be transferred simultaneously on the production server during the interruption of service in the e-Citiz design process.

In order to help the validation of the compatibility of the eazly™ model with the new e-Citiz™ model, a model analyzer is being planned. This analyzer is supposed to check every modification specified in the eazly™ model on the pages that were edited in the e-Citiz Studio™. This analyzer would then produce a report that list and sort the modifications that should be checked and validated depending on the nature of the evolution of the e-Citiz™ model. For instance, if a webpage has been deleted, the attached modifications are marked as orphan modifications.

- **Versioning of the instances of BPMN for both the UI constancy and the data consistency issues**

In order to ensure the constancy of the UI of the webpages displayed to the end-user in the different steps of his process of the BPMN, a versioning system of the eazly™ model has been developed. The goal of this versioning system is to guarantee the constancy of the UI when an end-user is navigating and thus to guarantee the constancy of the data persisted and presented to the user. This is done by registering the version of the eazly™ model used when an end-user starts a process of the BPMN and by always loading the same eazly™ models for every activity of the BPMN. This feature requires that all versions published by the Functional Expert are accessible by the e-Citiz™ application.

This versioning model is automatically handled by the eazly Studio™ and a new version is created when the Functional Expert publish his modifications on the application. This new version of the eazly™ model is used for every new instance of the BPMN created after the publication to benefit the last updates for the end-users.

As for the operations on the data of the end-users, the validation on the integration server and the constancy of the eazly™ model used throughout the activities of the process guarantee that there is no operation that requires a data that will not be available at the computing time.

10.3.5. Traceability of the evolutions of the design through the iterations

One of the particularities of the integration of eazly™ in the e-Citiz™ environment is that each versions produced with eazly™ is stored and used by users of the e-Citiz™ application as explained in the section 10.3.4.2.

Thus, the traceability of the evolution of the design of an e-Citiz™ application can be achieved to some extent since the information about a version such as the release note and publication information are available and maintained. However, there is still a lack in the traceability of the design of each version which might be inherent from the design process. Indeed, each published version are only representing the outcome of an iteration in the design process.

The information related to the reasons that motivated the evolution of the application, the design choices explored, the decisions made during the iterations are not stored nor specified during the design process.

The support of a diary integrating a decision system marking the different milestones in the course of the design process could solve this gap. The role of this diary would be to register the required information gradually along the activities of the design process. Moreover, as suggested in the Chapter 8, annotations could be used in this decision system for storing information on the evolution of the design.

10.4. Conclusions

This chapter presented a case study of the design and the maintenance of an application in an industrial context. This case study featured both technical and scientific challenges related to this PhD thesis and gave a concrete example of issues to be handled for the evolution of an application (e.g. management of user data models, merging of models, consistency of the UI from the end-user point of view) after the production started. This case study lead to the design of an editor called eazly Studio™. While this tool is still being developed and enhanced, it has been presented in a demo at the HCSE 2018³ conference at SophiaTech Campus of the University of Nice Sophia Antipolis in France as well as at the CoTer 2018⁴ national congress at Tours in France. On top of that, the eazly Studio™ has

³ <https://hcse-conference.com/programme/>

⁴ <http://coter-numerique.org/congres-2018/>

been commercialized with a first sale in February 2019 with the city Saint-Etienne-du-Rouvray which demonstrates the interest given to the work produced for this project.

Overall, this case study featured several aspects where the ideas explored during this PhD thesis could be implemented.

- **Annotations over the design**

Annotations have been added to the eazly™ framework for supporting communications over the design and for the publication system which can be compared to a decision system. These annotations are based on a similar structure to the one presented in the Chapter 6 with the use of references for defining multiple targets of different type and a body that can be customized.

However, many ideas explored during this PhD has not been featured yet in this version of the eazly Studio™ such as the management of annotations. Thus, the current implementation of the annotations is a simplified version of the contributions presented in this PhD thesis.

This simplification is due to the fact that the eazly™ environment had first to be built while ensuring its stability and its reliability for a commercial use. On top of that, the e-Citiz™ and the eazly™ environment are different from the CIRCUS environment, thus implying the development of a new architecture for displaying annotations over web interfaces.

On a side note, it is interesting to notice the similarity between a modification made with the eazly Studio™ and the annotations. Indeed, the modifications can be considered as a particular annotation since they feature a body (i.e. the definition of the modification) and they are attached to a target (i.e. the reference to their position in the application). The evolution of the application can have an impact on the modifications, as explained in the section 10.3.4.2.

- **Connecting models using annotations**

While the eazly™ and the e-Citiz™ environments share a lot of similarities to the previous case study (i.e. a variety of design artefacts used for a model-based design), the integration of annotations for connecting the models as showed in the “WXR” case study (see Chapter 9) has only been briefly analyzed.

We noted that the eazly™ model, the data model, the BPMN model in e-Citiz™ and the generated application could be linked together. Indeed, the eazly™ model and the BPMN model are complementary and thus connections can be made such as the integration of annotations in the e-Citiz Studio™ for featuring a representation of the modifications made with the eazly Studio™. Another example of the connection of models using annotations is the creation of annotations on the e-Citiz™ application that would then be connected to the other models.

These example of integration of annotations are raising a technical challenge rather than a scientific one. Indeed, in order to support this integration of annotations on these different platform (i.e. over the web and on the two different studios), the annotation system must be common and each platform should implement a plugin for supporting the annotation system similarly to the ARMADILLO annotation system (see Chapter 7).

- **Evolution of the design process for integrating a validation process that can use annotations**

The adding of eazly™ within the e-Citiz™ environment has impacted the design process for the creation of e-Citiz™ applications as explained in the section 10.3.1.1. In this design process, a loop has been added for the creation and validation of the modifications prior to their publication on the server.

This loop can be assimilated to a decision process which consists in evaluating the modifications and choosing if they are acceptable for the publication.

While the process of the validation and publication has not been fully specified for the eazly Studio™, the need to communicate over the design between the publishers and the designers of the eazly Studio™ have been identified. This need could be covered by the use of annotations which then could be included in the publication process similarly to the inclusion of annotations in the decision system of the PANDA ecosystem as presented in the Chapter 8 (i.e. using annotations to justify a decision).

- **Versioning of the design**

During the PhD thesis, the versioning of the design has been considered as a way to retrieve the context and the states of the artefacts in which an annotation has been created to better understand the content of the annotation. In other words, the consistency of the information contained in both the annotation and its targets is ensured by the synchronization of these artefacts.

This case study showed a similar issue due to the evolution of the application with the extensions brought with eazly™ and the instantiation of BPMN processes. The evolution of the application may induce a desynchronization with the instance of the BPMN process. This desynchronization can affect the use of the application when end-users are processing this instance.

Thus, we proposed to adapt the versioning system that was briefly defined in the section 7.3 which consists in storing several versions of the models to make them accessible anytime. Thus, when using an instance of a BPMN process, the application can retrieve the right version of the application, thus reconstituting the initial context in which the application was used for this instance without impacting other instances.

- **Traceability of the evolution of the design**

Regarding the study of the traceability of the evolution of the design, this can be achieved through the use of annotations to connect the representations of the interactive system to artefacts representing the decisions to as suggested in the Chapter 8.

Indeed, a further integration of annotations in the e-Citiz™ and the eazly™ environment would be to provide annotation tools to end-users who would be able to leave their feedback and attach them directly on the design of the application. These annotations could then be processed in the decision system for justifying or motivating a design choice to be made. These choices would then be stored with a track of the references used for the decision as a milestone of the evolution of the design. Then these milestones would form a history of the evolution of the application.

Overall, the application of the different contributions of this PhD thesis has been either partially implemented or studied but not fully implemented yet. Indeed, some relevant parts of contributions of this PhD thesis have already been adapted such as the versioning of the application. As for the annotations and the decision system, we briefly presented how they could fit in the e-Citiz™ design process. Thus, the concrete contributions of this PhD thesis toward the e-Citiz™ and eazly™ environment has been identified as well as their potential uses for the design process.

Chapter 11. Conclusions and Future Work

The goal of this PhD thesis was to study the annotations and their role during the design process of interactive systems due to their uses in various artefacts such as paper prototypes. With respect with the questions presented in the Chapter 1, we performed various studies and proposed contributions related to annotations in the context of the design of interactive system.

Indeed, we have identified the formalism of annotations about their structure and their many uses through the extensive study of the literature and their tool supports as presented in the Chapter 3 and Chapter 4. We noted that the implementation of annotations in the tools are limited on a few aspects such as the targeting of external resources. Regarding their support in prototyping tools, the promoted uses of annotations were their uses for collaborating and for gathering of feedbacks over the design during the evaluation activity which is a limited part of the many uses identified in the literature.

We noted that a standard for annotations over web resources has been recently proposed by the W3C and this model has already been adopted by a few tools. However, this annotation model is not suited for annotating every type of artefacts produced and used during a design process.

This analysis of annotations and their support motivated our following contributions.

Indeed, we formalized in the Chapter 5 the interaction between an annotation and its targets. We also identified potential uses for annotations along the activities of design process while noting the specificities of the consistency issue of information presented in both the annotation and the artefact when an evolution occurs during the design process or on one of the artefact. Indeed, annotations and artefacts are influencing each other through the decision activity: this activity can affect either design artefacts or annotations and can be motivated or justified by information contained either in the artefacts or the annotations. These maintenances reflect the necessity of maintaining a consistent representation of the interactive system over the different artefacts and the different annotations.

Following that, we proposed a new model-based approach for annotations in the design process as presented in the Chapter 6. In the annotation model, the flexibility and accuracy of the targeting of the annotations as well as the support for specifying different structures of body have been ensured for making them useful during the activities of the design process of interactive systems. Indeed, annotations can be used as a communication medium that can be used for various purposes as demonstrated by the literature. The support of different structures of body combined with the targeting system allow to specify annotations that can contain various information in relationship with its context. Thus, annotations can be considered as a versatile tool that can be used at different phases of the development process.

After that, we illustrated the uses of these contributions through three different cases studies in the Chapter 8, the Chapter 9 and the Chapter 10, each showcasing a specific aspect of the contributions. Indeed, these case studies were used for demonstrating several applications of annotations within different context that can contribute to the activities of the design process such as an automated test system, or the gluing of models. On top of these uses of annotations, we also identified other ways to exploit annotations such as their usage for traceability purpose and for assisting the designers during a decision process even though we did not yet implemented those features in eazlyTM.

In the current state of the contributions, we can still note several limitations regarding the support of annotations or its integration and exploitation within a design process. Several concepts were set aside to limit the complexity of the implementations of annotations such as the specification of a recipient for an annotation or the study of freeform annotations. However, these concepts do not question the

core concepts of the contributions of this PhD thesis which are the importance of annotations within a design process and the features to consider when implementing an annotation system for multiple type of artefacts. Moreover, these topics can be covered in future studies.

Regarding the integration of annotation and their processing during a design process, we can note several aspects that should be considered for future works. First, annotations can be created by anyone using any representations. As such, annotations made during a design process can contain implicit information that requires an analysis, they could be informal, irrelevant, or even contradictory to other sources (i.e. other annotations or artefacts) making them more difficult to process. Thus, an activity dedicated to the analysis, the filtering and the formalization of the information carried by annotations could be added before using them during the decision-making process.

However, by adding more and more activities related to the management of annotations, we can clearly identify an upcoming problem related to the overload of work with the adding of annotations. As such, an interesting study would be to identify if the investment over annotations would be worth for the development of interactive systems.

Overall, we can consider that our initial goals for our works has been achieved to a certain extent with the contributions presented in this PhD: the use of the targeting mechanisms inherent to annotations for connecting the design artefacts allowed us to structure the artefact of the design process. By materializing the decisions with artefacts such as annotations, our proposed model and process allow to connect ideas and design decisions to the artefacts.

Thus, the contributions of this PhD thesis are representing the foundation of future works on the usage of annotations in the context of the design of interactive systems. Indeed, we have defined several goals ranging from short-term goals to long-term goals.

We identified two short-term goals related to the current state of the work already performed:

- First, many contributions presented in this PhD are being edited for a publication in various conferences. The goal of publishing these contributions is to disseminate the knowledge provided by this PhD thesis.
- Secondly, the current implementation on the different tools presented in the case studies are not yet stabilized, thus requiring further developments for fixing bugs and finishing the implementation of different features.

Beyond the submission of the contributions of this PhD to conferences and workshops, we intend to pursue the work on different topics that will be based on the annotation model and architecture we presented:

- The first medium-term goal for the future works consists in a study on the decision making processes and the potential use of annotations for supporting this process. This goal would lead to an extension of the initial studies presented in the section 5.2.3. For this extension, we would investigate the evaluation of the annotations with their relevance and popularity, the extraction of decisions or design choices to make from the annotations as well as any other exterior sources that can impact the evolution of the design through the decision process.

- The second medium-term goal is to start working on an analysis and a study related to the traceability of the evolution of the design. Indeed, this topic is one of the goal we want to achieve through the use of annotations. One of the solution we propose to ensure the traceability of the design is to build a timeline around prototypes created during the design process. This timeline would feature the external influences that have an impact on the prototype which could be materialized by the decisions made during the design process. These decisions would then be documented with annotations that can list the origin of the decision and the resources used to make the decision.
- The third medium-term goal of this PhD thesis is to use the traceability of the evolution of the design for studying the traceability of the communication made by the actors of the design process and examine how those communication are followed up in the design process.
- The fourth medium-term goal is to pursue the work started on the management of annotations. Indeed, managing manually the consistency between an annotation and its target might be burdensome.
- The last medium-term goal is to perform experiments on field to study the annotation system we developed. These studies would be targeted toward the use of annotations during several iterations of a design process. Thus, we expect to find out if annotations could be a reliable and usable tool for structuring the artefacts of a project workspace, for assisting stakeholders during the decision process and for ensuring the traceability of the evolution of the design.

After the completion of these medium-term goal, we identified three different long-term goals:

- The first goal consists in extending the annotation system to tangible artefacts. Indeed, the design of interactive systems are not limited to digital artefacts and to the design of the UI and the behavior of the system. Tangible artefacts can be used and annotated early in the design process such as paper prototypes or whiteboards. On top of that, interactive systems may include the use of specific input devices or output devices that can be prototyped as well. To tackle this issue, several solutions can be imagined: the use of augmented reality to create annotations on tangible artefacts combined with the use of an image recognition system to identify the tangible objects, the use of a tool support for specifying a semantic to areas of a photo of the artefact combined with the annotation of these photos. Another approach for dealing with tangible artefacts is the study of transfer of artefacts from physical to digital and vice versa. This transfer could be done using an AI recognition system for identifying and generating digital annotations from a physical representation.
- The second goal is the study and the design of a platform for managing projects which would include a set of artefacts and of annotations. This platform would be used to gather the different contributions related to this PhD thesis and give a view of the artefacts complementary to the view provided by ARMADILLO.
- The third long-term goal is to engineer the traceability of previous projects in order to provide a recommendation system for the development of interactive system. Indeed, by gathering data on the decisions made on various projects, we assume that we can find patterns of design choices that could be proposed for future projects. These patterns could also be analyzed to identify their characteristics, their adoptions rate, their revisions, and the design choices often used together

List of publications

1. Thiago Silva, Jean-Luc Hak, Marco Winckler. A Review of Milestones in the History of GUI Prototyping Tools. Workshop on User Experience and User-Centered Development Processes 2015 (IFIP WG 13.2). INTERACT 2015 Adjunct Proceedings vol:22. p:267-279. University of Bamberg Press.
2. Silva, Thiago & Hak, Jean-Luc & Winckler, Marco. (2016). Testing Prototypes and Final User Interfaces Through an Ontological Perspective for Behavior-Driven Development. 9856. 86-107. 10.1007/978-3-319-44902-9_7.
3. Jean-Luc Hak, Marco Antonio Winckler, David Navarre. PANDA: prototyping using annotation and decision analysis. 8th ACM SIGCHI conference Engineering Interactive Computing Systems (EICS2016), Jun 2016, Brussels, Belgium. EICS '16: Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 171-176, 2016. <hal-01712526>
4. Rocha Silva, Thiago and Hak, Jean-Luc and Winckler, Marco Antonio. An Approach for Multi-Artifact Testing Through an Ontological Perspective for Behavior-Driven Development. (2016) Complex Systems Informatics and Modeling Quarterly, 7. 81-107. ISSN 2255-9922
5. T. R. Silva, J. Hak and M. Winckler, "A Behavior-Based Ontology for Supporting Automated Assessment of Interactive Systems," 2017 IEEE 11th International Conference on Semantic Computing (ICSC), San Diego, CA, 2017, pp. 250-257. doi: 10.1109/ICSC.2017.73
6. Silva, Thiago & Hak, Jean-Luc & Winckler, Marco. (2017). A Formal Ontology for Describing Interactive Behaviors and Supporting Automated Testing on User Interfaces. International Journal of Semantic Computing. 11. 513-539. 10.1142/S1793351X17400219.
7. Silva, T. R., Hak, J.-L., Winckler, M. & Nicolas, O. (2017). A Comparative Study of Milestones for Featuring GUI Prototyping Tools. Journal of Software Engineering and Applications, 10 (06), pp. 564-589. DOI: <http://doi.org/10.4236/jsea.2017.106031>. (Silva et al., 2017)
8. Jean-Luc Hak, Olivier Nicolas, Marco Winckler, Philippe Palanque, An Attempt to Fathom the Role of Annotations in User-Centered Design Process. Workshop on Research and Practice Challenges for Engineering Interactive Systems while Integrating Multiple Stakeholders Viewpoints, EISMS 2019

List of figures

Figure 1-1 Examples in Early Renaissance documents showing the use of annotations for explain a design/idea	1
Figure 1-2 Prototype of a mini-game	2
Figure 1-3 Three example of the use of annotations on prototype.....	2
Figure 1-4 Chapters of the PhD thesis.....	4
Figure 2-1 Overview of development process as an implicit black box (a) versus an explicit process (b)	8
Figure 2-2 Waterfall process (Royce, 1970)	9
Figure 2-3 V-model development process (McDermid & Ripken, 1983)	9
Figure 2-4 The Nabla model [12].....	10
Figure 2-5 Spiral model development process (Boehm, 1986).....	11
Figure 2-6 RUP development process	12
Figure 2-7 The steps of the software development with SCRUM retrieved from https://www.neonrain.com/agile-scrum-web-development/	13
Figure 2-8 eXtreme Programming development process	14
Figure 2-9 The Star model	14
Figure 2-10 The layered process model [20].....	15
Figure 2-11 OOUI design process	15
Figure 2-12 The iterative-cyclic process	16
Figure 2-13 Usability Design Process.....	17
Figure 2-14 Comparison table between the User-centered design and the Usage-centered design (retrieved from [24])	18
Figure 2-15 ISO 9241-210 standard for human-centred design processes for interactive systems	19
Figure 3-1 Simplified model of annotations defined by the Web Annotation Data Model [32].....	26
Figure 3-2 Formal definition of annotations according to Agosti M. and Ferro N. in [49].....	26
Figure 3-3 Example of freeform annotations (figure taken from [63])	34
Figure 3-4 Representation of the Web Annotation Data Model.....	37
Figure 4-1 Number of both academic and commercial tools per year	51
Figure 4-2 Seeheim model.....	55
Figure 4-3 Visual representation of the navigation in Pidoco.....	56
Figure 5-1 Artefact lifecycle diagram	59
Figure 5-2 Annotation lifecycle diagram	61
Figure 5-3 Artefact lifecycle interacting with the annotation lifecycle through decisions	62
Figure 5-4 Annotation validation process	67
Figure 6-1 Representation of the annotation model inspired from the W3C Web Annotation Data Model	73
Figure 6-2 Graphical representation of the XSD	75
Figure 6-3 XSD describing the main structure of a PandaAnnotation file.....	76
Figure 6-4 XSD of the "files" node	76
Figure 6-5 XSD of the "annotationssets" node.....	77
Figure 6-6 XSD of the "annotationElement" type	78
Figure 6-7 XSD of the "targetsElement" type.....	79
Figure 6-8 Example of annotation file	80
Figure 6-9 Example of annotation targeting several targets.....	81
Figure 6-10 XSD of the "targetsElement"	83
Figure 6-11 XSD of the "attributesElement" type	83

Figure 7-1 Overview of the tool and the files repository used by ARMADILLO	88
Figure 7-2 Example of a CIRCUS project workspace	89
Figure 7-3 Overview of the UI of the ARMADILLO plugin	91
Figure 7-4 Representation of a textual annotation.....	91
Figure 7-5 Example of a CIRCUS project.....	92
Figure 7-6 Screenshot of the PetShop editor	93
Figure 7-7 Screenshot of the targeting system of the annotation plugin	94
Figure 7-8 Screenshot of the fragment selection window for PANDA prototypes	95
Figure 7-9 Example of scenario annotation	95
Figure 7-10 Annotations created over an artefact in the PANDA editor	96
Figure 7-11 Overview of the annotations using ARMADILLO	97
Figure 7-12 "Design journal" extension.....	98
Figure 7-13 "Design journal" search feature	99
Figure 7-14 Details of an annotation in the "Design journal"	99
Figure 8-1 Overview of the interface of PANDA.....	101
Figure 8-2 Overview of annotations supported by PANDA.....	102
Figure 8-3 Template for specifying User Stories as defined by North [109] and Cohn [110]	105
Figure 8-4 Structure of a step from a scenario specified in the PANDA annotation tool support.....	105
Figure 8-5 "Scenario" annotation attached to two states in PANDA.....	106
Figure 8-6 Example of parsing of one step.....	106
Figure 8-7 Properties of "From" and "Search" widgets in the PANDA prototype and the result of the scenario tested	107
Figure 9-1 Image of the WXR application.....	112
Figure 9-2 The task model artefact of the WXR project in HAMSTERS notation using the HAMSTERS task model editor and simulator.....	113
Figure 9-3 The system model artefact of the WXR project using the ICO formalism within PetShop Tool	114
Figure 9-4 The prototype artefact of WXR project using PANDA tool	115
Figure 9-5 Annotations on the medium-fidelity prototype.....	116
Figure 9-6 Annotations displayed in the PetShop editor using the same location properties as the low-fidelity prototype.....	117
Figure 9-7 Annotations displayed in the PetShop editor after moving the annotations for the ICO model	118
Figure 9-8 Overview of the annotations using ARMADILLO	118
Figure 10-1 e-Citiz TM design process with eazly TM	123
Figure 10-2 "My Appeaz" page.....	124
Figure 10-3 Appeaz configuration	125
Figure 10-4 Screenshot of the eazlyStudio TM	126
Figure 10-5 Publishing page	127
Figure 10-6 User management page	128
Figure 10-7 Display of the targets when selecting an annotation	130
Figure 10-8 Details of an annotation.....	130
Figure 10-9 Prototype for the management of annotations created for each versions of the prototype	131
Figure 10-10 Details of the annotation in the annotation page.....	132
Figure 10-11 Integrating a recap of the annotations associated with the current version in the publication process	133
Figure 10-12 Integration of annotations within the editor.....	134

Figure 10-13 Integration of an "Annotation" button for displaying them	134
Figure Annex 7-1 Representation of the dialog model	171
Figure Annex 7-2 Representation of the presentation model	172
Figure Annex 7-3 Dependency diagram of the PANDA ecosystem	178
Figure Annex 7-4 Representation of the PANDA ecosystem with their artefacts within the UCD process as described in the Figure 2-15	181
Figure Annex 7-5 Palette of widgets instantiated in PANDA and properties of a "Button" widget defined by the ontology	183
Figure Annex 7-6 Specification of the presentation for the two states of the prototype	184
Figure Annex 7-7 Presentation of the cloning feature	185
Figure Annex 7-8 Prototype combining the presentation and the dialog	186
Figure Annex 7-9 Initial state of the prototype	187
Figure Annex 7-10 Active state outlined with a red stroke	187
Figure Annex 7-11 "Choose flight" state as the active state after clicking on the "Search" button ...	188
Figure Annex 8-1 e-Citiz™ application environment	191
Figure Annex 8-2 e-Citiz™ design process	192
Figure Annex 8-3 Example of a BPMN in the e-Citiz Studio™	193
Figure Annex 8-4 Screenshot of an e-Citiz™ application	194
Figure Annex 8-5 Data Structure of a declaration of arrival	196
Figure Annex 8-6 Homepage of eazly™ after logging in	198
Figure Annex 8-7 Creation of the first page of the Appeaz	199
Figure Annex 8-8 Specification of the page	200
Figure Annex 8-9 Adding a new page	200

List of table

Table 2-1 Synthetic analysis of the development process	22
Table 3-1 Table of development team roles extracted from [27]	29
Table 3-2 Observations and conclusions on the information necessary for the rationale design in [60]	32
Table 3-3 Compatibility between M. Agosti and N. Ferro's classification with AM. Naghsh's classification of annotation	33
Table 3-4 Compatibility between AM. Naghsh's classification with J. Virbel's classification of annotation	33
Table 3-5 Compatibility between M. Agosti and N. Ferro's classification with J. Virbel's classification of annotation	34
Table 4-1 Investigated topics and features for the state of the art on annotation tools	42
Table Annex 7-1 Table of the modules of the PANDA ecosystem	177
Table Annex 8-1 Main results of the 5M model analysis	198

Annex 1. List of annotation tools examined for the state of the art

Year of publication	Name of the tool	URL	Academic tool
1988			
	Quilt	https://dl.acm.org/citation.cfm?id=62282 and https://dl.acm.org/citation.cfm?id=45414	Yes
	Amaya	https://www.w3.org/Amaya/ and https://dev.w3.org/Amaya/doc/WX/Annotations.html	Yes
	Annozilla (based on the Annotea project)	http://annozilla.mozdev.org/	No
2005			
	Diigo	https://www.diigo.com/	No
2007			
	sense.us	http://vis.stanford.edu/papers/senseus	Yes
2008			
	Protonotes	http://www.protonotes.com/	No
2009			
	SparTag.us	https://dl.acm.org/citation.cfm?id=1385582	Yes
	HyperImage 3: Virtual Research Environment	http://hyperimage.ws/en/	Yes
2010			
	D.note	https://dl.acm.org/citation.cfm?doid=1753326.1753400	Yes
2011			
	List-it	https://dspace.mit.edu/handle/1721.1/73002	Yes
	LiquidText	https://dl.acm.org/citation.cfm?id=1979430	Yes
	Zydeco	https://dl.acm.org/citation.cfm?id=1979038	Yes
	Annotatorjs	http://annotatorjs.org/	No
2012			
	Elias' « unnamed prototype »	https://dl.acm.org/citation.cfm?id=2208288	Yes
	ChronoViz	https://dl.acm.org/citation.cfm?id=2208590	Yes
	GatherReader	https://dl.acm.org/citation.cfm?id=2208327	Yes
	Annotorious	https://annotorious.github.io/	Yes
	Domeo	https://jbiomedsem.biomedcentral.com/articles/10.1186/2041-1480-3-S1-S1	Yes
2014			
	Instant Annotation	https://dl.acm.org/citation.cfm?id=2557209	Yes
	Neonion	https://dl.acm.org/citation.cfm?id=2699012	Yes
2017			
	Dokieli	http://csarven.ca/dokieli-rww	Yes
	Hypothesis	https://web.hypothes.is/	No

	Authorea	https://www.authorea.com/	No
	Pundit Annotator	http://net7.github.io/pundit2/	Yes
	Ponga	https://www.ponga.com/	No

Annex 2. List of prototyping tools examined for the state of the art

Name of the tool	URL	Academic tool
iRise	https://www.irise.com/	No
SoftAndGui	http://www.softandgui.co.uk/	No
UxPin	https://www.uxpin.com/	No
AdobeXD	https://www.adobe.com/fr/products/xd.html	No
Microsoft visio	https://products.office.com/fr-fr/visio/flowchart-software	No
Smartdraw	https://www.smartdraw.com/	No
Silk	Landay, J. A., & Myers, B. A. (1995, May). Interactive sketching for the early stages of user interface design. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 43-50). ACM Press/Addison-Wesley Publishing Co.	Yes
Axure	https://www.axure.com/	No
DEMAIS	https://dl.acm.org/citation.cfm?id=642666	Yes
GUI Design Studio	https://www.carettasoftware.com/guidesignstudio/	No
MockupScreens	http://www.mockupscreens.com/	No
JustinMind	https://www.justinmind.com/	No
Balsamiq	https://balsamiq.com/	No
DesignerVista	http://www.designervista.com/	No
InPreso Screens	http://www.inpreso.com/inpresoscreens/	No
MockingBird	https://gomockingbird.com/home	No
PencilProject	http://pencil.evolus.vn/	No
Pidoco	https://pidoco.com/en	No
ProtoShare	http://www.protoshare.com/	No
WireframeSketcher	https://wireframesketcher.com/	No
ActiveStoryEnhanced	http://activestoryenhanced.codeplex.com/	Yes
Cacoo	https://cacoo.com/fr/	No
Crank Storyboard Designer	http://www.cranksoftware.com/	No
Creately	https://creatly.com/	No
FlairBuilder	http://flairbuilder.com/	No
ForeUI	http://www.foreui.com/	No
Gliffy	https://www.gliffy.com/	No
Microsoft SketchFlow	https://msdn.microsoft.com/fr-fr/library/cc296227.aspx	No
iPlotz	https://iplotz.com/	No
BluePrint	http://www.groosoft.com/blueprint/	No
FrameBox	http://framebox.org/	No
HotGloo	https://www.hotgloo.com/	No
LucidChart	https://www.lucidchart.com/	No
Mockflow	https://www.mockflow.com/	No
Sketch	https://www.sketchapp.com	No
Antetype	http://antetype.com/	No
Draw.io	https://www.draw.io/	No
Lumzy	http://www.lumzy.com/	No
MockupBuilder	http://mockupbuilder.com/	No

Mockups.me	http://mockups.me/	No
MockupTiger	https://www.mockuptiger.com/	No
PowerMockup	https://www.powermockup.com/	No
Proto.io	https://proto.io/	No
FluidUI	https://www.fluidui.com/	No
IndigoStudio	https://www.infragistics.com/products/indigo-studio	No
Moqups	https://moqups.com/	No
Prototyping on Paper (Marvel)	https://marvelapp.com	No
Alouka	http://www.alouka.com/	No
Concept.Ly	http://concept.ly/	No
InVision	https://www.invisionapp.com/	No
NinjaMock	https://ninjamock.com/	No
Notism	https://www.notism.io/	No
MockupPlus	https://www.mockplus.com/	No
SnapUp	http://www.quickfocus.com	No
Atomic	http://atomic.io/	No

Annex 3. Synthesis of the analysis of annotation support

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Number of tools														
2	ANNOTATING TOOLS														
3	PROTOTYPING TOOLS														
4	BOTH ANNOTATING AND PROTOTYPING														
5	Prototyping tools	Total number	Academic	Industrial	Total for Annotating tool		Total number	Academic	Industrial	Total for Prototyping tool		Academic	Industrial	Total for examined tools	
6	Prototyping tools examined	121													
7	Not supporting annotation	40													
8	Support annotation	81													
9	Prototyping tools updated with annotation features between the reviews	3													
10	Supporting annotations														
11	Total number of tools in this category	81	18	7	25		81	3	53	56		21	60	81	
12	Total number of tools examined	81	21	60	81		81	21	60	81		21	60	81	
13	Structure of annotations														
14	Annotation creation for annotating tool														
15	Independence of annotations	Total number	Academic	Industrial	Total for Annotating tool		Total number	Academic	Industrial	Total for Prototyping tool		Academic	Industrial	Total for examined tools	
16	Dependent annotations	13		6	19		29		10	10		13		16	
17	Not determined	1		0	1		1		0	0		1		0	
18	Total	25	18	7	25		25	0	0	0					
19															
20	Graphics														
21	Targeting first or Annotation first														
22	Annotation creation for annotating tool														
23	Annotation creation first	48	4	1	5		48	3	40	43		7	41	48	
24	Target selection first	29	13	6	19		29	0	10	10		13	16	29	
25	Not determined	1	1	0	1		1	0	0	0		1	0	1	
26	Both	3	0	0	0		3	0	3	3		0	3	3	
27	Total	81	18	7	25		81	3	53	56		21	60	81	
28															
29	Annotation creation for prototyping tool														
30	Annotation creation first	29	13	6	19		29	0	10	10		13	16	29	
31	Target selection first	1	1	0	1		1	0	0	0		1	0	1	
32	Both	3	0	0	0		3	0	3	3		0	3	3	
33	Total	81	18	7	25		81	3	53	56		21	60	81	
34															
35	Type of annotation provided														
36	Textual annotation														
37	Text (sticky note, text field)	79	16	7	23		79	3	53	56		19	60	79	
38	Discussion thread	36	8	5	13		36	0	23	23		8	28	36	
39	Tags	9	6	2	8		9	0	1	1		6	3	9	
40	Todo list	1	0	0	0		1	0	1	1		0	1	1	
41															
42	Graphical annotation														
43	Highlight	15	7	3	10		15	0	5	5		7	8	15	
44	Freeform sketch	17	5	1	6		17	2	9	11		7	10	17	
45	Callout/Markup (pins, icon, arrows...)	21	0	0	0		21	0	21	21		0	21	21	
46	Shape area (Rectangle, circle, braces, cross, scratches...)	17	2	1	3		17	0	14	14		2	15	17	
47	Image	3	0	2	2		3	0	1	1		0	3	3	
48															
49	Other annotation														
50	Reference (with an hyperlink, excerpt)	7	6	1	7		7	0	0	0		6	1	7	
51	Enclosed file	3	0	0	0		3	0	3	3		0	3	3	
52	Semantic annotation	3	3	0	3		3	0	0	0		3	0	3	
53	Voice recording	3	2	0	2		3	1	0	1		3	0	3	
54	Bookmark	3	3	0	3		3	0	0	0		3	0	3	
55	Modifications as revision	1	0	0	1		1	1	0	1		1	0	1	
56															
57	Targeting														
58	Object represented in the tool														
59	Text (word, pdf...)	8	6	2	8		8	0	0	0		6	2	8	
60	Image	4	3	1	4		4	0	0	0		3	1	4	
61	Html	12	7	5	12		12	0	0	0		7	5	12	
62	Presentation	56	0	0	0		56	3	53	56		3	53	56	
63	Dialogue	10	0	0	0		10	1	9	10		1	9	10	
64	Other (Visual representation of a data)	3	3	0	3		3	0	0	0		3	0	3	
65	Tools presenting both presentation & dialogue	D.Notes					Demaix								
66															
67	Target supported (features provided for it because manual referencing d														
68	Fragment of the artefact	20	13	7	20		20	0	0	0		13	7	20	
69	Artefact itself (global annotation)	7	3	2	5		7	2	0	2		5	2	7	
70	Presentation	56	0	0	0		56	3	53	56		3	53	56	
71	Dialogue	4	0	0	0		4	1	3	4		1	3	4	
72	Page of a prototype	22	0	0	0		22	1	21	22		1	21	22	
73	Multi target supported	20	6	0	6		20	1	13	14		7	13	20	
74	Data	3	3	0	3		3	0	0	0		3	0	3	
75															
76	Targeting method when a fragment can be annotated														
77	Selecting object or highlight	23	11	5	16		23	0	7	7		11	12	23	
78	Geographic position	51	1	2	3		51	3	45	48		4	47	51	
79	Shape delimiting an area	26	3	2	5		26	0	21	21		3	23	26	
80	Duration	1	1	0	1		1	0	0	0		1	0	1	
81	Extract or citation	2	2	0	2		2	0	0	0		2	0	2	
82	Reference in text	0	0	0	0		0	0	0	0		0	0	0	
83	Property association	9	0	0	0		9	0	9	9		0	9	9	
84															
85	Annotation Management														
86	Annotation Retrieval														
87	Lifespan of the artefact	9	3	2	5		9	1	3	4		4	5	9	
88	Lifespan of the annotation	19	1	2	3		19	0	16	16		1	18	19	
89	Search Support	15	9	3	12		15	1	2	3		10	5	15	
90	Listing annotation	29	6	4	10		29	0	19	19		6	23	29	
91	Filtering annotation	17	8	3	11		17	0	6	6		8	9	17	
92															
93	Navigation														
94	Annotation to Target	24	12	4	16		24	0	8	8		12	12	24	
95	Target to Annotation	30	8	5	13		30	0	17	17		8	22	30	
96															
97	Collaborative aspects for annotation														
98															
99	Multi user support	55	14	7	21		55	1	33	34		15	40	55	
100	Restriction based on users' rights	27	8	5	13		27	1	13	14		9	18	27	
101	Identification of the author possible	42	8	6	14		42	0	28	28		8	34	42	

Annex 4. XSD of the annotation file

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.irit.fr/ICS/PANDAANNOTATION/8.0" xmlns="http://www.irit.fr/ICS/PANDAANNOTATION/8.0">
3   <!-- *****-->
4   <!-- ANNOTATIONS ROOT ELEMENT -->
5   <!-- *****-->
6   <xs:annotation>
7     <xs:documentation xml:lang="en">
8       This Schema defines an Annotation
9     </xs:documentation>
10  </xs:annotation>
11  <xs:element name="pandaannotation">
12    <xs:complexType>
13      <xs:sequence>
14        <xs:element name="files">
15          <xs:complexType>
16            <xs:sequence>
17              <xs:element name="file" minOccurs="0" maxOccurs="unbounded">
18                <xs:complexType>
19                  <xs:sequence>
20                    <xs:element name="layers" minOccurs="1">
21                      <xs:complexType>
22                        <xs:sequence>
23                          <xs:element name="layer" minOccurs="0" maxOccurs="unbounded">
24                            <xs:complexType>
25                              <xs:sequence>
26                                <xs:attribute name="name" type="xs:string"/>
27                                <xs:attribute name="position" type="xs:int"/>
28                              </xs:sequence>
29                            </xs:complexType>
30                          </xs:element>
31                        </xs:sequence>
32                      </xs:complexType>
33                    </xs:element>
34                    <xs:element name="referenceSet" type="referenceSet" minOccurs="0" maxOccurs="unbounded"/>
35                  </xs:sequence>
36                </xs:complexType>
37              </xs:element>
38            </xs:sequence>
39          </xs:complexType>
40        </xs:element>
41      </xs:sequence>
42    </xs:complexType>
43  </xs:element>
44  <xs:element name="annotationsets">
45    <xs:complexType>
46      <xs:sequence>
47        <xs:element name="annotationset" minOccurs="0" maxOccurs="unbounded">
48          <xs:complexType>
49            <xs:sequence>
50              <xs:element name="historyitems">

```



```

51 |         <xs:complexType>
52 |             <xs:sequence>
53 |                 <xs:element name="historyitem" type="historyitemElement" minOccurs="0" maxOccurs="unbounded"/>
54 |             </xs:sequence>
55 |         </xs:complexType>
56 |     </xs:element>
57 |     <xs:element name="annotations">
58 |         <xs:complexType>
59 |             <xs:sequence>
60 |                 <xs:element name="annotation" type="annotationElement" minOccurs="0" maxOccurs="unbounded"/>
61 |             </xs:sequence>
62 |         </xs:complexType>
63 |     </xs:element>
64 | </xs:sequence>
65 | <xs:attribute name="versionnumber" type="xs:int" use="required"/>
66 | <xs:attribute name="status" type="annotationstatus" use="required"/>
67 | </xs:complexType>
68 | </xs:element>
69 | </xs:sequence>
70 | </xs:complexType>
71 | </xs:element>
72 | </xs:sequence>
73 | <xs:attribute name="username" type="xs:string" use="required"/>
74 | <xs:attribute name="session" type="xs:string"/>
75 | <xs:attribute name="date" type="xs:dateTime"/>
76 | </xs:complexType>
77 | </xs:element>
78 |
79 | <!--historyitemElement-->
80 | <xs:complexType name="historyitemElement">
81 |     <xs:attribute name="date" type="xs:string"/>
82 |     <xs:attribute name="comment" type="xs:string"/>
83 |     <xs:attribute name="newstatus" type="annotationstatus"/>
84 | </xs:complexType>
85 |
86 | <!-- *****-->
87 | <!-- ANNOTATION ELEMENT -->
88 | <!-- *****-->
89 |
90 | <xs:complexType name="annotationElement">
91 |     <xs:sequence>
92 |         <xs:element name="targets" type="targetsElement"/>
93 |         <xs:element name="core">
94 |             <xs:complexType>
95 |                 <xs:sequence>
96 |                     <xs:choice>
97 |                         <xs:element name="drawing" type="drawingElement"/>
98 |                         <xs:element name="externalfile" type="externalfileElement"/>
99 |                         <xs:element name="marking" type="markingElement"/>
100 |                         <xs:element name="simpletextual" type="xs:string"/>

```

```

101         <xs:element name="survey" type="surveyElement"/>
102         <xs:element name="textual" type="textualElement"/>
103         <xs:any namespace="##other" processContents="lax"/>
104     </xs:choice>
105 </xs:sequence>
106 </xs:complexType>
107 </xs:element>
108 <xs:element name="attributes" type="attributesElement"/>
109 <xs:element name="referenceSetIds" type="xs:int" minOccurs="0" maxOccurs="unbounded"/>
110 </xs:sequence>
111 <xs:attribute name="id" type="xs:ID"/>
112 <xs:attribute name="type" type="xs:string"/>
113 <xs:attribute name="refid" type="xs:IDREF"/>
114 </xs:complexType>
115
116 <xs:complexType name="attributeElement">
117     <xs:attribute name="name" type="xs:string"/>
118     <xs:attribute name="type" type="xs:string"/>
119     <xs:attribute name="value" type="xs:string"/>
120 </xs:complexType>
121
122 <!-- *****-->
123 <!-- TARGETS ELEMENT -->
124 <!-- *****-->
125 <xs:complexType name="targetsElement">
126     <xs:sequence>
127         <xs:element name="target" minOccurs="0" maxOccurs="unbounded">
128             <xs:complexType>
129                 <xs:sequence>
130                     <xs:element name="fragments">
131                         <xs:complexType>
132                             <xs:sequence>
133                                 <xs:element name="fragment" minOccurs="0" maxOccurs="unbounded">
134                                     <xs:complexType>
135                                         <xs:attribute name="id" type="xs:ID"/>
136                                     </xs:complexType>
137                                 </xs:element>
138                             </xs:sequence>
139                         </xs:complexType>
140                     </xs:element>
141                     <xs:element name="attributes" type="attributesElement"/>
142                 </xs:sequence>
143                 <xs:attribute name="id" type="xs:ID"/>
144             </xs:complexType>
145         </xs:element>
146     </xs:sequence>
147 </xs:complexType>
148
149 <!-- *****-->
150 <!-- REFERENCE SETS ELEMENT -->

```

```

149 <!-- *****-->
150 <!-- REFERENCE SETS ELEMENT -->
151 <!-- *****-->
152 <xs:complexType name="referenceSet">
153   <xs:sequence>
154     <xs:element name="referencedfile" minOccurs="0" maxOccurs="unbounded">
155       <xs:complexType>
156         <xs:sequence>
157           <xs:element name="tags" type="xs:string" minOccurs="0" maxOccurs="3"/>
158           <xs:element name="referencedobject" minOccurs="0" maxOccurs="unbounded">
159             <xs:complexType>
160               <xs:sequence>
161                 <xs:element name="tags" type="xs:string" minOccurs="0" maxOccurs="3"/>
162               </xs:sequence>
163               <xs:attribute name="id" type="xs:int"/>
164               <xs:attribute name="name" type="xs:string"/>
165             </xs:complexType>
166           </xs:element>
167         </xs:sequence>
168         <xs:attribute name="id" type="xs:int"/>
169         <xs:attribute name="layer" type="xs:string"/>
170         <xs:attribute name="file" type="xs:string"/>
171       </xs:complexType>
172     </xs:element>
173   </xs:sequence>
174   <xs:attribute name="id" type="xs:int"/>
175   <xs:attribute name="posX" type="xs:int"/>
176   <xs:attribute name="posY" type="xs:int"/>
177 </xs:complexType>
178
179 <!-- *****-->
180 <!-- ANNOTATIONS TYPE ELEMENT -->
181 <!-- *****-->
182 <xs:complexType name="drawingElement">
183   <xs:sequence>
184     <xs:element name="shape" type="shapeElement" minOccurs="0" maxOccurs="unbounded"/>
185   </xs:sequence>
186 </xs:complexType>
187 <xs:complexType name="shapeElement">
188   <xs:sequence>
189     <xs:element name="point" minOccurs="0" maxOccurs="unbounded">
190       <xs:complexType>
191         <xs:attribute name="positionx" type="xs:decimal" use="required"/>
192         <xs:attribute name="positiony" type="xs:decimal" use="required"/>
193       </xs:complexType>
194     </xs:element>
195   </xs:sequence>
196   <xs:attribute name="alpha" type="xs:decimal"/>
197   <xs:attribute name="blue" type="xs:decimal"/>
198   <xs:attribute name="green" type="xs:decimal"/>

```

```

197 <xs:attribute name="blue" type="xs:decimal"/>
198 <xs:attribute name="green" type="xs:decimal"/>
199 <xs:attribute name="red" type="xs:decimal"/>
200 <xs:attribute name="width" type="xs:decimal"/>
201 <xs:attribute name="type" type="shapeType"/>
202 </xs:complexType>
203 <xs:complexType name="externalfileElement">
204 <xs:simpleContent>
205 <xs:extension base="xs:string">
206 <xs:attribute name="filetype" type="xs:string"/>
207 <xs:attribute name="url" type="xs:string"/>
208 </xs:extension>
209 </xs:simpleContent>
210 </xs:complexType>
211 <xs:complexType name="markingElement">
212 <xs:simpleContent>
213 <xs:extension base="xs:string">
214 <xs:attribute name="type" type="markingType"/>
215 </xs:extension>
216 </xs:simpleContent>
217 </xs:complexType>
218 <xs:complexType name="textualElement">
219 <xs:sequence>
220 <xs:element name="items">
221 <xs:complexType>
222 <xs:sequence>
223 <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
224 <xs:complexType>
225 <xs:attribute name="value" type="xs:string"/>
226 </xs:complexType>
227 </xs:element>
228 </xs:sequence>
229 </xs:complexType>
230 </xs:element>
231 </xs:sequence>
232 <xs:attribute name="type" type="xs:string"/>
233 <xs:attribute name="folded" type="xs:boolean"/>
234 </xs:complexType>
235 <xs:complexType name="surveyElement">
236 <xs:sequence>
237 <xs:element name="items">
238 <xs:complexType>
239 <xs:sequence>
240 <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
241 <xs:complexType>
242 <xs:attribute name="id" type="xs:int"/>
243 <xs:attribute name="value" type="xs:string"/>
244 </xs:complexType>
245 </xs:element>
246 </xs:sequence>

```



```

246         </xs:sequence>
247     </xs:complexType>
248 </xs:element>
249 <xs:element name="votes">
250     <xs:complexType>
251         <xs:sequence>
252             <xs:element name="vote" minOccurs="0" maxOccurs="unbounded">
253                 <xs:complexType>
254                     <xs:attribute name="user" type="xs:string"/>
255                     <xs:attribute name="value" type="xs:int"/>
256                 </xs:complexType>
257             </xs:element>
258         </xs:sequence>
259     </xs:complexType>
260 </xs:element>
261 </xs:sequence>
262 <xs:attribute name="type" type="xs:string"/>
263 </xs:complexType>
264 <xs:simpleType name="shapeType">
265     <xs:restriction base="xs:string">
266         <xs:enumeration value="polyline"/>
267     </xs:restriction>
268 </xs:simpleType>
269 <xs:simpleType name="annotationstatus">
270     <xs:restriction base="xs:string">
271         <xs:enumeration value="annotating"/>
272         <xs:enumeration value="request review"/>
273         <xs:enumeration value="reviewing"/>
274         <xs:enumeration value="reviewed"/>
275         <xs:enumeration value="rebuttal"/>
276     </xs:restriction>
277 </xs:simpleType>
278 <xs:simpleType name="markingType">
279     <xs:restriction base="xs:string">
280         <xs:enumeration value="Info"/>
281         <xs:enumeration value="Help"/>
282         <xs:enumeration value="Ok"/>
283         <xs:enumeration value="Warning"/>
284         <xs:enumeration value="Error"/>
285     </xs:restriction>
286 </xs:simpleType>
287 <xs:simpleType name="tagType">
288     <xs:restriction base="xs:string">
289         <xs:enumeration value="none"/>
290         <xs:enumeration value="comment"/>
291         <xs:enumeration value="example"/>
292         <xs:enumeration value="explanation"/>
293         <xs:enumeration value="problem"/>
294         <xs:enumeration value="question"/>
295         <xs:enumeration value="reference"/>

```

```

295         <xs:enumeration value="reference"/>
296         <xs:enumeration value="reminder"/>
297         <xs:enumeration value="see also"/>
298         <xs:enumeration value="suggestion"/>
299         <xs:enumeration value="todo"/>
300         <xs:enumeration value="other"/>
301     </xs:restriction>
302 </xs:simpleType>
303 <!-- *****-->
304 <!-- ATTRIBUTES TYPE ELEMENT -->
305 <!-- *****-->
306 <xs:complexType name="attributesElement">
307     <xs:sequence>
308         <xs:element name="attribute" type="attributeElement" minOccurs="0" maxOccurs="unbounded">
309             </xs:element>
310         </xs:sequence>
311     </xs:complexType>
312 </xs:schema>

```


Annex 5. XSD of the PANDA dialog file

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema
3      xmlns:xs="http://www.w3.org/2001/XMLSchema"
4      targetNamespace="http://www.irit.fr/ICS/PANDAAUTOMATON/7.0"
5      xmlns="http://www.irit.fr/ICS/PANDAAUTOMATON/7.0"
6      elementFormDefault="qualified" >
7
8      <!-- ***** -->
9      <!-- AUTOMATON ROOT ELEMENT -->
10     <!-- ***** -->
11     <xs:element name="pandaautomaton">
12         <xs:complexType>
13             <xs:sequence>
14                 <xs:element name="states" minOccurs="1" maxOccurs="1">
15                     <xs:complexType>
16                         <xs:sequence>
17                             <xs:element name="state" minOccurs="0" maxOccurs="unbounded">
18                                 <xs:complexType>
19                                     <xs:attribute name="containerref" type="xs:integer"/>
20                                     <xs:attribute name="id" type="xs:integer"/>
21                                     <xs:attribute name="positionx" type="xs:decimal"/>
22                                     <xs:attribute name="positiony" type="xs:decimal"/>
23                                     <xs:attribute name="name" type="xs:string"/>
24                                 </xs:complexType>
25                             </xs:element>
26                         </xs:sequence>
27                     <xs:attribute name="id" type="xs:integer"/>
28                 </xs:complexType>
29             </xs:element>
30             <xs:element name="transitions" minOccurs="1" maxOccurs="1">
31                 <xs:complexType>
32                     <xs:sequence>
33                         <xs:element name="transition" minOccurs="0" maxOccurs="unbounded">
34                             <xs:complexType>
35                                 <xs:sequence>
36                                     <xs:element name="actions">
37                                         <xs:complexType>
38                                             <xs:sequence>
39                                                 <xs:element name="action" minOccurs="0" maxOccurs="unbounded">
40                                                     <xs:complexType>
41                                                         <xs:attribute name="name" type="xs:string"/>
42                                                     </xs:complexType>
43                                                 </xs:element>
44                                             </xs:sequence>
45                                         </xs:complexType>
46                                     </xs:element>
47                                     <xs:element name="conditions">
48                                         <xs:complexType>
49                                             <xs:sequence>
50                                                 <xs:element name="condition" minOccurs="0" maxOccurs="unbounded">
51                                                     <xs:complexType>
52                                                         <xs:attribute name="name" type="xs:string"/>
53                                                     </xs:complexType>
54                                                 </xs:element>
55                                             </xs:sequence>
56                                         </xs:complexType>
57                                     </xs:element>
58                                 </xs:sequence>
59                                 <xs:attribute name="event" type="xs:string"/>
60                                 <xs:attribute name="sourcestate" type="xs:integer"/>
61                                 <xs:attribute name="targetstate" type="xs:integer"/>
62                                 <xs:attribute name="widgetref" type="xs:integer"/>
63                             </xs:complexType>
64                         </xs:element>
65                     </xs:sequence>
66                 </xs:complexType>
67             </xs:element>
68         </xs:sequence>
69         <xs:attribute name="name" type="xs:string" use="required"/>
70         <xs:attribute name="initialstate" type="xs:integer" use="required"/>
71     </xs:complexType>
72 </xs:element>
73 </xs:schema>

```


Annex 6. XSD of the PANDA presentation file

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema
3      xmlns:xs="http://www.w3.org/2001/XMLSchema"
4      targetNamespace="http://www.irit.fr/ICS/PANDAPRESENTATION/7.0"
5      xmlns="http://www.irit.fr/ICS/PANDAPRESENTATION/7.0"
6      elementFormDefault="qualified" >
7
8      <!-- *****-->
9      <!-- PRESENTATION ROOT ELEMENT -->
10     <!-- *****-->
11     <xs:element name="pandapresentation">
12         <xs:complexType>
13             <xs:sequence>
14                 <xs:element name="canvas" type="containertype" minOccurs="1" maxOccurs="1"/>
15             </xs:sequence>
16             <xs:attribute name="ontologyloaded" type="xs:string"/>
17             <xs:attribute name="status" type="presentationstatus"/>
18         </xs:complexType>
19     </xs:element>
20
21     <xs:complexType name="containertype">
22         <xs:sequence>
23             <xs:element name="containers" minOccurs="0" maxOccurs="1">
24                 <xs:complexType>
25                     <xs:sequence>
26                         <xs:element name="container" type="containertype" minOccurs="0" maxOccurs="unbounded"/>
27                     </xs:sequence>
28                 </xs:complexType>
29             </xs:element>
30             <xs:element name="widgets" minOccurs="0" maxOccurs="1">
31                 <xs:complexType>
32                     <xs:sequence>
33                         <xs:element name="widget" type="widget" minOccurs="0" maxOccurs="unbounded"/>
34                     </xs:sequence>
35                 </xs:complexType>
36             </xs:element>
37             <xs:element name="ontologicalproperties" type="ontologicalpropiertiestype" minOccurs="0" maxOccurs="unbounded"/>
38         </xs:sequence>
39         <xs:attribute name="enable" type="xs:boolean" use="required"/>
40         <xs:attribute name="equivalentid" type="xs:integer"/>
41         <xs:attribute name="height" type="xs:decimal" use="required"/>
42         <xs:attribute name="id" type="xs:integer" use="required"/>
43         <xs:attribute name="name" type="xs:string" use="required"/>
44         <xs:attribute name="positionx" type="xs:decimal" use="required"/>
45         <xs:attribute name="positiony" type="xs:decimal" use="required"/>
46         <xs:attribute name="type" type="xs:string" use="required"/>
47         <xs:attribute name="value" type="xs:string" use="required"/>
48         <xs:attribute name="visible" type="xs:boolean" use="required"/>
49         <xs:attribute name="width" type="xs:decimal" use="required"/>
50     </xs:complexType>
51

```

```

51
52 <xs:complexType name="widget">
53   <xs:sequence>
54     <xs:element name="ontologicalproperties" type="ontologicalpropertytype" minOccurs="0" maxOccurs="unbounded"/>
55   </xs:sequence>
56   <xs:attribute name="enable" type="xs:boolean" use="required"/>
57   <xs:attribute name="equivalentid" type="xs:integer"/>
58   <xs:attribute name="height" type="xs:decimal" use="required"/>
59   <xs:attribute name="id" type="xs:integer" use="required"/>
60   <xs:attribute name="name" type="xs:string" use="required"/>
61   <xs:attribute name="positionx" type="xs:decimal" use="required"/>
62   <xs:attribute name="positiony" type="xs:decimal" use="required"/>
63   <xs:attribute name="type" type="xs:string" use="required"/>
64   <xs:attribute name="value" type="xs:string" use="required"/>
65   <xs:attribute name="visible" type="xs:boolean" use="required"/>
66   <xs:attribute name="width" type="xs:decimal" use="required"/>
67 </xs:complexType>
68
69 <xs:complexType name="ontologicalpropertytype">
70   <xs:sequence>
71     <xs:element name="ontologicalproperty" type="ontologicaltype" minOccurs="0" maxOccurs="unbounded"/>
72   </xs:sequence>
73   <xs:attribute name="ontologyIRI" type="xs:string" use="required"/>
74 </xs:complexType>
75
76 <xs:complexType name="ontologicaltype">
77   <xs:attribute name="name" type="xs:string" use="required"/>
78   <xs:attribute name="type" type="xs:string"/>
79   <xs:attribute name="value" type="xs:string"/>
80   <xs:attribute name="valueisvalid" type="xs:boolean"/>
81 </xs:complexType>
82
83 <xs:simpleType name="presentationstatus">
84   <xs:restriction base="xs:string">
85     <xs:enumeration value="none"/>
86     <xs:enumeration value="prototyping"/>
87     <xs:enumeration value="request annotation"/>
88     <xs:enumeration value="annotated"/>
89     <xs:enumeration value="request review"/>
90     <xs:enumeration value="reviewed"/>
91     <xs:enumeration value="synthesizing"/>
92     <xs:enumeration value="planning"/>
93     <xs:enumeration value="decision"/>
94     <xs:enumeration value="closed"/>
95   </xs:restriction>
96 </xs:simpleType>
97 </xs:schema>

```

Annex 7. Presentation of the PANDA ecosystem

Summary

This annex presents the details of the PANDA ecosystem.

The first section of this annex presents the different models managed by PANDA and how they can be connected together to obtain an executable prototype. The second section gives an overview of the modular architecture of the PANDA ecosystem through the presentation of its modules which are aimed at supporting the traceability of the design using annotations. The third section gives an example of creation of a prototype using PANDA and presents the preview system of the prototypes.

7.1. Introduction

This annex details the PANDA ecosystem and its implementation. This annex is complementary to the Chapter 8.

This framework consists in a set of modules dedicated to support activities of the design process while building a project workspace that will be used for studying solutions for the traceability of the evolution of the design.

Beyond the integration of annotations within a new editing tool, the goal of this framework is to feature a study platform oriented toward the activities of the UCD process in the CIRCUS tool suite. The traceability of the evolution of the design through the iterations of the design process using annotations is the current investigations targeted by this study platform. Thus, the current development of the PANDA ecosystem is aimed at supporting the management of prototypes and monitor their evolution during the UCD process.

7.2. User interface models described in the PANDA ecosystem

- **The dialog model**

The dialog model is defined by a state machine. A simple example of dialog model is illustrated in the Figure Annex 7-1. This model consists in two states “Find flight” and “Choose Flights” connected by two transitions.

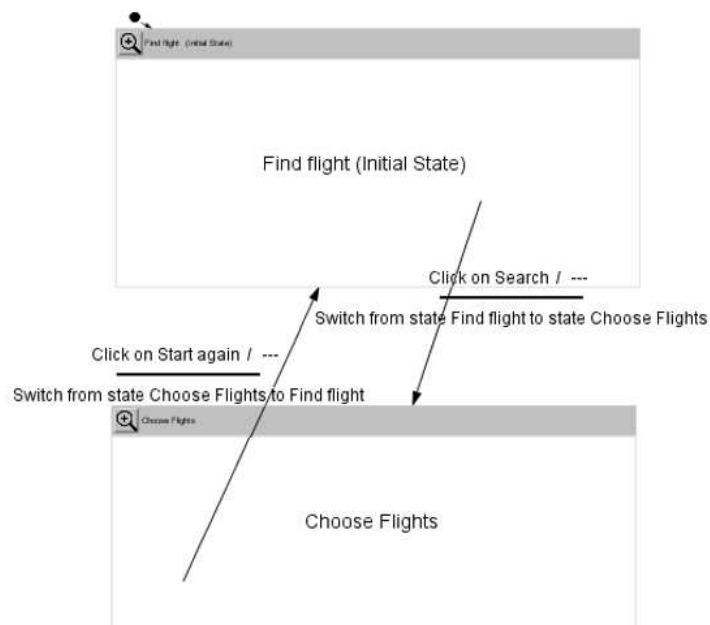


Figure Annex 7-1 Representation of the dialog model

The dialog model is defined by a set of states (represented by the rectangles), a set of transitions connecting those states (represented by the arrows and the triplet $\frac{\text{Event/Condition(s)}}{\text{Action(s)}}$), and an initial state (which is a state connected to the initial state marker represented by a black spot). The XSD file describing the format of a dialog model is available in the Annex 5.

Each state of the dialog will be used to define one state of the interactive system. A state is defined by an id and a name. A state also features other attributes used for its representation within the dialog editor (i.e. positions attributes) and for combining the dialog with a presentation model (i.e. “containerref” attribute which correspond to the ID of a container defined in the presentation model to connect).

The transitions define the changes of states possible during the execution of the dialog. A transition is composed of three main attributes and two child nodes.

A transition is first defined by a source state, a target state and an event. These attributes are defining in which state the transition can be used, what event might trigger its use during the execution of the dialog and which state will be activated after crossing the transition. The fourth attribute, namely the “widgetref” attribute, is used to associate the transition to a widget defined in a presentation model which can trigger the event.

The two child nodes of a transitions are used to specify the conditions to cross the transition when the event occurs and the actions triggered by the transition. In the current specification of the dialog model, conditions and events are described as a list of string.

While this implementation might be minimal at the moment, an evolution of the management of these aspects of the dialog is planned. Indeed, a module dedicated to the management of actions and conditions can be implemented within the PANDA framework. This module would load and process reusable scripts based on the specified name in the dialog model to check if the transition can be crossed for conditions or after the crossing of a transition for actions. These scripts could be coded with any compatible language for their integration into the final implementation of an application but they should be compliant with an API to ensure that these scripts can communicate with the dialog module.

- **The presentation model**

The presentation model is the representation of the UI in the PANDA prototyping tool. This model consists in the definition of containers and widgets as presented in the Figure Annex 7-2.

Search a flight				
Menu	From	To	Passenger	
	<input type="text" value="Empty"/>	<input type="text" value="Empty"/>	<input type="text" value="Empty"/>	
Flight only				
Flight & Hotel	Depart	Return		
Flight & car	<input type="text" value="Empty"/>	<input type="text" value="Empty"/>	<input type="button" value="Search"/>	

Figure Annex 7-2 Representation of the presentation model

The XSD file describing the format of a presentation model is available in the Annex 6.

The root of the presentation model is the “canvas” which is the main container corresponding to the drawing area within the PANDA prototyping tool. As a container, it is composed of attributes defining its look within the editor (i.e. positions, size...) and two child nodes: containers and widgets.

Containers can be associated to a state of the dialog which will make them represent the UI of the interactive system in the associated state. A container can contain other containers and any widgets.

A widget is defined by its attributes. These attributes can be sorted in two categories: native attributes and ontological attributes.

Natives attributes includes identifiers such as “Id”, “equivalentid” which is used in the cloning feature presented in the section Annex 7-7.4 and “name” which is a human readable identifier that can be used to distinguish the different widget. This category also include attributes used to manage the display of the widget in the PANDA prototyping tool such as their position, their size, the “value” for defining the label to display in the widget.

Ontological attributes are attributes that are defined according to an ontology file. Indeed, ontology files are used to specify the widgets that can be created within the PANDA prototyping tool. Thus, each properties of the class defined in the ontology used are stored in sets of ontological properties identified by an “ontologyIRI”. More details on the usage of the ontology within the PANDA prototyping tool will be given in the section Annex 7-7.3.3 and the section Annex 7-7.4.

Regarding the representation of the widgets defined in this model, they are handled separately from the structure of the prototype and their definition is managed by the PANDA prototyping tool. In the current implementation, a set of predefined UI has been defined for containers, images, labels, buttons and text fields. Other widgets are displayed with a placeholder image.

- **Connecting the two models**

The PANDA prototyping tool allow to specify both the dialog and the presentation of an interactive system while providing a visual representation of these two aspects of the application being designed. These two models are loosely tied by using the “containerref” attribute from the dialog model which references a container defined in the presentation model. As for the transitions, they are connected to widgets using the “widgetref” attribute to reference any widget.

The details of the simulator which allow to preview a prototype by combining these two models will be given in the section Annex 7-7.4.2.

7.3. The PANDA ecosystem: a framework for supporting traceability of the design process

7.3.1. Presentation of the framework

The PANDA ecosystem is an evolving framework designed for supporting the traceability of the evolution of the design during the UCD process. Thus, this framework can be used within an iterative process and its activities.

In order to ensure the traceability of the evolution of the design process, the PANDA framework will be used to build a project workspace that will contain the different versions of design artefacts produced and used during the iterations of the design process. This project workspace will then be used as a support for the development of features related to the management and visualization of the design process, of its artefacts and of the evolution of the design.

In order to structure the artefacts within the project workspace, annotations will be used as described in the Chapter 6 to connect the different artefacts between each other and as well as to reference every artefact concerned by design decisions (whether they are used as arguments for a decision or as targets of a decision).

7.3.2. A modular approach

The PANDA ecosystem framework consists in a set of modules that produce and consume artefacts. Each module is dedicated to achieve a set of goals related to the artefact.

Thus, we define a module of this framework as a component that provides a service around a concept which is supported by a tool. This service is expressed through the creation of a new resource or the alteration of an existing resource. A resource is defined as a file or set of files that represents the serialization of the information produced by the module. If a module requires a resource to process, it can be called individually anytime as long as the required resource is available. Two modules can also be linked together if one service use a resource produced by the previous service.

An example of module would be a “Prototyping module” which produces “Prototype resources” based on “Specification resources”. The implementation of this module could be any prototyping tool like “Balsamiq” for instance. In this example, as long as the specification are available, it is possible to build a prototype using this module.

This modular approach allows to separate our ecosystem into small independent parts that can be associated to address every aspects of the design process. On top of that, those parts can be removable if one part is not yet functional or if it is not relevant in the context of usage of the ecosystem provided that this module or the resource produced by the module is not a used by another module.

Another advantage of using a modular ecosystem is that a module can be instantiated with any tool as long as the tool can communicate with the other tools used in the ecosystem.

7.3.3. Modules of the PANDA ecosystem

The PANDA Ecosystem aims to build an environment which can be used to investigate the design process through data collection of discussions, resources produced and resources used in the process. This data will be complementary data collected through a UCD process with observation, interviews and users tests. The PANDA Ecosystem consists at the moment of 5 modules: Ontology, Prototypes, Annotations, Decisions and Versioning. These modules were designed to support a UCD process using prototypes through several iterations.

Each module consist the production of a resource based on the consumption of other resource. The Table Annex 7-1 located at the end of this section summarize the modules of this ecosystem by listing their required resources and the resources they produce.

In order to be able to use this ecosystem and actually produce and collect the different resources, we designed and implemented a first version of some modules for the software prototyping. Each of these implementations are gathered in the CIRCUS framework. This implementation will be used to test and investigating the potential of the ecosystem in a UCD process.

- **The Ontology module**

In the context of the study of the design process, the “Ontology” module will be used to provide a vocabulary for the design and the specification in the PANDA Ecosystem.

This vocabulary is prototype oriented and will define one or several set of classes that will describe:

- Usable widgets to specify the software or prototype
- Properties of those widgets and their type
- Usable events and actions to specify the software or prototype
- Available events and actions supported by the widgets

Using an ontology to describe the design and the specification will help to have a more accurate and rigorous description of the product being designed. Indeed, this ontology will force the design team to use a consistent and common vocabulary to describe the specification and the design. Even if this vocabulary is not detailed or incomplete in the beginning, it can be refined throughout the design process depending on the needs of the project or the available means given by the selected language of implementation for instance.

For the implementation of the PANDA Ecosystem, we used the software “Protégé” to build an OWL file describing a few widgets for web applications, their properties and their supported action/events.

• The Prototype module

In the PANDA Ecosystem, the « Prototype » module consists in producing a prototype that represents a view of the interactive system that is being designed. There is no restriction on the fidelity of this prototype or the part that is prototyped (presentation, dialog or interaction).

In the design process, this module is used to produce artefacts that will represent the solution to be evaluated.

To produce a prototype, this module needs a specification, requirements and users’ needs in order to produce a relevant design. Even if those resources are incorrect, incomplete or underspecified, they are required since they will be completed and corrected over time through iterations of the design process and through refinements over reviews, interviews and tests.

Depending on the prototype uses for the design team, many tools can be used to implement this module. For instance, low fidelity prototypes can be designed using pen and paper for rapid and disposable prototypes or using Balsamiq for interactive prototypes.

This module needs also a grammar and a vocabulary to describe the artefact and make it understandable by others. This grammar can be explicit or implicit depending on the tool used. On tools like Balsamiq, the grammar is already integrated in the tool and is represented by the library of widgets available in a toolbar. As for pen and paper, the grammar is implicit and not restricted since it is possible to draw anything and to assign it an arbitrary meaning. For instance, straight line can be associated to text as a “lorem ipsum”.

In our implementation of the PANDA Ecosystem, we designed a prototyping tool to produce XML based prototypes which use an OWL ontology for its grammar. Those prototypes are divided into two parts: the dialog and the presentation. By allowing designers to edit both the dialog and the presentation, our tool can produce interactive prototypes by combining those two parts and thus be run for testing purposes.

• The Annotation module

The Annotation module is responsible to handle the adding and management of annotations on artefacts and producing a file containing the annotations of the artefact. Thus, the only required resource for this module is to have an artefact to annotate.

This module should allow anyone to express themselves on a document by providing suitable means of expression depending on the intention of the annotator but also by giving them the tools to select and emphasize a fragment of the artefact they want to talk about.

Thus, the Annotation module can be used in the design process as a medium for users to review a prototype or share information on it.

This module should also allow anyone to acknowledge existing annotations on an artefact by displaying them with the fragment they concern and by providing a way to identify who the author of the annotation in order to make the communication, the discussion and argumentation possible between people over the artefact. Depending on the artefact to annotate, the tool to use might differ.

In our implementation of the PANDA Ecosystem, we designed an annotation system presented in the Chapter 7 that can annotate prototypes we created using our tool. This annotation system then produces an XML file describing the annotations and references to the annotated fragment of the prototypes or other relevant files to the annotations.

By using annotations and those references, it is possible to attach any resources to another one. For instance, it is possible to attach the specification to the prototype, to add a task planning, to report a problem or to attach feedback from users gathered during a user test.

After creating annotations on the prototype, they can be displayed on top of the prototype by anyone who have access to the annotation file. The annotation file being an available resource that can be annotated, communication and discussion can then be handled through the annotation of annotations.

- **The Decision module**

In the PANDA ecosystem, the Decision module consists in establishing a set of goals to achieve for the design process and relate each of these goals to the motivations and resources used that lead to their creation.

The set of goals established by this module can be a new set of goals or simply a refinement and update of the current goals for the following iterations of the design process. This set of goals can also be refined as more concrete objectives that can be achieved by carrying out short term objectives. This module does not detail the process of making a decision but instead focuses on the results of the decision making process regarding the progression of the design process.

In the design process, this module is used to make sure that the design process evolves in agreement with the users' need and the specification.

Linking resources with goals defined with this module allow to get a documented trace of reasons that lead to the evolution of the design process.

In order to be able to produce a new or updated set of objectives, this module will require 3 resources. The first resource needed is the decision strategy that will determine how the decisions will be made.

The second resource needed by this module is the set of resources considered by the decision process and decision strategy that will be assigned to each goals and objectives (e.g. requirements, users' need, specification, alternative choices to consider, available resources, business knowledge, previous experience in the domain, costs evaluation and impact of a decision).

The last resource is the current goals and objectives that need to be updated by this module.

While this module has not been implemented yet, the resources considered by the decision process will be the set of annotations on the prototype. Indeed, annotations can note problems in the design that must be addressed and those problems can be solved by modifying the prototype and thus the goals of the implementation process. Though annotations should be filtered and refined in order to extract options of modifications to consider on the prototype and in the design process. An XML file will be used to describe a task list and references other resources from the PANDA project. The task list will be used to monitor the design process and it will be updated carried over through the iterations.

- **The Versioning module**

The role of this module is to regularly store every resource that is provided to it. This versioning module is based on the duplication of the resource at a given time and thus store snapshots of the resources.

In the design process, this module will have to store every document used during the design process throughout the iterations. At the end of the design process, this module should have stored a copy of every resources created and used for each major modification. Doing so will build a history of every prototypes created along with their context of creation, the relationship between prototypes and results of users' tests.

In order to do so, this module require a set of resource to store as well as a repository to store the snapshots. Then, it will produce a snapshot that will be archived for future usage.

In our implementation of this module, the versioning module will be handled by providing a tool support that will assist designers to manage their workspace by duplicating the resources stored in the project workspace and adding a file containing metadata about the version of the snapshot. The hierarchy of the project will then make every snapshot available at any time.

- **Resources used and produced by the modules of the ecosystem**

The modules of the PANDA ecosystem is based on the consumptions of resources to produce another set of resources. The Table Annex 7-1 below lists the different resources required and produced by each of the module of the PANDA ecosystem:

Module	Required resources	Resources produced
Ontology	- (None)	- Ontology file describing classes
Prototype	- A grammar - Specifications - Requirements - Users' need	- Prototypes
Annotation	- An artefact to annotate	- Annotations file related to the artefact
Decision	- Decision method or strategy - Resources used by the decision process - Current goals and objectives	- Updated list of current goals - New/Updated long term objectives - New/Updated short term objectives - A set of resources attached for each goals and objectives
Versioning	- A set of resources to store - A repository to store the snapshots	- Snapshots of the resources

Table Annex 7-1 Table of the modules of the PANDA ecosystem

7.3.4. Combining the modules

In the previous section were listed the different modules of the PANDA ecosystem, their required artefacts and the artefact they produced.

On a micro-level, associating modules allows to use each of the tools on the same set of artefacts and to produce various artefacts relevant for the design process whether the artefacts are related to the interactive system being developed, related to the information on its users, or to the management of the design team and of the process itself.

On a macro-level, this association of module will be used as a solution for ensuring the traceability of the rationale used in the design process and of information exchanged during this process.

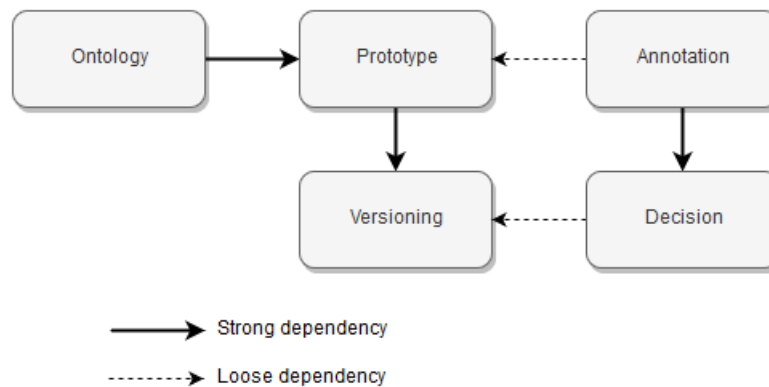


Figure Annex 7-3 Dependency diagram of the PANDA ecosystem

In this framework, modules can either be added, removed or replaced depending on their usefulness in the design project. Thus, the decision system has not been implemented yet but the other modules can still be used during the activities of the UCD process.

In the current implementation of the PANDA ecosystem, the following modules have been developed: Ontology, Prototype and Annotation. These modules have been developed to interact with each other through artefacts and models described in those artefacts. Thus, the implementation of the modules features dependencies between them as illustrated in the figure above. In this figure, thick arrows represent a strong dependency between modules (e.g. Prototype module requires the Ontology module) and dotted arrows represent loose dependencies (compatible). The following parts of this section will be dedicated to the presentation of the combination of the modules of the PANDA ecosystem.

- **Ontology and Prototype modules: Generating a custom library of widget for a flexible prototyping tool**

The Ontology module is providing the Prototype module a grammar to use when designing prototypes. Each ontology defines a set of classes, properties and supported operations by components. Those specifications of components are used to formalize the prototype model produced. Indeed, the Prototype module is used to build models of prototypes based on the composition of components. The properties of these components are defined by the ontology used during the creation of the prototype thus giving the prototype a formal specification.

By adding properties defined in ontologies on components, the prototype module will be able to specify values to sensible properties of the components used in the prototypes. By specifying those properties and clearly defining each component used in the prototype, the tool and the designers gives an unambiguous representation of the software prototyped when required. The precision of the prototype will depend on the details and completeness of the ontology but also on the effort put on the specification components.

The formalization of prototypes using an ontology has been used for the development and the specification of other features within the PANDA ecosystem. Indeed, the properties defined in the ontology has been used for specifying and run automated tests integrated in the prototype as annotations (see 8.3.2 of the Chapter 8), for specifying transformation rules for converting prototypes to other artefacts such as specification document or template of code for the migration of prototype to another level of fidelity.

- **Prototype and Annotation modules: Allowing communication and carrying data over prototypes**

By combining prototyping and annotations mechanisms, stakeholders of the design process will be able to collaborate during the design of the prototypes. Then, this design can be enriched with annotations that will add information like a specification document that will be referenced within the prototype but also with discussions from the collaborator.

Indeed, collaborators and reviewers of the prototype will be able to leave messages, remarks, data for scenarios, and opinion on it and specifying parts of the prototype that need attention by highlighting it. In other words, annotations create a common space for communication and arguing. Although, it should be noted that not sharing annotations on prototypes can also be a reviewing strategy that will limit the influence between reviewers during the process.

On top of that, annotation can also be used to add metadata on the prototype that will be used to give more context to the prototype in the design process.

Thus, annotations will generate data that could otherwise be scattered or not even recorded during the design process and condense this data in an accessible way from the prototype.

- **Decision, Annotation and Prototype module: Linking decision with their related resources using annotations on the prototype**

While the Decision module has not been implemented, it has been defined as a tool support for managing the results of decisions related to the design process and for managing the resources relevant to those results through the different iterations of the design process. For instance, decisions can be related to design choices on a prototype or on the tasks to perform within the design process. The results of these decisions might affect several artefacts within the project workspace.

Thus, annotations could be used as a linking mechanism to achieve the following goals:

- Target the resources used for the decision making for justifying choices made

While annotations can carry over information on prototypes like constraints, requirements or specifications, those annotation can also note problems revealed during users' tests and gather feedback from users. All those data can be used to weight the different modifications or choices of alternative design that will be decided in the decision process.

- Target the resources affected by a decision being made

By assigning targets to decisions results, the impact of a decision is made explicit within the design workspace. Moreover, using annotations allows to structure the artefacts of the project workspace for ensuring the traceability of the impact of decisions.

- Feature the decisions and their results within the representation of an artefact related to them

Every modification on the prototypes can then be linked to this specification and the decisions' task list using annotations. Thus, the ecosystem will feature concrete linking between a modification to a task and between a task and its motivations referenced by the resources used by the decision strategy. This linking will make it possible to trace back every significant modification to get the reasoning behind those modifications. This will also make it possible to ensure that every point described in the specification has been taken into account even if choices and concessions has been made in the design of the prototype.

- **Prototype, Annotation and Decision with the Versioning module: Storing every prototypes and the evolution of the design process**

By adding the Versioning module, the goal is to make sure to store the prototype and every related resource in a consistent state. The versioning will store snapshots of the design process at each milestone defined by the design team, even if all the design of the prototypes stored in those snapshots will not be featured in the final design.

Prototypes carry valuable information on the advancement of the design process and the objectives to achieve by designers at the moment of its creation. Indeed, every prototype reflects a vision of how the final product should look like, how it should work or how the final product should behave given an understanding of the users' need and of the requirements even if those understanding and requirements were incomplete or inaccurate. Indeed, accepted and rejected prototypes can give a vision of constraints of the project or requirements not met in the design and thus a glimpse of the rationale of the design team. Moreover, prototypes can be used either for exploring acceptable design or for restricting design choices.

By collecting every prototype produced regardless their matching with the final requirements and specifications, we want to be able to reconstruct and retrace the different visions and understanding of the project and replace them in the history of the design process and in their context along with the other resources used in this process. We also expect to be able to trace back the different designs envisaged during the process and know the reasons as to why they were not selected or how the final solution has come to be.

However, a differentiation has to be made between accepted prototypes and rejected one as well as which prototype or part of prototype reflects a right vision of the final product or a wrong vision. Annotations can be used to add metadata on those prototypes and put some context on them regarding the requirements and the specifications.

7.3.5. Integration of the modules of the PANDA ecosystem within the UCD process

The following figure is a representation of the interactions between the modules through the artefacts produced and of their integration within the activities of the UCD process.

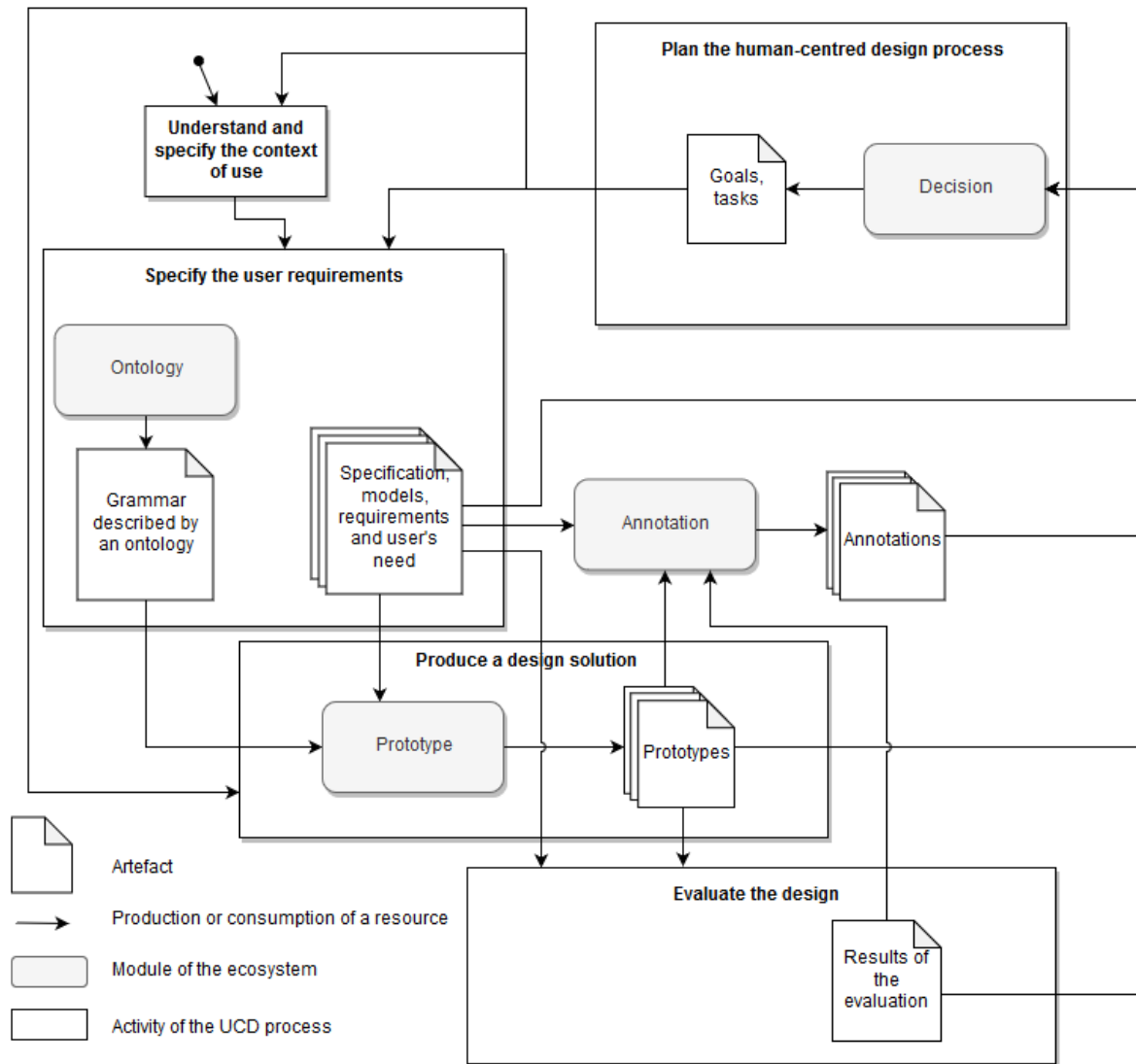


Figure Annex 7-4 Representation of the PANDA ecosystem with their artefacts within the UCD process as described in the Figure 2-15

The first use of the PANDA framework can be made during the “specification of the user requirements” activity with the Ontology module. This usage is particular due to the nature of the module. Indeed, early iterations of the design process tends to focus on the quick and successive iterations of rapid prototypes using paper which are opposed to formalized prototypes. Thus, several solutions can be adopted for using the PANDA ecosystem in early iterations: either reuse an existing ontology or iterate on the creation of the ontology.

After the creation of the ontology, it can be used to generate a palette within the Prototype module during the activity “Produce a design solution”. Thus, the designers can design a prototype based on the specifications and requirements defined in the previous activity using the PANDA prototyping tool.

The prototypes can be used for the activity “Evaluate the design” which will produce results of evaluations. These results can be analyzed and integrated within the understanding of the context of use of the design.

The specifications, the prototypes and the results of the evaluation can be annotated anytime during the design process for various purpose (see section 3.4.2) using the Annotation module.

At the end of an iteration, the decision process can be started with the activity “Plan the human-centered design process” and the Decision module. This module uses the different artefacts produced in the other activities to produce and update the current goals and tasks of the design process.

7.4. Creating and previewing a PANDA prototype

7.4.1. Creating a prototype using PANDA

The creation of a prototype using PANDA is done in 4 steps:

- Specification of the prototype behavior using a state machine
- Initialization of components that will be used to build the prototype using an ontology
- Specification of the presentation of each state specified in the previous step
- Specification of transitions between the states to make the prototype interactive in the “Preview mode”

The prototype produced following these steps allow to have an interactive medium fidelity prototype specifying both the dialog and the presentation of the interactive system.

- **Specification of the Prototype Behavior**

PANDA allows the description of both the presentation and the behavior of user interface prototypes. The description of the behavior is driven by an executable dialog that can be also used to simulate the execution of prototype. Figure Annex 7-1 illustrate how automata are visualized in PANDA. A state describes the different widgets that are displayed by the system like in Figure 8-1.D and is represented by a rectangle with a grey header. The transitions between two states are represented with arrow as shown in the Figure Annex 7-1 and are specified with the triplet $\frac{\text{Event} / (\text{Conditions})}{\text{Actions}}$.

For this case study, we have defined first version of the behavior of the application featuring three states: the “Find flight” state which will be used to specify the information of the flight and which is the initial state of the application, the “Choose flight” state will list the flights matching the criteria specified by the user and allow him to choose one flight and the “Validation” state which will prompt the user to validate its selection, propose him to add options to his purchase and proceed to the payment of the flight.

- **Initialization of components**

As many other prototyping tools, PANDA is based on a library of widgets that will be drag and dropped to compose the prototype similarly to the prototyping tool Balsamiq [102]. While there is other interaction techniques to create prototypes such as hand-draw (ex. Denim [103]), combination of images import and hotspots (ex. Marvel [99]), and implementation through code, the usage of a library was more adapted for this study. On the one hand, the creation of prototypes using widgets allows to define the interface more accurately than importing one image since it structures the presentations using components supporting customization of properties. Moreover, widgets are flexible and it is possible to use widgets to import images representing components of the presentation. On the other hand, adding a drawing recognition system to support hand-drawn prototypes is not necessary nor relevant for the support of this study on prototyping tools and the activities of the design process.

One of the notable features of PANDA is that the library of widgets is dynamic and automatically instantiated from an ontology specific to an application domain. The underlying idea of this feature is that depending on the application domain targeted by the design team, the widgets needed to build the prototype will be different. Using an ontology for the creation of prototype has several advantages.

For instance, it allows to add properties and a semantic meaning to each element of the prototype. The association of elements and the value of their properties can also be controlled by rules specified in the ontology. For instance, a “Button” component can be located in a container such as a window or within a form for “Submit” buttons of a website. Thus, the properties associated with the elements can be formally tested.

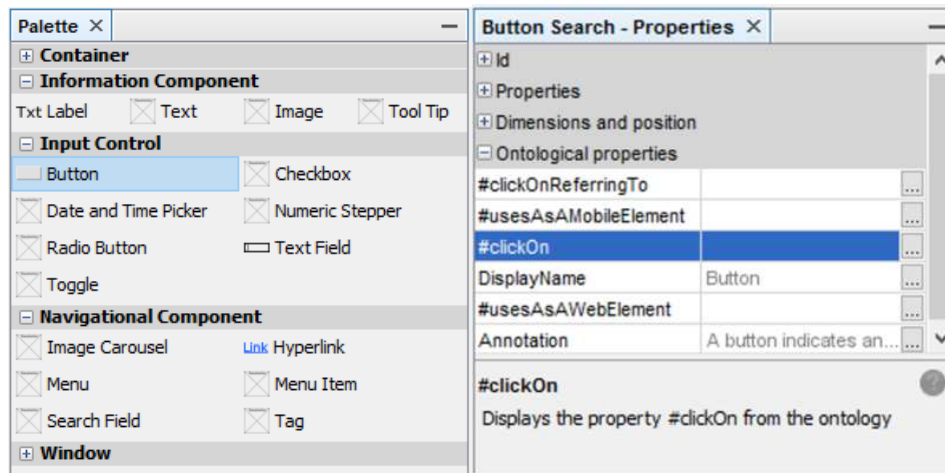


Figure Annex 7-5 Palette of widgets instantiated in PANDA and properties of a "Button" widget defined by the ontology

Thus, before composing the presentation of the states of the PANDA prototype, it is necessary to initialize the library of widgets by importing an ontology file. In the current implementation, PANDA is only supporting ontologies based on the OWL format. Once the ontology is loaded, the palette is automatically generated depending on the classes and properties described in the ontology as illustrated in the Figure Annex 7-5 above.

In the PANDA architecture, the properties of a widget are sorted into two distinct sets. The first set is the “common” properties. This set includes properties used by the prototyping tool to manage the widgets such as an identifier, the name of the widget, their type, their positions and their size in the drawing area. The second set of properties called “Ontological properties” which is defined according to the ontology. Depending on the property, a value can be assigned while other properties are defining the widget with their presence (e.g. the “#clickOn” property specify that the widget support the mouse click interaction).

While the ontology is defining the properties supported by a widget, their representation is not defined. In the current implementation of the PANDA prototyping tool, the representation of the widgets is predefined for a set of widgets. Other widgets are represented using a placeholder.

In this case study, an ontology describing widgets for web applications has been loaded. After initializing the library of component, the creation of the prototype can be done by dropping elements from the palette located in the figure 6.B (such as a button or a text field) to the editing area located in the Figure 8-1.D.

While the palette is dynamically generated for the specification of the presentation of the application, some widgets are always available in PANDA regardless of the ontology loaded since they are independent from the application domain of the prototype. These widgets are: the state for defining the dialog of the prototype and the annotations which can be applied to any artefacts.

- **Specification of the presentation**

Once the dialog has been defined and the palette initialized, the presentation can be specified in the PANDA prototyping tool. The specification of the presentation is based on the dialog. Indeed, each state of the dialog will represent what the user will see from the user interface of the application. Thus, the widgets initialized in the previous step have to be instantiated in the different states of the application. Once dropped in the drawing area, the widgets can be customized by editing their properties.

Find flight (Initial State)

Search a flight

Menu

Flight only

Flight & Hotel

Flight & car

From: Empty

To: Empty

Passenger: Empty

Depart: Empty

Return: Empty

Search

Choose Flights

Depart	Arrives	Flight operator	Economy	Premium Economy	Business Class
10:45 21 Apr	13:25 21 Apr	OpenSkies BA8005	€1420	€1560	€4736
11:15 21 Apr	13:30 21 Apr	American Airlines BA1535	€1423	Not available	€4738

Start again

Continue

Figure Annex 7-6 Specification of the presentation for the two states of the prototype

Regarding the duplication of widgets between states of the prototype, the PANDA prototyping tool is featuring a cloning system. This feature allows to create a clone of either a widget or a group of widgets. However, cloning a widget should be distinguished from copying it in PANDA. Indeed, a clone of a widget is keeping a reference to every other instances of the widgets using a shared ID system. This shared ID unique for each group of clones is used to identify each widgets or group of widgets equivalents across the different states of the prototype. The following Figure Annex 7-7 highlight two widgets sharing the same clone ID.

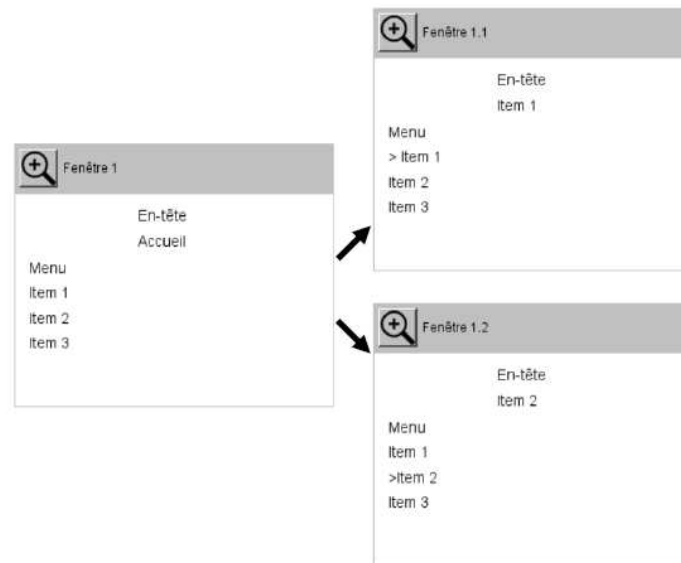


Figure Annex 7-7 Presentation of the cloning feature

While these clones appear to be similar, each clone has their own values for each properties which reflect their evolution through the different states of the interactive system.

Two features exploiting this cloning mechanism have been initiated. The first feature is the automatic deductions of transformations between two states of the dialog by calculating the differences between the widgets and their properties displayed in each state. The second feature is the spreading of the values of the properties of a widget to its clones. This second feature can be used for instance to edit the position of a menu on every states of the prototype.

The cloning feature is similar to the definition of symbol in Balsamiq which allow to create group of widgets that can be instantiated across the different pages of the prototype. However, this implementation is necessary for the development of further features as described previously.

Overall, the goal of this cloning mechanism is to make the prototype closer to a formal specification of the interactive system. Indeed, the UI of the prototype will show that there are components that are reused and edited across the different states of the prototype.

- **Specification of transitions between the states which combine the presentation with the dialog**

After defining each state of the prototype and each UI of the different states, the next step in the creation of the prototype is to associate the behavior of the prototype with the presentation.

To do so, each transition defined in the dialog has to be assigned to one widget of the presentation. In the Figure Annex 7-8 below, the transitions have been assigned to the “Search” button and the “Start again” button.

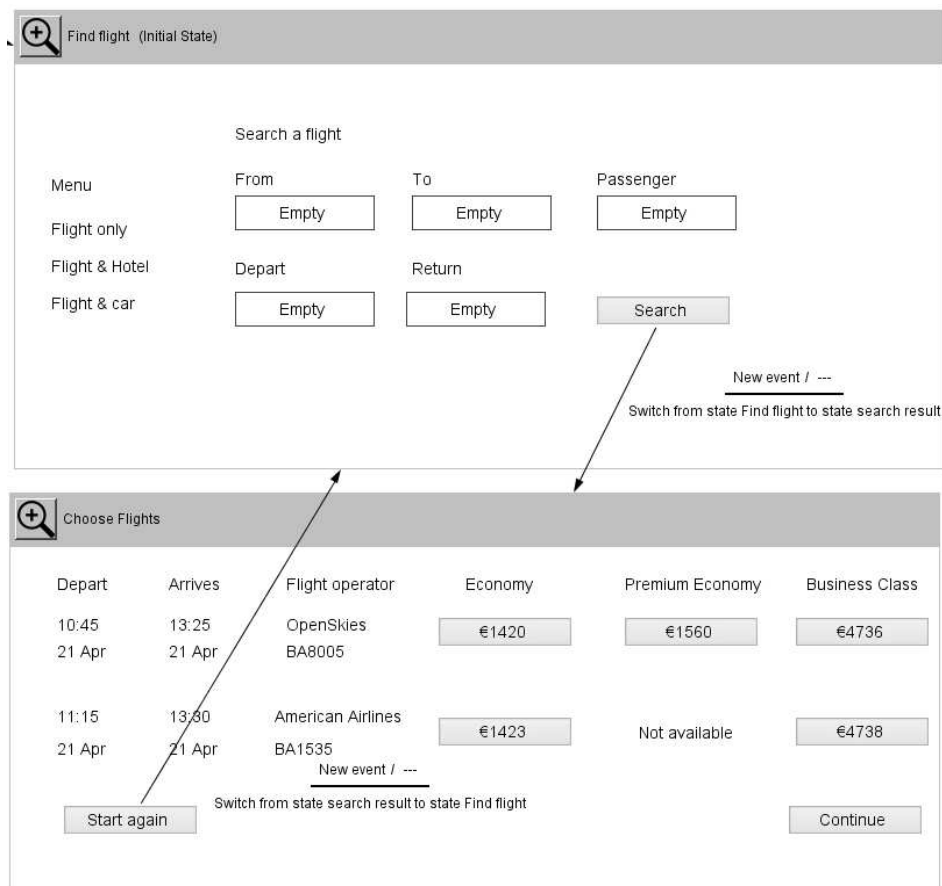


Figure Annex 7-8 Prototype combining the presentation and the dialog

This association is registered in the dialog model as the optional attribute called “widgetref” of the “transition” element from the dialog model (see 7.2). This attribute will store the id of the widget from the presentation model that can trigger the transition. This binding allows to connect the two models without imposing a strong dependency between the models.

7.4.2. Previewing the PANDA prototype

After having created the dialog and the presentation of the prototype and after having connected them, the prototype can be previewed. This preview will allow the user to interact with the prototype by navigating through the different states defined in the prototype.

- **Presentation of the preview system**

The preview starts with the display of the initial state of the dialog which is indicated by its connection to the start marker represented with a black spot as shown in the Figure Annex 7-9 below.

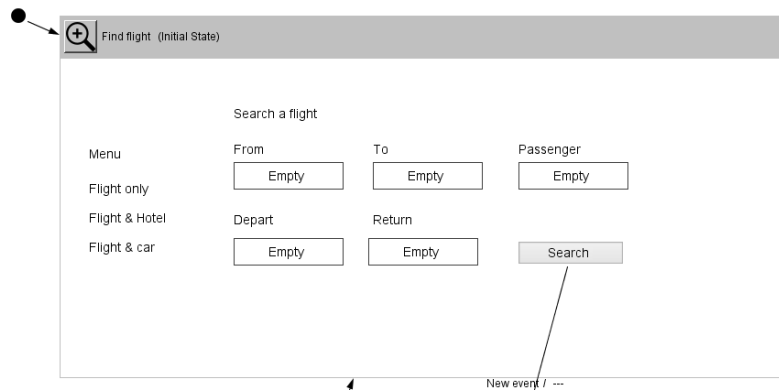


Figure Annex 7-9 Initial state of the prototype

In the preview mode, the current state of the prototype is outlined by a red stroke and the different widgets of the presentation contained in this active state can be clicked. When clicking on a widget that is associated to a transition, the active state is switched to the state targeted by the transition. This switching allows to simulate a navigation through the prototype.

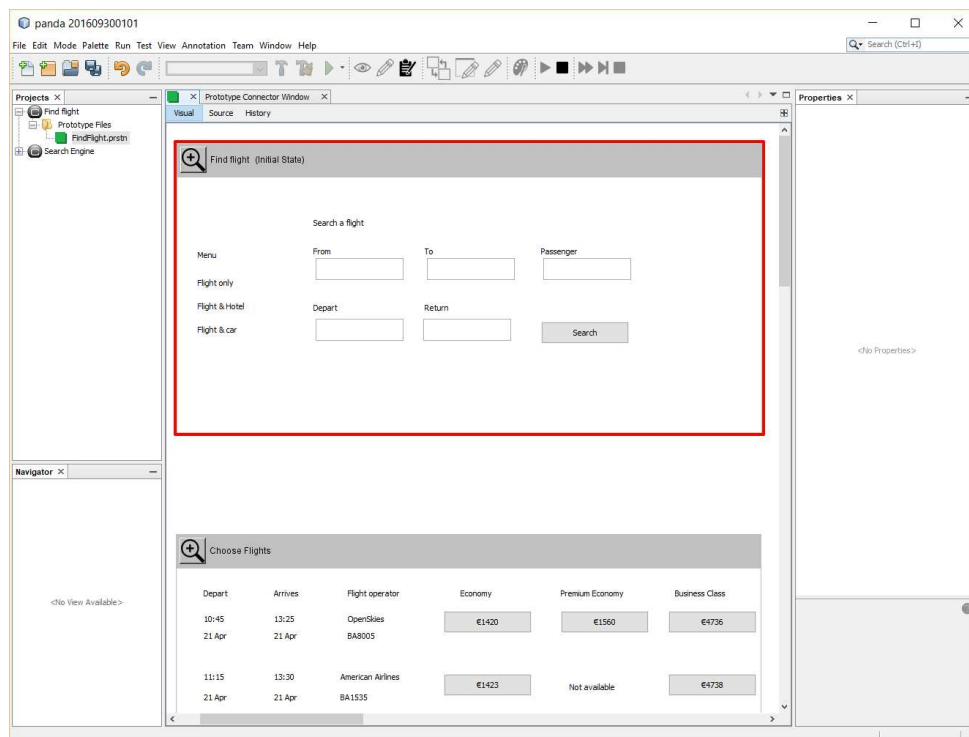


Figure Annex 7-10 Active state outlined with a red stroke

In the case study, when the state “Find flight” is active, clicking on the button “Search” will activate and display the state “Choose flights” according to the transition defined earlier. Though, the interactions in PANDA are on a high level of abstraction since it does not process data and the transitions handle only left-click events on widgets.

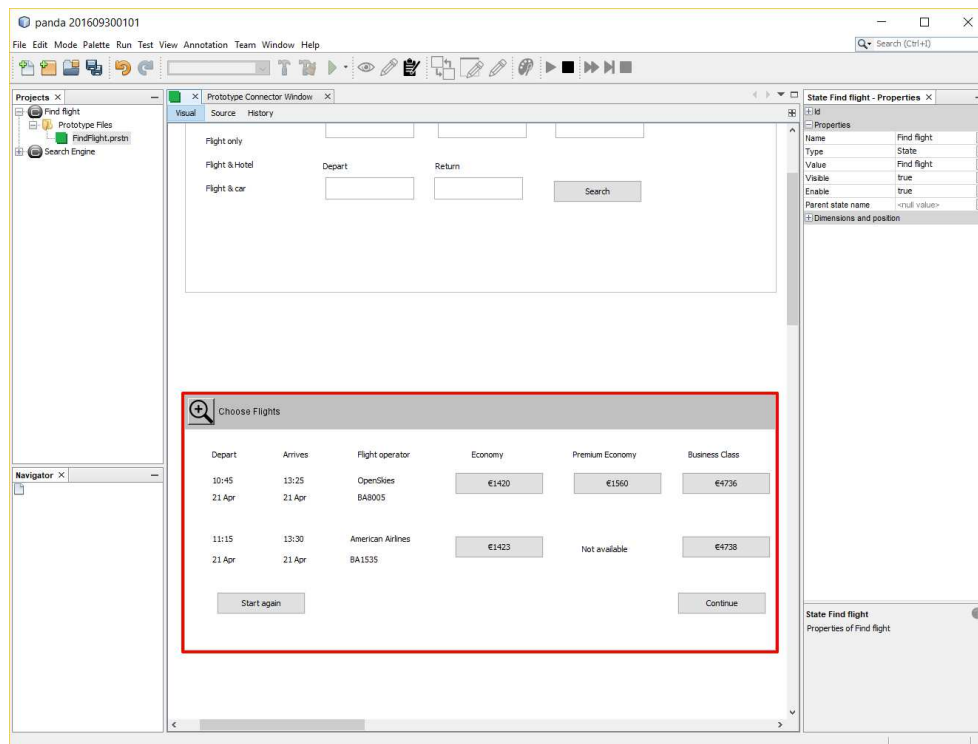


Figure Annex 7-11 “Choose flight” state as the active state after clicking on the “Search” button

- **Presentation of the simulator**

The preview system allowing the interactivity of the prototype is ensured by an integrated simulator. This simulator is executing the dialog model describing the behavior of the prototype and is reacting to user interactions on the presentation according to the transitions defined in the prototype.

The communication between the dialog and the presentation and thus interactivity of the prototype is ensured by an event-based mechanism. In the dialog, transitions between states are defined by five properties: the source state, the targeted state, the conditions to fulfill to enable this transition, the events triggering the transition, and the actions to perform.

When using the preview mode of the PANDA prototyping tool, the simulator is initialized with the initial state as the starting point. When interacting with the presentation of the prototypes, events are triggered depending on the actions of the user. The simulator checks if any accessible transition from the current state can be triggered by the event raised by the user. If a transition has been found, events for each action associated with the transition from the dialog are raised to notify any model prototyping the actions.

As a complement to the simulation of the prototype, it is possible to develop and integrate to the PANDA ecosystem an engine dedicated to the execution of scripts or mock-up that could simulate the behavior of a back-end. This backend can be perceived as another dimension that can be used for prototyping and specifying the interactive system. For instance, the engine could be based on the list of conditions and actions specified in the dialog model. After retrieving the conditions list of the transition being crossed in the simulator of the prototype, the engine could fetch a script or a mock associated with the name of the condition and return a corresponding value that can be used by the simulator to either accept or reject the transition. In case the condition has been accepted, the engine could proceed similarly for handling the actions triggered by the transition. An advantage of defining

this backend is that depending on its implementation, it can be reused and integrated in the final product.

Annex 8. Presentation of the e-CitizTM framework

Summary

This annex presents the details of the e-CitizTM framework.

The first section of this chapter will give an overview of the e-CitizTM framework which is a platform for creating the e-service applications that will be edited with “eazlyTM”. This overview features the presentation of the different actors involved in the framework, a presentation of the design process with the e-Citiz StudioTM and a presentation of two models created in this studio.

The second section will introduce the limits of the e-CitizTM framework as well as the two consecutive projects to tackle these limits: “Process 2.0” and “eazlyTM”.

8.1. Introduction

This annex presents and details the e-CitizTM framework. This annex is complementary to the Chapter 10.

The section 8.2 provides an overview of the e-CitizTM framework which consists in an editor for generating e-CitizTM applications from a “Business Process Model and Notation” standard (BPMN) as well as the design process associated to the generation of those applications. The section 8.3 gives a brief presentation of the different models of the e-CitizTM application. Lastly, the 8.4 exposes the limits of the e-CitizTM framework and the necessity of an evolution of this framework for tackling these limits.

8.2. Overview of the e-CitizTM framework

The e-CitizTM framework consists in an editor called “e-Citiz StudioTM” which allows its users called “Business Analyst” to generate automatically a e-service application from the BPMN without any prior knowledge aside the business processes and the data manipulated within these processes. The generated application can then be deployed on a server to be available for its use by any users such as the citizens of the local communities or an agent from the back-office of the local communities. This overview is illustrated in grey in the Figure Annex 8-1 below:

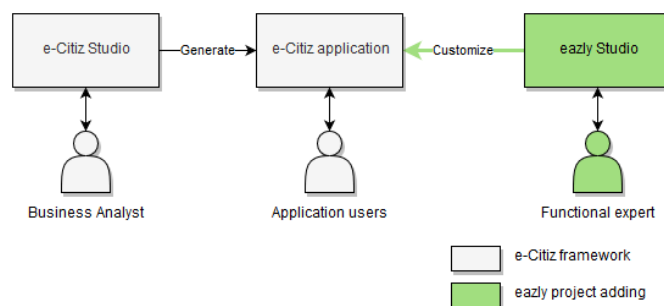


Figure Annex 8-1 e-CitizTM application environment

The green parts of this figure represent the addition of the eazlyTM project to the e-CitizTM framework which will be detailed in the section 10.3. This addition consists in a new editor called “eazly StudioTM” that will be used by a “Functional Expert” who belongs to the organization represented by the customer. This studio allows its users to customize a deployed e-Citiz application using a Wysiwyg interface. At the moment, the customizable parts of the applications includes forms and the pages displaying information. Thus, it is possible to add new fields to an existing form so that users can input new information. These information are then stored in the e-Citiz application database and attached to the user of the application the same way as any data manipulated by the base application. These information can then also be edited and displayed as any existing data from the application.

8.2.1. The actors of the e-Citiz™ framework

As mentioned in the previous section, it is possible to identify three types of actors within the e-Citiz™ framework: the “Business Analyst”, the “Functional Expert” and the “Application user”.

The role of the Business Analyst is to build a BPMN model for each services of the future e-service application and to generate the e-Citiz™ application from these models using the e-Citiz Studio™. For that, the Business Analyst must inquire from the Functional Expert and any relevant people about these services such as the actors involved, the procedure, the order of tasks to perform, or the data required to perform those tasks.

The role of the Functional Expert is to represent the Customer of the application and is characterized by his mastery of the services to dematerialize. He is in charge of validating the application generated by the Business Analyst.

The role of Application user is to achieve his tasks through the tasks provided by the e-Citiz™ application. For instance, an end-user can request an ID card through an e-Citiz™ application.

8.2.2. The iterative design process for the design of an e-Citiz™ application

The development of an e-Citiz™ application is done using the “e-Citiz Studio™”. This studio is a proprietary software that allow its users to model one or several BPMNs which describes processes and the actors of those process. From those BPMNs, the studio will generate an application which can be deployed on a server.

The design process used for e-Citiz™ applications is composed of two main phases as illustrated in the Figure Annex 8-2.

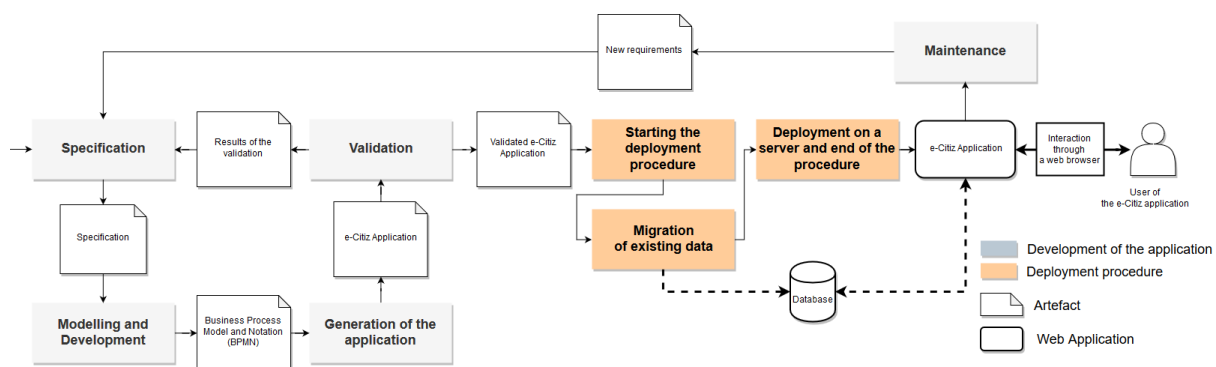


Figure Annex 8-2 e-Citiz™ design process

The first phase is the development of the application represented in grey on the figure which corresponds to the iterative design process. This phase consists in generating an e-Citiz™ application that will match to the existing business process to dematerialize.

The second phase is the deployment of the application on the production server represented in beige on the figure. The goal of this phase is to provide the application to its end-users. This phase is linear and critical since it is affecting the availability of the service to the users even if some clustering techniques can minimize the downtime. At the end of this phase, it is possible to go back to the first phase and proceed to a new iteration of the e-Citiz™ design process.

8.2.3. The development of the e-Citiz™ application

The development of the e-Citiz™ application can be divided in five steps as illustrated in the Figure Annex 8-2.

- **Specification**

The first step is the “Specification” which is performed by the Business Analyst. In this activity, the Business Analyst discusses with the Customer, collects information, investigates the context, analyzes the services and inquires about the stakeholders of the business process to dematerialize in order to produce the specifications and the requirements artefacts. These artefacts will then be used during the next step: the “Modelling and Development”.

- **Modelling and Development**

During this step, a Business Analyst is responsible of modeling in the e-Citiz Studio™ one or several BPMNs that match with the existing processes that need to be dematerialized. The modeling of the BPMN also include the definition of a data structure manipulated during the activities of the BPMN (see 3.4).

To illustrate this step, we will present an example of service called “Arrival procedure” which consists in the registration of the arrival of a “Newcomer” that is processed by an “Instructor”. The Figure Annex 8-3 below show the corresponding BPMN modeled using the “e-Citiz Studio™”.

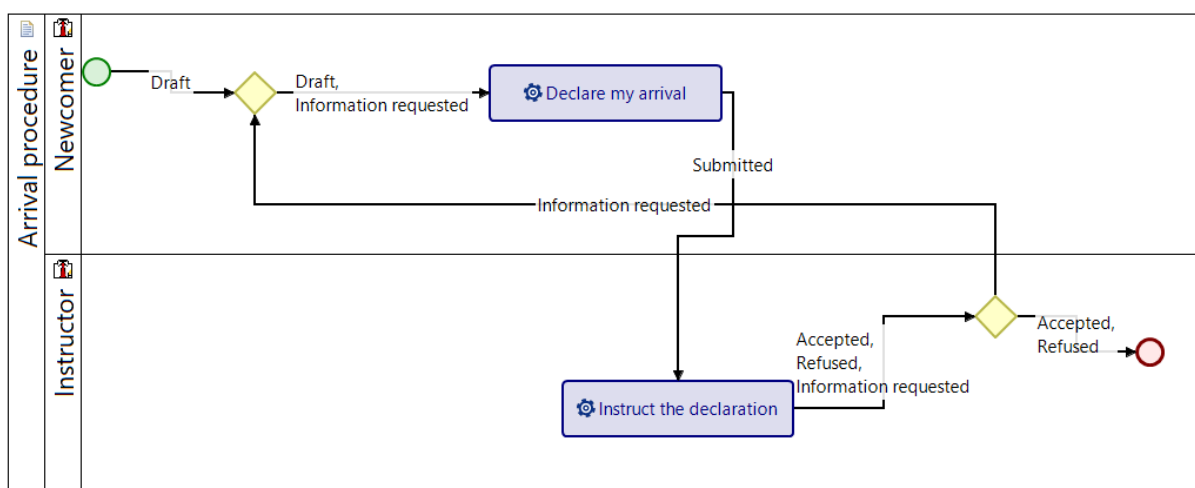


Figure Annex 8-3 Example of a BPMN in the e-Citiz Studio™

This model consists of the representation of the actors involved in the process and the sequence of tasks performed by them. The details of the definition of the actors and of the definition of the tasks are represented by other models built by the Business Analyst using the e-Citiz Studio™. For the sake of simplicity, these models will not be detailed.

In the BPMN, the actors involved in the process are represented by the lanes of the model. Thus, each lane is defining the involvement of a different role of end-users of the e-Citiz™ application for this “Arrival procedure”.

The business process is defined by the oriented graph representing the sequence of tasks. Each task is represented by a blue rectangle and the lane in which they are positioned indicates which actor performs the task. The sequence starts with the initial status represented by the green circle and ends with the final status represented by the red circle. The arrows connecting the tasks are defining the sequence of tasks as well as the status of a process. Indeed, each new use of the service will create an instance of this graph. Thus, the first step of the registration is done by the “Newcomer” who will create an instance of this BPMN using the e-Citiz™ application. The Newcomer will fill in a form to enter the details of his arrival in the first task “Declare my arrival”. Once this task is done, his file is

marked as “Submitted” and can be handled by an instructor. The Instructor who will take the file will be able to start the task “Instruct the declaration” in which he will be able to either accept, refuse or ask for more information. Depending on the choice of the Instructor, the file is either archived or retransmitted to the Newcomer.

- **Generation of the application**

Once the modeling is done, the next step of the development of the e-Citiz™ application is the “Generation of the application”. In this step, the Business Analyst generates the e-Citiz application using the e-Citiz Studio™. This application can be considered as a high-fidelity prototype that can be deployed on a test server.

The Figure Annex 8-4 below is a screenshot of the e-Citiz™ application generated by the studio. This page corresponds to the page in which the Newcomer declares his arrival.

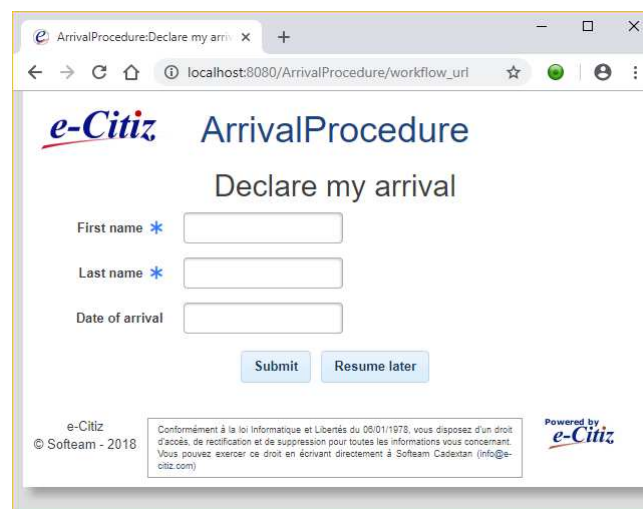


Figure Annex 8-4 Screenshot of an e-Citiz™ application

- **Validation**

After the generation of the application, it can be validated by the Functional Expert. Depending on the results of this validation, the application can either be refined by iterating on the development process

During this step, the prototype can either be refined in the e-Citiz Studio™ by the Business Analyst or be ready to be deployed on a production server. If the application has been validated, the deployment procedure can be started.

- **Maintenance**

After the deployment of the application, it can be maintained and updated by the Business Analyst. For that, the new requirements of the application are integrated into the specifications. After that, a new cycle of iterations of the development of the application can be started with the existing BPMN while a version of the application is still in production. The Business Analyst specifies the modifications to perform, edits the application using the e-Citiz Studio™ and generates a new version of the application. Once the new version of the application is generated, it can be validated and used to replace the former version of the application in the server through the deployment procedure.

Modifications that can be done during the maintenance range from minor changes such as editing labels to fix spelling mistake to more important modifications such as the edition of existing or new BPMN.

- **The deployment of the application**

The deployment of the application consists in three different steps: the start of the deployment procedure, the migration of existing data if necessary, and the deployment of the application on the server followed by the return of service.

First, the deployment procedure is started which include a node switching procedure to avoid the interruption of the service during the update of the application. After that, if the Business Analyst has edited the data structure associated to the application in the e-Citiz Studio™, the existing data are migrated to make them compatible with the new data structure defined by the Business Analyst. Then, the application can be deployed on a server (either a test, validation or production server) and the procedure can be ended. Thus, any end-users will be able to access to the services provided by the new version of the application.

8.3. The different models of the application

8.3.1. The e-Citiz™ model

The e-Citiz™ application is generated with the e-Citiz Studio™ based on the model produced by the Business Analyst. This model includes the BPMN in which the different tasks are defined for each actor, the definition of business rules and a data structure for the data that is manipulated or produced during the business process. In the description of a task is also included the different data to prompt or to display to the user. The content of the application is generated based on the tasks descriptions for the content of the webpages. The navigation and the processing are determined by the BPMN and any custom code implemented by the Business Analyst.

The maintenance of this model implies that a new version of the application must be generated and deployed. Thus, it is necessary to start a new iteration of the design process.

8.3.2. The user data structure

In the e-Citiz Studio™, the Business analyst defines several structures of data manipulated in the BPMN. The structures of data includes the specification of the different data that are manipulated in the application, their type, and their relationship with other data. He also defines business rules that can define the possible value of those data, defines specific processing depending on the value of those data, and specifies which data are used in each part of the BPMN and each tasks of the actors of the application.

The Business analyst does not manipulate data owned by a user but only the structure of the data (i.e. the metadata). The Figure Annex 8-5 show an example of data structure for the service “Arrival Procedure”. In this example, this service will manage three types of data: two text attributes (“First name” and “Last name”) and a date attribute (“Date of arrival”).

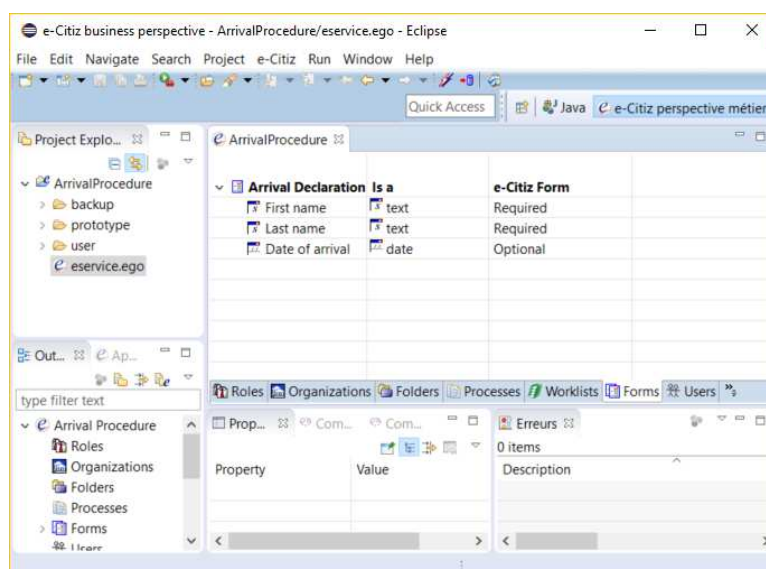


Figure Annex 8-5 Data Structure of a declaration of arrival

Once the e-Citiz™ application is generated and deployed on a server, users of the e-service will be able to log in and benefit the services provided by the application by instantiating processes of the BPMN. Each instance of a BPMN created by a user of the e-Citiz™ application will contains data that will follow the structure defined by the Business analyst. Those data are filled, consulted and edited by the different actors of the process during the activities defined in the BPMN as the process progress.

Each version of a BPMN might define a different version of the data structure. Thus, each update of a BPMN might require a migration of the data stored in the databases of the application.

8.4. Conclusions on the e-Citiz™ framework

8.4.1. The lack of the e-Citiz™ framework

Overall, the e-Citiz™ framework associated with the design process presented in the section 8.2.2 is an industrial solution for generating e-service applications based on a specification. While this solution is satisfying regarding the robustness of the applications generated and regarding the suitability of the applications for providing the services, the use of this framework in various projects showed that e-Citiz's customers have the need to refine and customize the applications they order to better match the services they provide to their end-users. However, the e-Citiz™ framework is not suited for answering this need in a usable manner (i.e. in an efficient, effective and satisfactory manner): this framework have two major drawbacks for covering this need.

The first drawback is the lack of flexibility regarding the update of the application. Indeed, any update of the application requires to go through every steps of the design process and to involve every actors of this process. This constraint is slowing down the updating process of the application.

The second drawback of this framework is the accessibility of the framework for novice users. Indeed, the usage of the e-Citiz Studio™ requires a good understanding of the BMPN notation as well as the analytic skills for designing and architecting the models such as the data model. These required skills limit the use of the e-Citiz Studio™ to designers.

From these observations, two successive new projects were initiated: the "Process 2.0" project and the "eazly™ project".

8.4.2. The “Process 2.0” project

The goal of the “Process 2.0” project was to provide a new studio to Functional Experts for generating e-CitizTM applications. Thus, this studio would be used as a replacement of the e-Citiz StudioTM. To facilitate its usage for generating the applications, the use of BPMN and models have been dropped in favor to the use of a Wysiwyg editor for designing processes and the web pages of the application.

At the end of the project, two results have been obtained: a set of recommendations based on a study on the future users of the studio (namely Functional Experts) and a prototype of a studio for the design of e-service application.

- **Study of users and their context**

As the part of the “Process 2.0” project, a series of individual interviews has been performed with 17 different subjects from March 1st to the 5th 2010 and from March 29th to April 1st 2010. Those subjects originated from two different communities: “Région Limousin” and “Région Midi-Pyrénées”.

The selection of the 17 subjects was made to reflect a large variety of profile in diversity of position, of function, qualification, skills, knowledge and expertise.

The study revealed two distinct profiles:

- A **“Technical”** profile which gathers users who master the technical and technological aspects of the projects, digitalized business process and services they are involved in. This profile displays a variety of skills, knowledge and master its technical environment, language and tools depending on their professional career and their teaching. Those profiles are involved in their project during its definition, the feasibility evaluation, the development, the estimation, maintenance and evolution.
- A **“Functional”** profile or **“Business”** profile gathers users who will express conceptual needs or the need of the creation of a teleprocedure

The targeted user of the “Process 2.0” project are the users who match the “Functional” profile.

Each of those profiles are also divided into three categories in the results of this study:

- **“Decision Maker”** or **“Supervisors”** who initiates a project and ensures its progression according to the time and cost constraints
- **“Designers”** who are dedicated to the design of the project, gathers data and monitors the project
- **“Instructors”** who are processing the files of the teleprocedure

Overall, the study which used the 5M model (Man, Machine, Medium, Method, and Material) noted that the users were encountering most of the difficulties on the following points regarding the dematerialization of a service: Man, Method and Machine. The Table Annex 8- give a synthesis of the main results found in this analysis.

Domain	Problem identified
Man	Lack of technical and computer science knowledge for “Functional” profiles
	Lack of formation, experience and mastery of tools
	Difficulty to describe a need, write a specification
	Lack of mastery of the vocabulary and language used for the specification of their need
	Lack of analytical skills and reflection for the design of an application
Method	Lack of method and support which is noted as the most recurrent problem

Machine	The cumbersome aspect and lack of flexibility in the maintenance, update, evolution, enrichment of tools, application and services
----------------	--

Table Annex 8-1 Main results of the 5M model analysis

The study has also identified the following needs for the users of the new studio being developed:

- Be able to describe and specify a process with the least technical knowledge possible
- Collaborate with multiple users with different skills

- **Prototype of a studio for the design of e-service applications**

This prototype called “eazly™” (shown in the Figure Annex 8-6) allows to manage and to create “Appeaz” which is the name given to the applications created with this tool. The name “eazly™” will later be reused for the following project presented in the section 8.4.3.

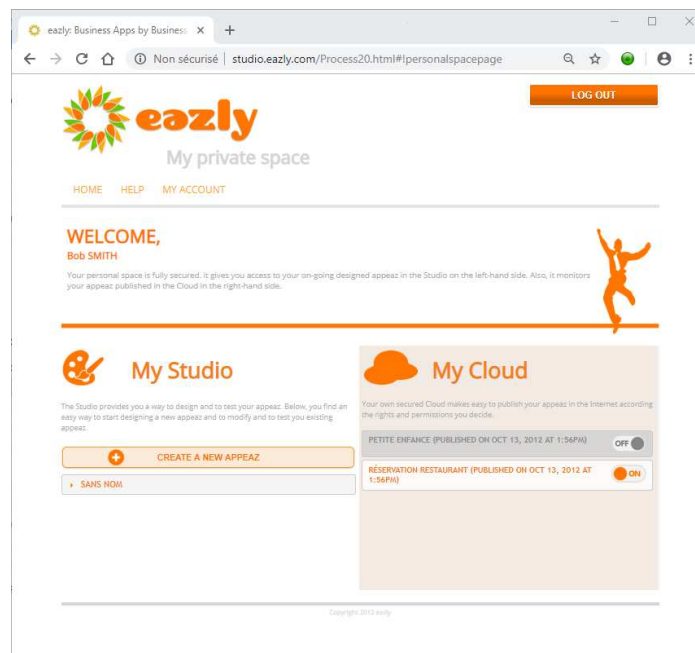


Figure Annex 8-6 Homepage of eazly™ after logging in

After logging in, the user can create new Appeaz to access the eazly Studio™ by clicking on the button on the left side “Create a new Appeaz”. Doing so will open the studio as shown in the Figure Annex 8-7 in which the user will be prompted to create the first page of the Appeaz by specifying the title of the page and by specifying the name and the role of the actor who will be able to see this page (i.e. “Visitor” or Anonymous User, “User”, “Manager”).

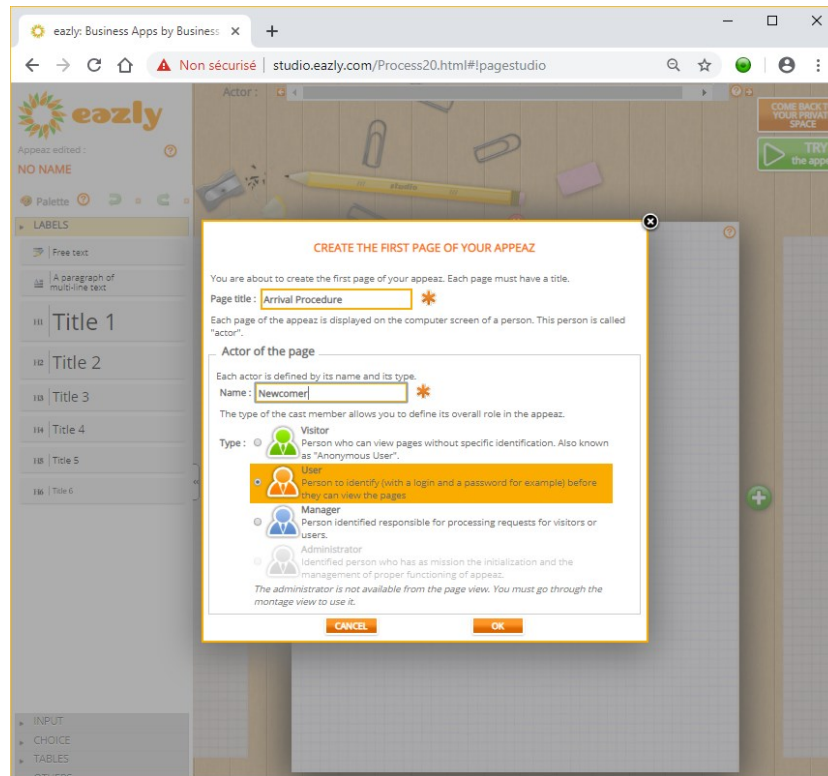


Figure Annex 8-7 Creation of the first page of the Appeaz

After the creation of the first page, the studio shows the UI dedicated to the specification of the e-service application which is illustrated in the Figure Annex 8-8. The upper part of the figure gives a representation of the sequence of the tasks associated to the service. Each task is performed by the user specified by the designer. The left part of the prototype is the palette which gives a library of widgets that can be dragged and dropped to build the UI of the task of the service in the center of the figure. Each field dropped can be edited (e.g. the label associated with the field or the parameter such as the “mandatory” property represented as a star on the right side of the field).

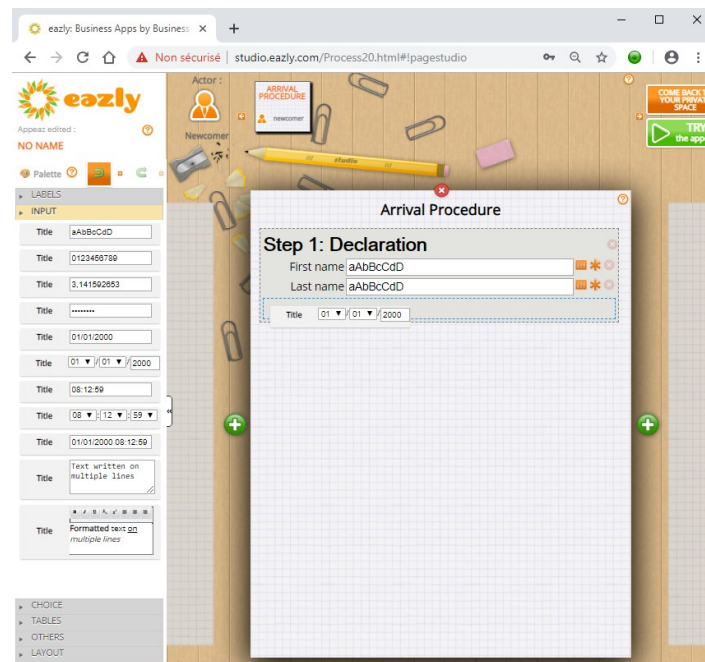


Figure Annex 8-8 Specification of the page

Once the first page has been specified, other pages can be added by clicking on the “(+)” buttons. Doing so will create a new page as shown in the upper part of the Figure Annex 8-9.

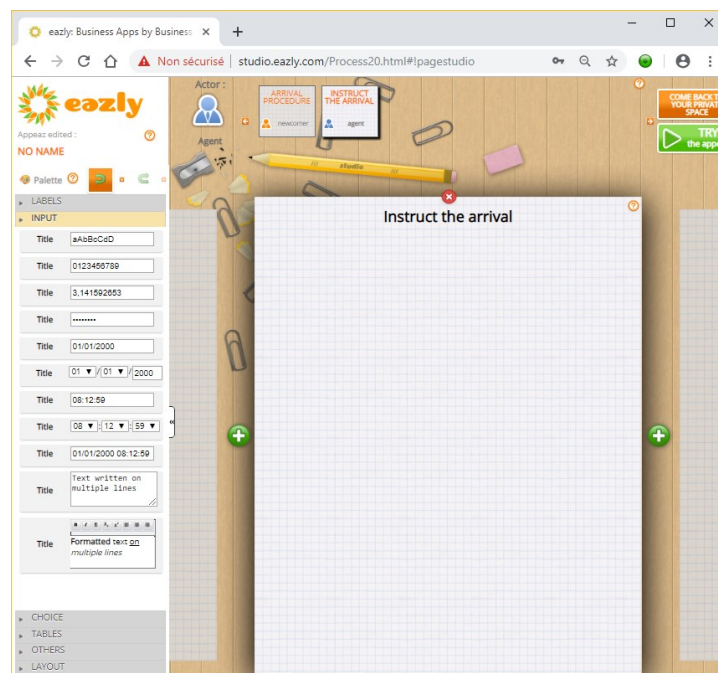


Figure Annex 8-9 Adding a new page

8.4.3. Toward the “eazly™” project

From the results of the “Process 2.0” project, a second project called “eazly™” has been started as a follow-up for empowering the involvement of Functional Experts in the design of their e-Citiz™ application.

The “eazly™” project has a different approach for tackling the limits of the e-Citiz™ framework identified in the section 8.4.1. Indeed, this project aims to provide an industrial tool that can be commercialized for the customization and fine-tuning of e-Citiz™ applications. For that, the development of this product has been defined around two milestones.

The first milestone is to provide a tool for the customization of an existing e-Citiz™ application. This choice has been motivated by the goal of providing a compatible solution with existing e-Citiz™ applications ordered by several customers. Thus, the product of this project can be considered as an extension of the e-Citiz™ framework which at the moment possess numerous features for designing and tuning reliable e-service applications.

The second milestone of this project is to provide the features required for designing a new e-Citiz™ application from a set of templates or from scratch.

Bibliography

1. Teeuwen, M. (2018). Practices of Appropriation: Writing in the Margin. In E. Kwakkel & R. Thomson (Eds.), *The European Book in the Twelfth Century (Cambridge Studies in Medieval Literature)*, pp. 139-156). Cambridge: Cambridge University Press.
doi:10.1017/9781316480205.010
2. Daniele Metilli, "French Tachygraphic Notes in a 1504 copy of Homer's Odyssey",
<http://metilli.com/files/Metilli-Odyssey-2014-05-05.pdf>
3. WikiArt - Visual Art Encyclopedia. Available at: <https://www.wikiart.org/en/leonardo-da-vinci/drawings-of-water-lifting-devices> (last visit April 17th, 2019).
4. Maristella Agosti, Giorgetta Bonfiglio-Dosio, and Nicola Ferro. 2007. A historical and contemporary study on annotations to derive key features for systems design. *Int. J. Digit. Libr.* 8, 1 (October 2007), 1-19. DOI=<http://dx.doi.org/10.1007/s00799-007-0010-0>
5. Marisela Gutierrez Lopez, Gustavo Rovelo, Mieke Haesen, Kris Luyten, Karin Coninx. Capturing Design Decision Rationale with Decision Cards. *INTERACT* (1) 2017: 463-482
6. Amir M. Naghsh, Andy Dearden, and Mehmet B. Özcan. 2005. Investigating annotation in electronic paper-prototypes. In *Proceedings of the 12th international conference on Interactive Systems: design, specification, and verification (DSVIS'05)*, Stephen W. Gilroy and Michael D. Harrison (Eds.). Springer-Verlag, Berlin, Heidelberg, 90-101.
DOI=http://dx.doi.org/10.1007/11752707_8
7. Jackson, M. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Addison-Wesley Professional; 1 edition (September 2, 1995) 228 pages.
8. Winckler, Marco. "Engineering Interactive Systems across application domains." HDR, Université Toulouse III -Paul Sabatier, July 2016.
9. Martinie De Almeida, Célia. « Une approche à base de modèles synergiques pour la prise en compte simultanée de l'utilisabilité, la fiabilité et l'opérabilité des systèmes interactifs critiques. » Ph.D. dissertation, Université Toulouse III -Paul Sabatier, 2011.
10. Royce, W. (1970). *Managing the development of large software systems: Concepts and techniques*. WESCON technical paper.
11. McDermid, J., & Ripken, K. (1983). Life cycle support in the Ada environment. *ACM SIGAda Ada Letters*, III(1).
12. Christophe Kolski, Pierre Loslever, An HCI-Enriched Model for Supporting Human-Machine Systems Design and Evaluation, *IFAC Proceedings Volumes*, Volume 31, Issue 26, 1998, Pages 419-424, ISSN 1474-6670, [https://doi.org/10.1016/S1474-6670\(17\)40129-7](https://doi.org/10.1016/S1474-6670(17)40129-7).
(<http://www.sciencedirect.com/science/article/pii/S1474667017401297>)
13. Boehm, B. (1986). A spiral model of software development and enhancement. *SIGSOFT Software Engineering Notes*, XI(4, pp.14-24).
14. Kruchten, P. (2004). *The Rational Unified Process: an Introduction* (Vol. 3rd edition). Boston, MA: Pearson Education, Inc.
15. Object Management Group. (2011). *OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4*.
16. Cockburn, A. (2002). *Agile Software Development*. Boston, MA: Addison-Wesley.

17. Shwaber, K., & Beedle, M. (2002). *Agile Software Development with SCRUM*. Upper Saddle River, NJ: Prentice-Hall.
18. Beck, K. (1999). *Extreme Programming Explained*. Palo Alto, CA: Addison-Wesley.
19. Hartson, H., & Hix, D. (1989). Human-computer interface development: concepts and systems for its management. *ACM Computing Surveys*, 21(1).
20. Curtis, B., & Hefley, B. (1994). A WIMP no more: the maturing of user interface engineering. *Interactions*, 1(1).
21. Collins, D. (1995). *Designing Object-Oriented user interfaces*. Readwoods City, CA: Benjamin/Cummings Publishing, Inc.
22. Rauterberg, M. (1992). An Iterative-Cyclic Software Process Model. *International Conference on Software Engineering and Knowledge Engineering*. Capri, Italie: IEEE.
23. Gulliksen, J., & Goransson, B. (2003). Usability Design: Integrating User Centered System Design in the Software Development Process. *IFIP International Conference on Human-Computer Interaction, INTERACT*. Zurich, Suisse: Springer
24. L. L. Constantine and L. A. D. Lockwood, "Usage-centered engineering for Web applications," in *IEEE Software*, vol. 19, no. 2, pp. 42-50, March-April 2002.
doi:10.1109/52.991331 keywords: {Internet; software engineering; user centred design; user interfaces; Web pages; software engineering; model-driven programming; user interface design; usability; software development; Internet; usage centered engineering; Usability; Application software; Software engineering; User interfaces; Graphics; User centered design; Prototypes; Programming profession; Marketing and sales; Software prototyping},
URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=991331&isnumber=21378>
25. ISO 9241-210 2018. Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems. Standard. International Organization for Standardization, Geneva, CH.
26. Iyad Khaddam, Hanaa Barakat, Jean Vanderdonckt: Enactment of User Interface Development Methods in Software Life Cycles. RoCHI 2016: 26-35
27. Catherine C. Marshall. 1997. Annotation: from paper books to the digital library. In *Proceedings of the second ACM international conference on Digital libraries (DL '97)*. ACM, New York, NY, USA, 131-140. DOI=10.1145/263690.263806
<http://doi.acm.org/10.1145/263690.263806>
28. Catherine C. Marshall. 1998. Toward an ecology of hypertext annotation. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space---structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems (HYPERTEXT '98)*. ACM, New York, NY, USA, 40-49.
DOI=<http://dx.doi.org/10.1145/276627.276632>
29. Bringay S., Barry C., Charley J., Annotations: A new type of document in the Electronic Health Record. Paper presented at the 2nd International Conference on Document Research and Development in Sciences, arts and business: DOCAM 2004, University of California, Berkeley, Etats-Unis, octobre 2004.
30. Winckler, Marco & Palanque, Philippe & Farenc, Christelle & Pimenta, Marcelo. (2019). Who does what with whom in Web Development?.
31. José Kahan and Marja-Ritta Koivunen. 2001. Annotea: an open RDF infrastructure for shared Web annotations. In *Proceedings of the 10th international conference on World Wide Web*

- (WWW '01). ACM, New York, NY, USA, 623-632.
DOI=<http://dx.doi.org/10.1145/371920.372166>
32. Robert Sanderson, Paolo Ciccarese, Herbert Van de Sompel, 2013. Open Annotation Data Model, Community Draft. Available at: <http://www.openannotation.org/spec/core/>
 33. Sanderson, Robert, Paolo Ciccarese, and Benjamin Young, eds. 2017. "Web Annotation Data Model." W3C Recommendation, February 23. Accessed 2019-04-18.
 34. Sanderson, Robert, Paolo Ciccarese, and Benjamin Young, eds. 2017. "Web Annotation Vocabulary." W3C Recommendation, February 23. Accessed 2019-04-18.
 35. Devopedia. 2018. "Web Annotation." Version 2, June 25. Accessed 2019-04-19.
<https://devopedia.org/web-annotation>
 36. Annotation Studio. 2017. Accessed 2019-04-19. <https://www.annotationstudio.org/>
 37. Hypothes.is. Accessed 2019-04-19. <https://web.hypothes.is/>
 38. Annotator. Accessed 2019-04-19. <http://annotatorjs.org/>
 39. Plugins - Annotator. Accessed 2019-04-19. <http://annotatorjs.org/plugins/index.html>
 40. Max G. Van Kleek, Wolfe Styke, m.c. schraefel, and David Karger. 2011. Finders/keepers: a longitudinal study of people managing information scraps in a micro-note tool. In Proceedings of the 2011 annual conference on Human factors in computing systems (CHI '11). ACM, New York, NY, USA, 2907-2916. Version: Author's final manuscript
 41. UxPin retrieved from <https://www.uxpin.com/>
 42. Craig S. Tashman and W. Keith Edwards. 2011. LiquidText: a flexible, multitouch environment to support active reading. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11). ACM, New York, NY, USA, 3285-3294. DOI: <https://doi.org/10.1145/1978942.1979430>
 43. Lichan Hong, Ed H. Chi, Raluca Budiu, Peter Pirolli, and Les Nelson. 2008. SparTag.us: a low cost tagging system for foraging of web content. In Proceedings of the working conference on Advanced visual interfaces (AVI '08). ACM, New York, NY, USA, 65-72. DOI: <https://doi.org/10.1145/1385569.1385582>
 44. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization Jeffrey Heer, Fernanda Viégas, Martin Wattenberg ACM Human Factors in Computing Systems (CHI), 1029–1038, 2007
 45. Capadislis S., Guy A., Verborgh R., Lange C., Auer S., Berners-Lee T. (2017) Decentralised Authoring, Annotations and Notifications for a Read-Write Web with dokieli. In: Cabot J., De Virgilio R., Torlone R. (eds) Web Engineering. ICWE 2017. Lecture Notes in Computer Science, vol 10360. Springer, Cham
 46. Protoshare retrieved from <http://www.protoshare.com/>
 47. Pidoco retrieved from <https://pidoco.com/en>
 48. Sanderson, Robert, eds. 2017. "Web Annotation Protocol." W3C Recommendation, February 23. Accessed 2019-04-18.
 49. Agosti M. and Ferro N. 2007. A formal model of annotations of digital content. ACM Trans. Inform. Syst. 26, 1, Article 3 (November 2007), 57 pages. DOI = 10.1145/1292591.1292594
<http://doi.acm.org/10.1145/1292591.1292594>
 50. Li, C & McMahon, Chris & Newnes, Linda. (2009). Annotation in design processes: Classification of approaches. DS 58-8: Proceedings of ICED 09, the 17th International Conference on Engineering Design. 251-262.
 51. Zacklad, Manuel & Lewkowicz, Myriam & Boujut, Jean-François & Darses, Françoise & Et, Françoise & Détienne, Françoise. (2003). Formes et gestion des annotations numériques collectives en ingénierie collaborative.
 52. Manuel Zacklad. Annotation : attention, association, contribution. Annotations dans les Documents pour l'Action, Hermes science publications, pp.29-46, 2007. <sid 00180781>

53. Levy, D.M. (1997) "I read the news today oh boy: reading and attention in the digital library" in Proceedings of Digital Libraries '97, Philadelphia, Pennsylvania (July 23-26, 1997)
54. Gaëlle Lortal. Médiatiser l'annotation pour une herméneutique numérique : AnT&CoW, un collectif pour une coopération via l'annotation de documents numériques. Autre [cs.OH]. Université de Technologie de Troyes, 2006. Français. <tel-00136042>
55. José Kahan and Marja-Ritta Koivunen. 2001. Annotea: an open RDF infrastructure for shared Web annotations. In Proceedings of the 10th international conference on World Wide Web (WWW '01). ACM, New York, NY, USA, 623-632. DOI=<http://dx.doi.org/10.1145/371920.372166>
56. Lortal G., Lewkowicz M., Todirascu-Courtier A., 2005, Annotation: Textual Media for Cooperation, in Proceedings of Annotation for Cooperation Workshop November 24-25th (p.41-50)
57. Amir M. Naghsh, Andy Dearden, and Mehmet B. Özcan. 2005. Investigating annotation in electronic paper-prototypes. In Proceedings of the 12th international conference on Interactive Systems: design, specification, and verification (DSVIS'05), Stephen W. Gilroy and Michael D. Harrison (Eds.). Springer-Verlag, Berlin, Heidelberg, 90-101. DOI=http://dx.doi.org/10.1007/11752707_8
58. Virbel Jacques. Annotation dynamique et lecture expérimentale : vers une nouvelle glose ?. In: Littérature, n°96, 1994. Informatique et littérature. pp. 91-105. doi : 10.3406/litt.1994.2354 http://www.persee.fr/doc/litt_0047-4800_1994_num_96_4_2354
59. Shipman, F.M., McCall, R.J.: Integrating Different Perspectives on Design Rationale : Supporting the Emergence of Design Rationale from Design Communication. Artif. Intell. Eng. Des. Anal. Manuf. 11, 141–154 (1997)
60. R. Gruber, Thomas & Russell, Daniel. (1994). Generative Design Rationale: Beyond the Record and Replay Paradigm.
61. Jean-François Boujut, Françoise Darses, Sylvie Guibert. Etude des annotations en situation collaborative de conception mécanique. Pascal Salembier, Manuel Zacklad. Annotations dans les documents pour l'action, Lavoisier, pp.127-152, 2006. <hal-00290132>
62. Weick K.E. (1979). The Social Psychology of organizing, New York, Random House
63. Craig J. Sutherland, Andrew Luxton-Reilly, Beryl Plimmer, *Freeform digital ink annotations in electronic documents: A systematic mapping study*. Computers & Graphics, Volume 55, April 2016, Pages 1-20.
64. Oren, Eyal & Möller, Knud & Scerri, Simon & Handschuh, Siegfried & Sintek, Michael. (2006). What are semantic annotations.
65. Pagare, Reena & Shinde, Anita. (2012). A Study on Image Annotation Techniques. International Journal of Computer Applications. 37. 42-45. 10.5120/4616-6295.
66. Marisela Gutierrez Lopez, Gustavo Roveló Ruiz, Kris Luyten, Mieke Haesen, and Karin Coninx. 2018. Re-thinking Traceability: A Prototype to Record and Revisit the Evolution of Design Artefacts. In *Proceedings of the 2018 ACM Conference on Supporting Groupwork* (GROUP '18). ACM, New York, NY, USA, 196-208. DOI: <https://doi.org/10.1145/3148330.3148334>
67. Thiago Silva, Jean-Luc Hak, Marco Winckler. A Review of Milestones in the History of GUI Prototyping Tools. Workshop on User Experience and User-Centered Development Processes

- 2015 (IFIP WG 13.2). INTERACT 2015 Adjunct Proceedings vol:22. p:267-279. University of Bamberg Press.
68. Silva, T. R., Hak, J.-L., Winckler, M. & Nicolas, O. (2017). A Comparative Study of Milestones for Featuring GUI Prototyping Tools. *Journal of Software Engineering and Applications*, 10 (06), pp. 564-589. DOI: <http://doi.org/10.4236/jsea.2017.106031>. (Silva et al., 2017)
 69. Garcia, Andrei & Da Silva, Tiago & Silveira, Milene. (2017). Artifacts for Agile User-Centered Design: A Systematic Mapping. 10.24251/HICSS.2017.706.
 70. Diigo retrieved from <https://www.diigo.com/>
 71. Anozilla retrieved from <http://annozilla.mozdev.org/>
 72. InVision retrieved from <https://www.invisionapp.com/>
 73. Ninjamock retrieved from <https://ninjamock.com/>
 74. Notism retrieved from <https://www.notism.io/>
 75. HotGloo retrieved from <https://www.hotgloo.com/>
 76. Moqups retrieved from <https://moqups.com/>
 77. Ken Hinckley, Xiaojun Bi, Michel Pahud, and Bill Buxton. 2012. Informal information gathering techniques for active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 1893-1896. DOI: <https://doi.org/10.1145/2207676.2208327>
 78. Ponga retrieved from <https://www.ponga.com/>
 79. Landay, J. A., & Myers, B. A. (1995, May). Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 43-50). ACM Press/Addison-Wesley Publishing Co.
 80. Authorea retrieved from <https://www.authorea.com/>
 81. Protonotes retrieved from <http://www.protonotes.com/>
 82. Micheline Elias and Anastasia Bezerianos. 2012. Annotating BI visualization dashboards: needs & challenges. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 1641-1650. DOI: <http://dx.doi.org/10.1145/2207676.2208288>
 83. LucidChart retrieved from <https://www.lucidchart.com/>
 84. JustInMind retrieved from <https://www.justinmind.com/>
 85. MockFlow retrieved from <https://mockflow.com/>
 86. Claudia Müller-Birn, Tina Klüwer, André Breitenfeld, Alexa Schlegel, and Lukas Benedix. 2015. neonion: Combining Human and Machine Intelligence. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing* (CSCW'15 Companion). ACM, New York, NY, USA, 223-226. DOI: <https://doi.org/10.1145/2685553.2699012>
 87. Paolo Ciccarese, Marco Ocana, Tim Clark (2012) DOME0: a web-based tool for semantic annotation of online documents. *Bio-Ontologies* 2011. <http://bio-ontologies.knowledgeblog.org/297>

88. Di Donato, Francesca & Morbidoni, Christian & Fonda, Simone & Piccioli, Alessio & Grassi, Marco & Nucci, Michele. (2013). Semantic annotation with Pundit: A case study and a practical demonstration. 10.1145/2517978.2517995.
89. Amaya Home Page. (2018, March 14). Retrieved from <https://www.w3.org/Amaya/>
90. Best prototyping & wireframe tool with inline requirements management, iRise. (2018, March 14). Retrieved from <https://www.irise.com/>
91. Concept.Ly retrieved from <http://www.concept.ly/>
92. Robert S. Fish, Robert E. Kraut, and Mary D. P. Leland. 1988. Quilt: a collaborative tool for cooperative writing. In Proceedings of the ACM SIGOIS and IEEECS TC-OA 1988 conference on Office information systems (COCS '88), Robert B. Allen (Ed.). ACM, New York, NY, USA, 30-37. DOI=<http://dx.doi.org/10.1145/45410.45414>
93. Cacao retrieved from <https://cacao.com/>
94. iPlotz retrieved from <https://iplotz.com/>
95. Michel Beaudouin-Lafon and Wendy Mackay. 2002. Prototyping tools and techniques. In *The human-computer interaction handbook*, Julie A. Jacko and Andrew Sears (Eds.). L. Erlbaum Associates Inc., Hillsdale, NJ, USA 1006-1031
96. Izza Naqvi, Study and Analyses of tools for annotating artifacts used to develop interactive systems. (2018)
97. <http://gs.statcounter.com/os-market-share/mobile/worldwide>
98. Prott, <https://play.google.com/store/apps/details?id=com.prottapp.android>
99. Marvel, <https://marvelapp.com/>
100. <https://play.google.com/store/apps/details?id=net.dobrien.quickproto>
101. Uren V., Cimiano P., Iria J., Handschuh S., Vargas-Vera M., Motta E., Ciravegna F. Semantic annotation for knowledge management: requirements and a survey of the state of the art Web Semant. Sci. Serv. Agents World Wide Web, 4 (2006), pp. 14-28
102. Balsamiq retrieved from <https://balsamiq.com/>
103. Lin, J., Newman, M. W., Hong, J. I., & Landay, J. A. (2000, April). DENIM: finding a tighter fit between tools and practice for Web site design. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 510-517). ACM.
104. Eric Barboni, Jean-François Ladry, David Navarre, Philippe Palanque, and Marco Winckler. 2010. Beyond modelling: an integrated environment supporting co-execution of tasks and systems models. In Proceedings of the 2nd ACM EICS '10. ACM, New York, NY, USA, 165-174
105. Silva, Thiago & Hak, Jean-Luc & Winckler, Marco. (2016). Testing Prototypes and Final User Interfaces Through an Ontological Perspective for Behavior-Driven Development. 9856. 86-107. 10.1007/978-3-319-44902-9_7.
106. Silva, Thiago & Hak, Jean-Luc & Winckler, Marco. (2017). A Formal Ontology for Describing Interactive Behaviors and Supporting Automated Testing on User Interfaces. International Journal of Semantic Computing. 11. 513-539. 10.1142/S1793351X17400219.

107. Rocha Silva, Thiago and Hak, Jean-Luc and Winckler, Marco Antonio. An Approach for Multi-Artifact Testing Through an Ontological Perspective for Behavior-Driven Development. (2016) Complex Systems Informatics and Modeling Quarterly, 7. 81-107. ISSN 2255-9922
108. Jean-Luc Hak, Marco Antonio Winckler, David Navarre. PANDA: prototyping using annotation and decision analysis. 8th ACM SIGCHI conference Engineering Interactive Computing Systems (EICS2016), Jun 2016, Brussels, Belgium. EICS '16: Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 171-176, 2016. <hal-01712526>
109. North, D. (2016). What's in a story? <http://dannorth.net/whats-in-a-story/>.
110. Cohn, M. (2004). User stories applied: For agile software development. Addison-Wesley Professional.
111. T. R. Silva, J. Hak and M. Winckler, "A Behavior-Based Ontology for Supporting Automated Assessment of Interactive Systems," 2017 IEEE 11th International Conference on Semantic Computing (ICSC), San Diego, CA, 2017, pp. 250-257. doi: 10.1109/ICSC.2017.73
112. Kenton O 'Hara, Sellen A. A comparison of reading paper and on-line documents. In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI 97 '. New York, NY, USA: ACM; 1997. p. 335–42
113. Diaper, D. & Stanton, N. A. (eds.) The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, 2004. 650 pages.
114. Winckler, M., Vanderdonckt, J., Trindade, F., Stanciulescu, A. Cascading Dialog Modeling with UsiXML. International Workshop on the Design, Verification and Specification of Interactive Systems (DSVIS'2008). Kingston, Ontario, Canada, July 16-18 2008. Springer LNCS 5136. pp. 121-135.
115. Marisela Gutierrez Lopez, Gustavo Rovel, Mieke Haesen, Kris Luyten, Karin Coninx. Capturing Design Decision Rationale with Decision Cards. INTERACT (1) 2017: 463-482
116. Célia Martinie, David Navarre, Philippe A. Palanque, Camille Fayollas. A generic tool-supported framework for coupling task models and interactive applications. EICS 2015: 244-253.
117. Martinie C., Palanque P., Winckler M.. Structuring and Composition Mechanisms to Address Scalability Issues in Task Models. IFIP TC13 Human Computer Interaction 2011 (INTERACT). p:134-152. Springer-Verlag.
118. Navarre D., Palanque P., Ladry J-F. & Barboni E. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact., 16(4), 18:1–18:56. 2009
119. Jean-Luc Hak, Marco Winckler, David Navarre:PANDA: prototyping using annotation and decision analysis. EICS 2016: 171-176
120. Dominique L. Scapin & J. M. Christian Bastien (1997) Ergonomic criteria for evaluating the ergonomic quality of interactive systems, Behaviour & Information Technology, 16:4-5, 220-231, DOI: [10.1080/014492997119806](https://doi.org/10.1080/014492997119806)