Processus d'ingénierie des exigences dans un environnment à base de modèles selon les normes automobiles

Morayo Adedjouma

DELPHI

Requirements engineering process according to automotive standards in a model-driven framework

UNIVERSITY OF PARIS SUD XI

Graduate School
ÉCOLE DOCTORALE D'INFORMATIQUE DE PARIS SUD

To achieve the degree of
DOCTEUR EN INFORMATIQUE

Company
DELPHI FRANCE

Research Lab
CEA LIST - LISE

By

MORAYO ADEDJOUMA

July 12, 2012

Jury Members

Guy VIDAL-NAQUET	President	Professor, Université Paris Sud
Walter SCHÖN	Reviewer	Professor, Université de Technologie Compiègne
Mark VAN DEN BRAND	Reviewer	Professor, Eindhoven University of Technology
François TERRIER	Examinator	Professor, CEA LIST (Director)
Hubert DUBOIS	Examinator	Researcher, CEA LIST (Supervisor)
Claude MARCHÉ	Examinator	Senior Scientist, INRIA
Sophie DUPUY-CHESSA	Examinator	Lecturer, Université Pierre Mendès France
Nicolas DEROUBAIX	Examinator	Engineer, DELPHI

Acknowledgements

My gratitude goes firstly to Mr. François Terrier, who has directed this thesis, and to Hubert Dubois, my thesis supervisor, for all their attention and to the resources they have provided over the last three years. I would particularly like to thank Hubert Dubois. Without his help, this work would never have been started, carried out, and finished. His constructive input, which was so generous both in human and scientific terms and his confidence in my capacity contributed to helping me finish this research project which spanned three years. I would like to thank both of them for their excellent skills and valuable advice which helped me carry out this work. Secondly I would like to thank my first supervisor at DELPHI, Kamel Maaziz. I would like to thank him for giving me this opportunity. His supervision and support has been a privilege.

I would like to thank all my CEA LIST colleagues that I have met during the last three years and before. Their help and the positive atmosphere they created have been invaluable. It is impossible to thank everyone individually; however, I especially want to thank Fadoi Lakhal for her encouragement and advice, and also Vincent Lorenzo. I am also grateful to Agnès Lanusse and Arnaud Lapitre for their constructive feedback about this research project, highlighting subjects requiring further thought.

In addition, I also thank some past and present colleagues at DELPHI, including André Tchuinkam Fotso and Philippe Natchoo for all their interesting discussions, and Huong Lan Trinh for his sympathy and good humor. I would like to thank Pierre-Yves Gueguen for his fruitful contribution throughout this thesis and for helping me to acquire an industrial view which has been so worthwhile in my work. I express my gratitude to Raphael Pezet for his interest in my work and to Yann Lelong for his warm welcome. Particular acknowledgement is due to Wojciech Machnik for his help during the pilot application evaluation. Many thanks to Hao Hu, Serge Aradj and Boina Ali Nourdine. Their support has provided a very important foundation to this thesis, although they may be unaware of their contribution.

I thank all the colleagues involved in the (international) research and co-operation projects in which I participated and with whom I could share ideas. I am very grateful for their co-operation.

I am also very grateful to Prof. Guy Vidal-Nacquet, in agreeing to chair my jury, and reviewers Mark Van Der Brand and Walter Schön for their valuable comments and suggestions. I also thank the others members of my jury: Claude Marché and Sophie Dupuy-Chessa for their time and their interest in my work. I finally thank Nicolas Deroubaix from DELPHI.

Mahuna Akplogan was the person mainly responsible for motivating me to start this Ph.D. project: I am grateful for its support, insights, co-operation during the past years… and so more.

Finally, thanks to my family and my friends who, despite sometimes their silly questions and their incomprehension about my work, nevertheless provided me with an important stimulus with their inexpert interest. Very special thanks go to my parents. I would like to express my gratitude by dedicating this thesis to them.

August 1st, 2012.

Abstract

Requirements engineering process according to automotive standards in a model-driven framework

Abstract

The embedded safety-critical systems industry is facing an exponential increase in the complexity and variety of systems and devices while costs, performance in terms of intelligence, features, capacities and time to market are constantly challenged. The main objective for automotive manufacturers and suppliers is now becoming the control of quality and the dependability of embedded and mechatronic systems. The existence of internationally recognized standards such as the Automotive SPICE and ISO26262 is a further constraint that must be managed to meet this objective. Nevertheless, ensuring sound management of safety and viewpoints is insufficient. It is also essential to ensure that we produce a system that is not only compliant and well-defined, but also that we produce the *"right*" system. Therefore, this leads to greater consideration of the requirements.

In this thesis, we address the challenge of development of automotive embedded systems following the model-driven engineering paradigm that meet the user needs and the regulatory constraints of the domain and that further mastered the quality of developed product. We resolve the problem in many steps which are subsequently used jointly. In the first phase, we define a merging approach which embodies a product quality and process quality approaches regarding the ISO26262 and SPICE standards following the model-driven engineering paradigm. Then, in a certification assessment purpose, we propose a generic methodology where an SPICE assessment and a functional safety audit is simultaneously performed without altering their original meanings. This commitment results into the definition of a tooled framework where we apply the SPICE assessment method to the common metamodel defined from the merging work. In a second phase, we define a metamodel for managing safety assets regarding these automotive standards at product level. This metamodel defines how the requirements and architecture of a system can be captured in such a way that they can be traceable from each other and from origin specifications documents. Finally, a model-based approach where the interaction of process and product models is managed to address requirements identified in the preceding phases is developed to support project management. The approach uses process modeling and measurement to improve the control and the monitoring of project and to reduce the cost and frequency of re-planning.

The benefits of the contribution are demonstrated on an ongoing automotive pilot application, thereby validating the research work against the weaknesses identified previously in the context.

Keywords: Automotive systems – Requirement engineering – Process engineering - Model driven engineering - System architecture - certification support – Traceability – Safety - Quality assessment - HIS Automotive SPICE - ISO26262

Résumé

Processus d'ingénierie des exigences dans un environnement à base de modèles selon les normes automobiles

Long Résumé

1 Contexte

L'industrie des systèmes embarqués est aujourd'hui consciente que la maîtrise de la sécurité innocuité est actuellement essentielle dans le développement de leurs produits. Afin de gérer ces aspects de sécurité innocuité, les normes propres à des domaines industriels spécifiques sont définies dans des optiques de certification. Dans le contexte automobile, ceci est réalisé par l'introduction de la nouvelle norme ISO26262, dont le but est de se conformer aux besoins spécifiques des systèmes (E/E) électriques ou électroniques dans les véhicules, pour toutes les activités liées à la sécurité innocuité durant le cycle de vie de ces systèmes. En fait, la norme met l'accent sur l'évaluation de la sécurité innocuité fonctionnelle, en proposant un système de classification basé sur l'ASIL (Automotive Safety Integrity Levels, en français niveaux d'intégrité de la sécurité innocuité automobile), des processus supplémentaires, des activités, des techniques et méthodes, les livrables attendus en sortie à travers un modèle d'application et un environnement pour illustrer la compétence dans la gestion des systèmes. Un point essentiel est que cette norme est utilisée par différents acteurs qui l'appréhendent différemment. Par exemple:

- Les architectes veulent s'assurer que le système qu'ils produisent est conforme à ce qui leur est demandé de concevoir;
- Les testeurs ont besoin de vérifier que ce qu'ils livrent est conforme et sécuritaire par rapport aux exigences et que l'intégration des sous système n'atténue pas l'intégrité de l'ensemble du système
- Les entités de certification doivent posséder tous les éléments pour procéder à la certification du système.

Les différents points de vue et cette approche de compréhension différente du même système ajoutent de la complexité aux processus pour la spécification, le développement et la mise en œuvre des systèmes. Afin de clarifier ces attentes, l'Ingénierie Dirigée par les Modèles (IDM) est largement adoptée. Ainsi, le modèle aide non seulement à gérer la complexité des systèmes, mais permet également aux développeurs de se concentrer sur uniquement certains aspects des systèmes grâce à des modèles abstraits, plutôt que sur des concepts liés aux algorithmes et à la programmation (par exemple, la programmation orientée objet).

Néanmoins, une gestion saine de la sécurité innocuité et des points de vue est insuffisante. Il est également essentiel de veiller à ce que nous produisons un système qui est non seulement sûr et bien défini, mais aussi que nous produisons le « bon » système. Par conséquent, cela conduit à une plus grande prise en compte des exigences du cahier des charges du produit à développer. En effet, négliger les exigences peut entraîner un produit inadéquat, quel que soit la justesse de sa conception et de son implémentation. D'ailleurs plusieurs études effectuées ces dernières années sur une grande variété de projets et d'entreprises ont confirmé ce problème de gestion des exigences. L'impact de l'ingénierie des exigences sur les succès des projets de développement de systèmes ne peut donc plus être ignoré.

Un autre point crucial est que les processus de développement sont définis sans tenir compte de leur qualité, comme ils sont pour la plupart construits empiriquement sur les réalités commerciales des équipementiers, les besoins des utilisateurs et les systèmes existants. Mais, comme il est communément admis que la qualité d'un produit dépend de la qualité d'un processus, beaucoup d'entreprises industrielles ont essayé par conséquent d'améliorer leurs processus logiciels. Dans la situation actuelle, les acteurs du domaine doivent prouver les capacités de leurs processus grâce à des modèles de maturité et des normes telles que CMMI (Capability Maturity Model Integration), ISO/IEC 15504 aussi connu comme SPICE (Software Process Improvement and Capability Determination), ou encore HIS Automotive SPICE dans le domaine automobile.... Ces processus définissent un ensemble de pratiques à respecter au cours du développement de logiciels. Ils fournissent les bonnes pratiques pour évaluer la capacité de développement de logiciels par l'entreprise et, selon l'évaluation qui en résulte, ils permettent d'identifier les points forts, ainsi que les faiblesses et les risques pour les empêcher. Malheureusement, parce que ces derniers n'incluent pas les aspects de sécurité innocuité, ils ne satisfont pas aux exigences d'une gestion cohérente de la sécurité innocuité.

En partant de ces états de fait, l'objectif de cette thèse est d'appréhender la gestion des exigences dans le contexte de l'IDM, en prenant en considération les besoins définis dans les standards de certification et, plus précisément la norme ISO26262 (une adaptation de l'IEC 61508 pour le domaine automobile) tout en garantissant le respect du référentiel HIS Automotive SPICE.

2 Préoccupations autour de la certification

Avant de discuter plus en détail des objectifs de cette thèse, les prérequis de notre contexte doivent être explicitement spécifiés, à savoir les différentes normes automobiles à respecter obligatoirement et leur impact sur la certification comme cela influence largement notre travail.

En effet, la certification a été l'objet de beaucoup d'attention ces dernières années. L'ISO définit la certification comme *« une procédure par laquelle une tierce partie donne une assurance écrite qu'un produit, un processus ou un service est conforme aux caractéristiques spécifiées».* Dans l'industrie, c'est un acte volontaire qui peut donner aux entreprises un avantage concurrentiel. La certification doit donc être comprise comme *« le processus vérifiant la valeur d'une propriété associée à quelque chose, et fournissant un certificat qui peut être utilisé comme une preuve de validité ».* Un organisme de certification indépendant peut fournir une garantie de qualité du produit par l'émission d'un tel certificat pour les entreprises et les pouvoirs publics. Pour les logiciels embarqués, deux orientations concernant la certification sont usuelles dans la pratique. La première est une orientation produit, qui met l'accent sur la spécification et l'évaluation des attributs de qualité du produit lui-même au moyen d'un ensemble de pratiques (outils, techniques et méthodes). La seconde est une orientation processus, qui met l'accent sur l'amélioration continue du processus de développement, sous l'hypothèse que les organisations avec des méthodes et processus avancés produisent des produits de qualité supérieure. L'ISO26262 suit la démarche de certification produit et l'Automotive SPICE suit l'approche de certification processus qualité. Notre conclusion est qu'il n'est clairement pas question de choisir entre l'une ou l'autre des orientations et que tous deux doivent être appliqués ensemble. Ainsi, une évaluation du produit logiciel doit être dépendante du processus de développement bien qu'il se soit révélé difficile de les gérer ensemble dans la pratique.

3 Objectifs scientifiques et critère de succès

Cette thèse est à la croisée de plusieurs préoccupations: nous devons faire face à la sécurité innocuité qui est un besoin commun de tous les acteurs du domaine de l'embarqué. Nous traitons de l'ingénierie des exigences nécessaire quand l'objectif est de concevoir un produit (physique, service...). Nous avons également constaté que l'IDM est utile pour gérer les différentes attentes des acteurs du domaine. L'objectif principal est **définir une synthèse entre les développements produit et processus selon un paradigme à base de modèle et de permettre la prise en compte des normes de référence du domaine formulées en termes d'exigences.**
Cet argument peut être décomposé en plusieurs questions de recherche:

- Quelles propriétés sont nécessaires à prendre en compte dans le cadre d'une modélisation de l'ingénierie des exigences?
- Quelles sont les exigences et les attributs nécessaires pour formaliser les processus de développement afin d'assurer leur efficacité, leur mesure et leur contrôle?
- Quelle méthodologie est suffisamment efficace pour fusionner les approches de certification produit et processus?

Ces questions cachent une autre question sous-jacente: comment interfacer l'IDM et les autres techniques actuellement utilisées au cours du cycle de développement ?

La thèse vise à contribuer à la fois à la recherche théorique et aux pratiques actuelles dans l'automobile. Le critère de réussite est donc que les résultats obtenus contribuent à d'autres projets de recherche universitaire et puissent être appliqués dans l'industrie.

4 Etat de l'art

Le but de ce chapitre est de positionner les objectifs de cette thèse dans l'état de l'art. Dans l'automobile, la production des systèmes embarqués critiques devient de plus en plus complexe à cause de nombreuses contraintes: délais serrés et réduction des coûts, gestion de la sécurité innocuité, amélioration et augmentation des performances en termes d'intelligence, de capacités, de caractéristiques, d'innovations, etc.... Une des questions principales dans ce contexte est la difficulté de gérer une multitude d'objectifs (de différents acteurs) au cours du développement de produits automobiles lorsque l'on doit essayer non seulement de construire un système sûr mais aussi le «*bon*» système. L'IDM semble être une bonne alternative pour une intégration transparente des différents points de vue. L'ingénierie des exigences permet de s'assurer que les besoins des utilisateurs sont satisfaits... Ainsi, nous avons examiné les approches de gestion des exigences dans le cadre de l'IDM pour l'industrie automobile. La nécessité de respecter les modèles de certification est un autre facteur. L'évaluation de la qualité efficace doit être possible pour démontrer la qualité de produits et ceux des processus dans un objectif de certification. Nous étudions également les travaux sur les processus et leur mesure pour le développement de systèmes embarqués toujours dans le cadre de l'IDM. En outre, toute cette étude de l'état de l'art est analysée avec la condition d'être conforme aux normes ISO26262 et SPICE.
En ce qui concerne la première question de recherche, nous pouvons citer les langages tels que SysML et diverses extensions comme DARWIN pour la spécification des exigences et EAST-ADL2 pour la description de l'architecture. Cependant, parfois ils échouent à satisfaire les recommandations

des standards. Par exemple, l'ISO26262 indique que: « *les exigences en matière de sécurité innocuité doivent avoir les attributs suivants: a) identifiant unique qui reste inchangé tout au long du cycle de vie de sécurité innocuité; b) statut; and c) ASIL* ». SysML n'est pas conforme à cette déclaration car il n'inclut pas de notion d'ASIL par exemple. Dans DARWIN, c'est la traçabilité avec les éléments architecturaux qui est manquante alors que les normes stipulent qu'une ingénierie des exigences satisfaisante et complète doit allouer également les exigences sur des éléments d'architecture. Bien qu'EAST-ADL2 réponde quant à lui à cette dernière exigence, beaucoup d'autres attributs et caractéristiques sur les exigences résumés dans le Chapitre 6 « Spécification et gestion des exigences de sécurité innocuité » de l'ISO26262-8 ne sont pas pris en compte, sont incomplets ou mal documentés. Une autre faiblesse de ces langages est leur classification des exigences qui n'est pas conforme avec la structure hiérarchique et organisationnelle imposée par l'ISO26262. Ils reposent généralement sur la répartition proposée par Glinz. Ce dernier classe les « exigences de sécurité innocuité » comme des exigences de qualité alors que dans l'ISO26262, toute exigence (même fonctionnelle) peut être considérée comme une « exigence de sécurité innocuité ». Le tableau suivant résume les attentes des normes qui ne sont pas respectées par ces langages.

Critères	SysML	DARWIN	EAST-ADL2
C1. Attributs et caractéristiques des exigences	✗	Basé sur l'ISO9126 ✓	Pas applicable
C2. Allocation des exigences sur les éléments architecturaux	Avec les tables d'allocation et le Block diagramme ✓	Pas applicable	Plusieurs vues d'architecture différentes ✓
C3. Structure et hiérarchie des exigences	✗	✗	✗

En réponse à la seconde question de recherche, SPEM semble être la meilleure solution pour la modélisation de processus comme il est expressément défini dans un objectif de développement de système. Il vise à améliorer non seulement le *produit*, mais aussi le *processus* qui conduit au produit, ce qui rend le développement de logiciels de haute qualité plus abordable. La qualité des produits peut s'évaluer par l'utilisation de normes portant sur la terminologie de la qualité des produits et en spécifiant cette qualité en termes mesurables. La qualité des processus peut être aussi évaluée à la modélisation et l'évaluation des processus de développement de logiciels établis. L'ISO26262, de la manière dont elle est définie peut servir comme un outil pour évaluer la qualité du produit et pareillement, SPICE peut servir à mesurer la qualité du processus. Néanmoins, on se doit également de prendre en compte les caractéristiques de qualité de l'ISO26262, comme par exemple la détermination de l'ASIL, ce qui n'est pas pris en compte dans SPEM. Les différentes étapes de SPICE à suivre pour les activités d'ingénierie système qui comprennent l'élicitation des exigences (ENG1), l'analyse des exigences système (ENG2) et la conception de l'architecture système (ENG3) doivent être modélisées, comme des activités spécifiques de sécurité innocuité. Mais une fois de plus, la classification ou hiérarchie des méthodes, outils et propriétés selon les niveaux d'ASIL au sens de l'ISO26262 ne peut pas être prise en compte comme il n'y a pas de concept équivalent dans SPEM.

Pour répondre à la troisième question, le processus de développement ainsi que le logiciel en lui-même peuvent être évalués ce qui permet un meilleur contrôle et des retours d'expérience. L'évaluation est donc une condition préalable et un excellent outil pour guider l'amélioration des processus car il fournit des informations sur les effets du processus sur la qualité des produits.

SPEM n'intègre pas la couche où cette mesure et le contrôle sont possibles lorsqu'un processus est en cours d'exécution. L'outil Permeter défini dans le cadre du projet européen CESAR tente de combler cette lacune grâce à son extension de l'outil EPF (Eclipse Process Framework) qui implémente le métamodèle SPEM. Un résumé des lacunes de SPEM est fourni dans le tableau ci-dessous.

Critères	SPEM
C1. Modélisation des processus	La classification ou hiérarchie des méthodes, outils et propriétés selon les niveaux d'ASIL ne peut être modélisée avec SPEM ⊗
C2. Mesure de processus	Règles de qualité de l'ISO26262, par exemple la détermination de la ASIL ne sont pas pris en compte dans SPEM ⊗
C3. Configuration de processus spécifiques	A partir d'un processus générique, il n'est pas possible de générer automatiquement un processus spécifique pour un projet ⊗
C4. Processus de surveillance	Le contrôle de processus n'est pas possible tant que SPEM ne fournit pas une couche pour l'exécution ⊗

La question sous-jacente traite de l'interopérabilité entre le processus de développement et le système en cours de développement, ainsi que l'interopérabilité entre des langages de modélisation et d'autres techniques actuellement utilisés au cours du cycle de développement comme les modèles Matlab Simulink, par exemple. Dans l'état des pratiques, cette question n'a pas de solution. L'outil Papyrus MDT est utilisable pour la modélisation du système; EPF avec les extensions de Permeter est en mesure de gérer les processus de modélisation et d'évaluation de façon plus ou moins limitée. Aucun des deux outils ne peut fournir les fonctionnalités proposées par l'autre. En outre, ils sont incapables de communiquer entre eux. Toutefois, étant donné que les avantages de l'IDM peuvent permettre l'interaction des modèles de processus et des modèles de systèmes et leur réalisation dans un outil, l'interrelation des propriétés des processus et des systèmes doit être gérée.

5 Méthodologie de recherche

Cette thèse a l'intention de fournir une solution cohérente qui permet la combinaison de la modélisation en matière de gestion des exigences et la prise en compte des normes automobiles dans une approche axée sur le modèle. Il est organisé autour des thèmes suivants:

- L'identification dans l'ISO26262 et SPICE des différentes exigences qui doivent être couvertes et leurs moyens de production d'une manière unique. Il en résulte une définition d'une approche fusionnée qui incarne les approches qualité produit et processus basées sur les standards ISO26262 et SPICE;
- La prise en compte par le biais et à travers les modèles des exigences identifiées ainsi que les aspects de sécurité innocuité pour soutenir tout le cycle de vie de ces exigences depuis leur définition jusqu'à leur allocation architecturale. Cette prise en compte tient compte des mécanismes de traçabilité comme la traçabilité entre les exigences, la traçabilité entre les exigences modélisées et une architecture système, la traçabilité depuis les documents de spécifications d'origine, etc…;

- La définition d'une méthodologie globale pour guider la conception des systèmes dans un tel environnement. L'approche utilise l'évaluation et la modélisation de processus afin d'améliorer le contrôle d'un projet et réduire les coûts et la fréquence de replanification.

L'intégration de ces mécanismes dans une plateforme outillée basée sur l'utilisation de plusieurs formalismes de modélisation est considérée dans la solution.
Le chapitre suivant détaille la contribution apportée pour chacun de ces points.

6 Contributions de la thèse

6.1 Fusionner (« Merge ») les normes ISO26262 et SPICE suivant une approche unique

Notre travail sur ce sujet est d'élaborer un instrument pour mesurer la qualité des produits d'une organisation qui développe des systèmes automobiles sûrs. Les deux normes ISO26262 et SPICE ont pour but de standardiser le développement des systèmes critiques automobiles pour gérer leurs complexités. Une étude a été effectuée pour extraire les concepts de sécurité innocuité et les règles liées aux processus applicables au domaine de l'automobile. Parmi les plus importants, le chapitre 6 « Spécification et gestion des exigences de sécurité innocuité » et les premiers processus de l'HIS Automotive SPICE (appelés ENG): élicitation des exigences (ENG1), analyse des exigences système (ENG2), conception de l'architecture système (ENG3) ont été analysés. Un environnement de modélisation semblait être le meilleur moyen de formaliser et d'exploiter ces éléments. J'ai donc proposé un métamodèle étendu pour décrire les deux normes dans un cadre commun sans modifier leurs contenus respectifs.
Compte tenu de l'objectif de certification, j'ai proposé une méthodologie générique où une évaluation de l'HIS et une vérification de la sécurité innocuité fonctionnelle est simultanément effectuée. Cette contribution se traduit par la définition d'un environnement où est appliquée une méthode d'évaluation au métamodèle commun défini auparavant. Cette méthode d'évaluation est basée de celle de SPICE. En effet, inspirée par le modèle d'évaluation des processus SPICE qui a déjà fait ses preuves dans l'industrie automobile, j'ai adopté sa notion de cotation (rating scale) afin de déterminer la maturité d'un système qui serait pleinement conforme aux deux normes automobiles. Ainsi, les ingénieurs système peuvent évaluer la pertinence de leurs processus de développement et également la qualité des systèmes qu'ils développent vis-à-vis des normes et se faire une idée du niveau de maturité qu'ils ont atteint.
L'approche a été testée sur un sous-ensemble des deux normes. Le principal avantage de notre proposition sous forme d'un tel processus d'évaluation intégré est qu'il réutilise les pratiques déjà présentes dans l'industrie, réduisant ainsi les efforts d'introduction de la nouvelle norme.
La technologie choisie pour l'implémentation de l'environnement est Excel™. Néanmoins, compte tenu de la quantité de données et la mise en œuvre de nombreux algorithmes, des difficultés à maintenir ou ajouter d'autres fonctionnalités sont rencontrées. Une solution est de trouver un format plus approprié afin d'assurer une évaluation efficiente et efficace. Une comparaison de notre métamodèle commun avec le métamodèle SPEM suggère qu'il serait possible de le traduire dans ce langage de processus avec l'ajout de certaines extensions afin de développer et de couvrir toutes nos notions. Cette solution est présentée dans la dernière partie de ce chapitre.

6.2 Spécification des activités d'ingénierie des exigences avec prise en compte des aspects de sécurité innocuité dans un environnement d'Ingénierie Dirigée par les Modèles

Parce que la qualité du produit est également un facteur de la certification, un objectif de la thèse visait à identifier dans les différentes normes les besoins à couvrir dans le cadre de la gestion des exigences et les mettre dans un cadre de modélisation. Cet objectif est réalisé par le profil ReMIAS. A travers celui-ci, les différents moyens et méthodes de production des exigences qui permettent d'être conforme aux normes automobiles et aux techniques d'ingénierie des exigences système sont exploités. Une combinaison des langages dédiés pour la spécification des exigences SysML et DARWIN a inspiré la spécification des exigences. Le but du processus d'ingénierie des exigences est de lier les exigences analysées aux éléments d'architecture pour leur validation. Des sous-ensembles d'EAST-ADL2 sont étendus pour gérer la partie de conception d'architecture et celle de vérification et validation. La traçabilité est assurée à partir des liens de traçabilité hérités de DARWIN. Cette traçabilité est également assurée par la réalisation d'un plugin Eclipse *Office2Papyrus* qui permet un import automatique des exigences à partir des documents de spécification (Microsoft Word™ et Excel™) dans l'environnement de modélisation. Le plugin permet de renseigner les exigences directement sous forme d'éléments de modèles (diagramme d'exigences SysML) évitant une redondance de travail aux ingénieurs, sans changer leurs pratiques et sans perdre les informations préalablement existantes lorsqu'ils valident les exigences à travers ces modèles. Le résultat en est que les approches modèle et texte sont intégrées de manière normalisée. Autre avantage du plugin *Office2Papyrus* est qu'il peut être utilisé dans n'importe quel outil. L'utilisation de la norme *Requirement Interchange Format* (ReqIF) permet de confirmer cette déclaration. En anticipant un alignement progressif des concepteurs d'outil de gestion des exigences avec la norme ReqIF, l'interopérabilité avec les outils commerciaux serait possible.
Grâce à cette contribution, on est en mesure de définir le produit à travers un modèle détaillé, intégrant les différents points de vue de même que les exigences et les phases de conception peuvent être retracées tout au long du processus de développement.

6.3 Modélisation et mesure des produits et processus selon les spécifications de qualité

La dernière expansion nécessaire à l'objectif de la thèse est la modélisation et l'évaluation du système et du processus qui a mené à sa conception selon des contraintes de qualité. En prenant en compte les extensions apportées à SPEM, la méthode de configuration d'un processus générique selon les contraintes pour obtenir un processus spécifique à un projet et les extensions apportées par l'outil Permeter, l'objectif est atteint. La modélisation du processus est assurée par le métamodèle commun défini et présenté au début du chapitre. Le métamodèle est le métamodèle commun défini qui reprend tous les concepts des deux normes. L'interrelation des artefacts processus et de ceux du système est gérée grâce aux éléments d'extensions apportés à SPEM au niveau des *Workproducts*. L'instanciation du métamodèle permet d'obtenir un processus organisationnel générique conforme aux normes automobiles au centre de nos préoccupations. A partir de ce modèle de processus générique, une démarche pour l'adapter en processus spécifiques en fonction du contexte et de la caractérisation des projets est définie, afin de créer des produits plus efficaces et plus efficients. Étant donné que ce processus d'adaptation est pensé pour être automatique, il est prévu de parvenir à une réduction du temps et des coûts, et aussi d'avoir moins d'erreurs liées à l'adaptation car seuls les éléments de processus qui sont requis pour le contexte du projet particulier sont considérés. En outre, on peut s'attendre à ce que la qualité soit améliorée,

parce que le processus est ajusté à l'objectif du contexte particulier de chaque projet. L'objectif d'évaluation est abordé avec l'utilisation des outils EPF et Permeter. Un inconvénient est que l'écriture des requêtes OCL pour la définition des métriques qui servent à cette évaluation est trop complexe pour l'utilisateur standard. Pour être en mesure de définir les métriques, il est nécessaire d'avoir des connaissances approfondies sur le langage OCL. Une interface graphique pour faciliter la tâche à l'utilisateur a été développée afin de rendre l'outil convivial et accessible à toute personne.
La pertinence de ces résultats a été éprouvée sur une application automobile, un système automobile actuellement en développement nommé BSG_E (Boîtier de Servitude Générique – Electronique).

7 Evaluation des contributions

La thèse poursuit l'objectif d'utiliser les exigences dans un environnement de modélisation pour définir des systèmes sécuritaires et conformes aux normes automobiles, suivant un processus de modélisation hiérarchique qui vise les activités d'ingénierie système. Dans l'état de l'art et des pratiques, nous avons trouvé trois grands domaines qui avaient besoin d'expansion. J'ai présenté une contribution pour chacune d'elles, à savoir:

- la fusion des normes ISO26262 et SPICE suivant une approche unique
- la spécification des principales activités d'ingénierie des exigences dans un environnement basé sur le modèle en tenant compte des aspects de sécurité innocuité et des activités d'ingénierie système
- la modélisation et l'évaluation du système et du processus de développement du point de vue de la qualité

Pour les évaluer, les résultats obtenus au travers des développements sur notre application pilote ont été évalués par rapport aux tableaux de critères définis suite à l'étude de la littérature.
Le premier tableau présente les critères induits par les normes pour assurer une gestion appropriée des exigences. L'analyse a montré que ces critères étaient couverts par différents langages de modélisation, mais à des degrés différents et aucun ne les satisfaisait tous en même temps.
Le premier critère concerne les caractéristiques de qualité des exigences et leurs attributs. Le profil DARWIN remplissait déjà partiellement cette exigence et il a été réutilisé de fait dans le profil ReMIAS avec les apports ciblés.
L'allocation des exigences sur les éléments architecturaux nécessite la possibilité de définir et d'utiliser des éléments de conception. La partie d'architecture d'EAST-ADL2 incarne cette fonctionnalité dans le profil ReMIAS. Le troisième critère n'était pas rempli par l'état de la pratique parce qu'il s'agit surtout d'une spécificité propre à la nouvelle norme ISO26262. À travers les règles mises en œuvre pour le définir comme profil statique, ce challenge est aussi résolu. Avec le plugin Office2Papyrus et les artefacts de traçabilité du profil, l'automatisation des liens entre les différentes phases du processus au niveau de développement est assurée. Parce qu'il réutilise des parties de profil qui satisfont déjà certains des critères et qu'il implémente en plus d'autres exigences recommandées par les normes sous forme de règles dans un profil statique, le profil ReMIAS atteint tous les critères concernant l'objectif de départ.
Le deuxième tableau des critères était sur l'appropriation des normes et leur manipulation d'un point de vue processus qui inclut leur modélisation et leur contrôle, principalement avec l'outil Permeter. Ces critères sont principalement couverts par la fusion des deux normes ainsi que par l'environnement d'évaluation proposé. Ils sont aussi couverts par la troisième contribution à travers

les extensions apportées au langage de modélisation ainsi que la démarche de configuration de processus spécifique. Bien qu'ils ne soient pas développés sur le plan technique, leur évaluation théorique suggère que ce sont des pistes intéressantes.
L'autre avantage est que grâce à cette dernière contribution, la partie processus de développement est liée avec la partie développement du système car l'extension du métamodèle SPEM proposée par le projet CESAR permet d'examiner les modèles du système comme artefacts dans le modèle de processus.
La solution proposée est un atout prometteur pour le développement de systèmes embarqués dans le domaine des systèmes de transport intelligents, où il y a une marge énorme pour l'amélioration des processus de développement. Elle vise à fournir un meilleur environnement pour le développement des logiciels et des systèmes, la gestion des exigences et des architectures incluant des métamodèles, des méthodes et des outils pour le développement des systèmes temps réel critiques tout en les rendant interopérables. A la fin de l'évaluation, il est possible d'affirmer que la contribution couvre les besoins majeurs de notre application pilote.

8 Validation des objectifs scientifiques

Dans ce chapitre, nous passons en revue les contributions par rapport aux questions de recherche. Rappelons que cette thèse soutient **la définition d'une synthèse entre les développements produit et processus selon un paradigme à base de modèles et de permettre la prise en compte des normes de référence du domaine formulées en termes d'exigences.**

Cet objectif avait été décomposé en plusieurs questions de recherche. Tout d'abord: Quelles propriétés sont nécessaires à prendre en compte dans le cadre d'une modélisation de l'ingénierie des exigences?
Cette question a été abordée en développant le profil ReMIAS basée sur une lecture de l'ISO26262 pour gérer les attributs des exigences et SPICE pour s'inspirer du processus à suivre pour ces activités. En résumé, les principales caractéristiques de la contribution sont:
Spécification des exigences. Cela a été identifié comme une méthode efficace pour apporter une qualité dans la gestion des exigences.
Gestion des aspects de sécurité innocuité dans l'ingénierie des exigences. Cet aspect considère la détermination et la classification de l'ASIL pour les exigences, première étape pour la gestion des exigences de sécurité innocuité et pour gérer les recommandations ISO26262.
Structure hiérarchique et classification. La structure des exigences et leur hiérarchie dans l'ISO26262 est différente de celles existantes car la « sécurité innocuité » n'est plus considérée comme une exigence de qualité particulière mais comme une nouvelle classe d'exigences qui suit des règles spécifiques afin d'assurer la sécurité innocuité des systèmes automobiles.
Définition d'une architecture système et allocation des exigences sur ces éléments architecturaux. ReMIAS offre la traçabilité des exigences vers des éléments architecturaux comme recommandé par les normes.

La seconde question de recherche était: Quelles sont les exigences et les attributs nécessaires pour formaliser les processus de développement afin d'assurer leur efficacité, leur mesure et leur contrôle?
Cette question a été traitée par l'élaboration d'un métamodèle de processus avec un métamodèle d'évaluation. En résumé, les principales innovations sont:

Métamodèle de processus intégrant des artefacts définis dans l'ISO26262. Cela était nécessaire pour gérer des éléments de processus à suivre et pour évaluer la conformité aux exigences de l'ISO26262 comme l'ASIL, la classification des outils et des méthodes selon les niveaux de sévérité recommandés.
Définition d'un processus spécifique selon les objectifs spécifiques d'un projet. Obtenir un processus qui respecte la norme automobile et qui est dans le même temps spécifique à un projet selon son contexte et ses contraintes propres permet d'améliorer la gestion de projet.
Définition des métriques et des objectifs au niveau processus. La définition d'objectifs et de contraintes est la manière de mesurer l'efficacité et l'évolution des activités et des tâches des processus. La mesure est utilisée à deux fins: pour l'évaluation de la qualité des produits et pour l'évaluation des effets des processus. Grâce à une exécution, ces métriques aident à suivre l'intégralité des activités du processus et à améliorer le suivi de projet.

La dernière question de recherche était: Quelle méthodologie est suffisamment efficace pour fusionner les approches de certification produit et processus?
L'élaboration d'un environnement où les exigences de l'ISO26262 et de SPICE sont fusionnées permet de répondre à cette question. Les innovations clés dans cette contribution sont la mise en relation des normes et l'élaboration d'un outil d'audit unifié pour la certification. Cette contribution permettra aux ingénieurs système d'évaluer la pertinence ainsi que la qualité de leurs systèmes et des processus de développement qui ont conduit à les définir à des fins de certification.

Les contributions additionnelles suivantes ont été réalisées dans la thèse:
Définir d'un langage commun pour soutenir la communication entre les différents acteurs. Un défi clé dans les projets industriels est la gestion d'une perspective commune. Ceci est géré par une représentation riche et intuitive, basée sur la notation graphique. Cette représentation libère également de l'utilisation de plusieurs formalismes différents.
Des outils basés sur le paradigme des modèles. Pour soutenir la traçabilité depuis les documents de spécification à la définition de l'architecture, l'approche définie propose certains outils, tous basés dans un environnement de modélisation et interopérables entre eux pour la limitation des activités manuelles communément source d'erreurs.
Ainsi, les contributions de cette thèse répondent à tous les objectifs définis au préalable.

Le critère de succès retenu stipulait que cette recherche pourrait être bénéfique tant pour la recherche académique que pour l'industrie. Cela a été entièrement atteint.
Application dans l'industrie. L'environnement de certification a été utilisé par DELPHI pour élaborer une méthodologie de vérification qui impliquait des recommandations de SPICE et de l'ISO26262 ensemble. Le profil ReMIAS avec sa gestion des exigences et de l'architecture a servi aux ingénieurs de système pour améliorer leur conformité au processus de spécification de l'architecture système du standard SPICE (ENG 3), qui n'était pas du tout couvert. La théorie pour obtenir un processus spécifique à partir d'un processus générique selon le contexte, a suscité beaucoup d'intérêts par les ingénieurs qualité dans le sens où ils peuvent se concentrer directement sur les recommandations des normes qui ont des conséquences dans le projet actuel et qu'ainsi ils se libèrent de ceux qui ne sont pas applicables dans ce contexte. Ces utilisateurs ont investi beaucoup de temps dans l'application des résultats de recherche, indiquant ainsi leur pertinence.
Contribution à la recherche. L'étude de l'ISO26262 constitue un élément important du projet de recherche collaboratif SASHA. La gestion de la traçabilité depuis les documents de spécifications

vers un environnement de modélisation a été perçue comme essentielle, tout comme dans le cadre du projet collaboratif européen CESAR. Ces faits valident la contribution à la recherche.

De plus, ces résultats ont fait l'objet de plusieurs publications et ont été sources de contacts de collaboration avec plusieurs industriels. Cela appuie encore plus la pertinence des contributions pour la recherche et l'industrie.

Declaration

Some of the material presented in this thesis has been published in the following papers and reports:

[1] Adedjouma, M.: Prise en compte de la Sûreté de Fonctionnement dans une Approche d'IDM (French title). Master's Thesis, Compiègne (2008)

[2] Cancila, D., Dubois, H., Adedjouma, M.: Etude de la sûreté de fonctionnement dans un processus basé sur l'ingénierie dirigé par les modèles (French title). In: Proceedings of 5emes journées de l'ingénierie Dirigé par les modèles (IDM), pp. 73-78, Nancy (2009)

[3] Adedjouma, M., Dubois, H., Maaziz, K., Terrier, F.: A Model-Driven Requirement Engineering Process Compliant with Automotive Domain Standards. In: Hein, C., Wagner, M., Mader, R., Keis, A., Armengaud, E. (eds.) Model-Driven Tool & Process Integration (MDTPI). Fraunhofer Institute for Open Communication Systems, pp. 85-96, Paris (2010)

[4] Adedjouma, M.: Model-Based Requirements Process for Safety Automotive Applications. In: the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ Poster), Essen (2011)

[5] Adedjouma, M.: A Model-Based Requirements Engineering Framework in an Automotive Certification purpose. In: Doctoral Symposium of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Essen (2011)

[6] Adedjouma, M., Dubois, H., Terrier, F.: Requirements Exchange: From Specification Documents to Model. In: the 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 350-354, IEEE Press, Las Vegas (2011)

[7] Adedjouma, M., Dubois, H., Terrier, F.: Merging the Quality Assessment of Processes and Products in Automotive Domain. In: the 13th International conference on Product-Focused Software Development and Process Improvement (PROFES), Madrid (2012)

[8] Adedjouma, M., Dubois, H., Terrier, F., Kitouni, T.: An Experiment on a Merging Quality Assessment in Automotive Domain. In: International conference on Software Process Improvement and Capability DEtermination (SPICE), Palma de Mallorca (2012)

[9] Aboutaleb, H., Bouali, M., Adedjouma, M., Suomalainen, E.: An Integrated Approach to Implement System Engineering and Safety Engineering Processes: SASHA project. In: the 6th European Congress on Embedded Real-Time Software and Systems (ERTS²), Toulouse (2012)

[10] Adedjouma, M., Wojciech, M., Dubois, H., Terrier, F.: Efficient Methodology from Requirements to Design Models for an Automotive Application. In: the 6th European Congress on Embedded Real-Time Software and Systems (ERTS²), Toulouse (2012)

[11] Peraldi-Frati, M-A, Goknil, A., Adedjouma, M., Gueguen, P.-Y: Modeling a BSG-E automotive system with the Timing augmented description language. In: the 5th International Symposium On Leveraging Applications of Formal Methods (ISoLa), Heraclion (2012)

Some other parts of my research were published as outcome parts of R&D collaborative works, mainly the CESAR and SASHA projects.

[12] EU Project CESAR: Detailed Analysis Report on Requirements Engineering. In: D_SP2_R1.1_M1 deliverable, Ed. Ganesh Pai et al. (2010)

[13] EU Project CESAR: RE Language Definitions to formalize multi-criteria requirements. In: D_SP2_R2.2_M2 deliverable, Ed. Andreas Mitschke et al. (2010)

[14] EU Project CESAR: RE Process Framework Definition. In: D_SP2_R3.2_M2 Vol 1 deliverable, Ed. Francesco Lanteri et al. (2011)

[15] EU Project CESAR: Requirement Capturing Specification of Improved Methods. In: D_SP2_R3.3_M3_Vol2 deliverable, Ed. Herbert Zojer et al. (2011)

[16] EU Project CESAR: Requirement/Traceability Management. In: D_SP2_R3.3_M3_Vol3 deliverable, Ed. Emmanuel Leroy et al. (2011)

[17] EU Project CESAR: Completeness/Consistency/Correctness. In: D_SP2_R3.3_M3_Vol4 deliverable, Ed. Guillaume Allain, Marc Malot et al. (2011)

[18] EU Project CESAR: Processing of Requirements. In: D_SP2_R3.3_M3_Vol5 deliverable, Ed. Ralf Bogusch et al. (2011)

[19] EU Project CESAR: First RE Toolset. In: D_SP2_R4.1_M2 deliverable, Ed. Bernhard Josko et al. (2010)

[20] EU Project CESAR: Assessment of Requirements Engineering Methods. In: D_SP2_R6.3_M3 deliverable, Ed. Silke Köhler, Jan Gacnik, Álvaro Catalá-Prat, Uday Biswas, Ralf Bogusch, Erwin Reyzl et al. (2012)

[21] EU Project CESAR: Public evaluation report for automotive RTP v3. In: D_SP5_R4.4_M4 deliverable, Ed. Alois Rainer et al. (2012)

[22] French Project SASHA: ”Rapport de déroulement du projet à T0+12". In: Livrable 0.1, Ed. Jouvene-Faure Franck et al (2011)

Contents

Figures

Tables

1

General introduction

1 General Introduction

1.1 Context

The embedded safety-critical systems industry is facing an exponential increase in the complexity and variety of systems and devices while costs, performance in terms of intelligence, features, capacities and time to market are constantly challenged.

In the automotive domain in particular, maintaining global markets and increasing market shares is achieved through implementing differentiating innovations. It is considered that most innovations in the automobile field rely on the implementation of electrical and electronic technologies. This has led to a profusion of enablers (new processes, methods and tools), which are neither integrated nor interoperable because they have been developed more or less independently. This lack of interoperability among these innovations developed is a limit in terms of industrial performance when companies have to make a selection from these enablers. In this environment, a major factor in the success of a project is: correctly taking into account the functional needs, i.e. requirements, reducing time and costs through reusing design and links to the code and, finally, the flexibility to adapt the applications to various existing or future technologies. The Model Driven Engineering [FRA 07b] (MDE) paradigm forms a unified methodological framework to handle this matter by addressing the whole issue of software engineering. It conceives the entire software life cycle as a production process of iterative refinement and continuous integration by introducing an additional level of abstraction, the model [BEZ 02], from which the program traditionally written by hand is automatically generated.

On the other hand, the complexity of the developed features, such as electronic and computer-based architectures which have been implemented for instance, risk, if poorly controlled, either preventing the maturity of innovations involved, thus resulting in a lack of offers on the market (no increased income) or generating non-perceptible quality and thus creating a poor public image (decline in sales) or product. For example, introducing the driver assistance systems or the electrification of critical systems such as braking to the market must be mastered. The transfer of more and more responsibility for conducting from the driver to mechatronic systems in the car requires a harmonized strategy of product introduction between the industrialist and the market. Manufacturers can be held responsible and a harmonization of approaches to systems development related to dependability is necessary, whereas safety is no longer a desirable quality for critical systems. It is however required, when it is considered that any software or hardware failure, can lead to serious loss of life, security and / or material.

The main objective for automotive manufacturers and suppliers is now becoming the control of quality and the dependability of embedded and mechatronic systems. The existence of internationally recognized standards such as the Automotive SPICE [AUT 08] and ISO26262 [ISO 11] is a further constraint that must be managed to meet this objective.

This thesis focuses on the means which ensure the development of automotive embedded systems following the model-driven engineering paradigm that meets the user needs and the regulatory constraints of the domain and that further master the quality of developed product.

1.2 Context issues

The embedded systems industry is aware that mastering safety is currently crucial in their product development. In order to manage safety aspects, standards are defined in different areas of certification

expectations, in a particular area of special hardware and software production. In the automotive context, this is achieved by introducing the ISO26262 standard, with the goal of complying with needs specific to the application sector of electrical and/or electronic (E/E) systems within road vehicles for all activities during the safety lifecycle of safety-related systems [ISO 11]. In fact, the standard focuses on the assessment of functional safety, proposing a classification system with ASIL (Automotive Safety Integrity Levels) [ISO 11] levels, additional processes, activities, techniques and methods, expected output data through an application model and framework to illustrate the competence for managing systems. A critical point is that the challenge is faced by different actors who generally have different goals often related to specific business sectors. For instance:

- Designers who want to ensure that the system they produce is compliant with what they are asked to design;
- Integrators who need to check that what they are delivering is compliant and safe compared to requirements and that the integration does not mitigate the integrity of the whole;
- Certification entities which must possess all the elements to allow the certification system to proceed.

The varying points of view and different understanding of the same system increase the complexity of processes for their specification, development and implementation. In order to clarify these different expectations, the model driven engineering is widely adopted. According to [BEZ 02],"*MDE seems to be able to answer some of those challenges for which the Object Oriented technology has failed to find answers internally. Among these we could mention: (1) Separate management aspects; (2) Homogeneous consideration of functional and nonfunctional elements; (3) Seamless integration of different viewpoints (rules, services, processes, architecture, etc…), etc…The MDE seems to subsume the Object Oriented technology and provide a smooth evolution path to current solutions based on objects and components*". So, the model not only helps to manage the complexity of systems, but also allows developers to focus on how to treat aspects of systems through abstract models, rather than on concepts related to algorithms and programming.

Nevertheless, ensuring sound management of safety and viewpoints is insufficient. It is also essential to ensure that we produce a system that is not only compliant and well-defined, but also that we produce the *"right*" system. Therefore, this leads to greater consideration of the requirements. Indeed, neglecting requirements may result in an inadequate product, no matter how elegant its design and implementation [STE 94]. Some 40 years later, different surveys covering a wide variety of organizations and projects [STA 95] have confirmed this problem with requirements on a much larger scale. The impact of requirements engineering on successful and customer oriented systems development can thus no longer be ignored [POH 11].

Another critical point is that the development processes are defined without any regard to their quality, as they are in most cases empirically constructed over business realities from OEM (Original Equipment Manufacturers), user needs and existing systems. But, it is commonly accepted that the quality of a product depends on the quality of a process, and consequently many industrial companies have tried to improve their software processes. In the current situation, suppliers have to actually prove process capabilities to OEM through maturity models, and standards such CMMI (Capability Maturity Model Integration) [CMM 02], ISO/IEC 15504 also known as SPICE (Software Process Improvement and Capability Determination) [AUT 10a, AUT 10b], HIS Automotive SPICE [HIS 12] in the automotive domain, etc... These processes are described as *"process-based*", in that they define a set of practices to be adhered to during the software development. They provide good strategies to assess the organization's software development capability and, based on the resulting assessment, they allow identification of the process

strengths, weaknesses and risks for preventing them. Unfortunately, because the latter do not include safety aspects, they do not satisfy requirements for a consistent safety management.

1.3 Research motivation

Building on the previous statements, the relevant goal in this thesis is to take the requirements management into account in the context of model driven engineering, with the objective of considering the needs identified in the certification standards and, more specifically, in the automotive domain with the ISO26262 standard and the Automotive SPICE referential.

For this purpose, consideration of certification needs calls for a real justification of the various phases of the modeling process to meet expectations. Better consideration of requirements is also necessary to ensure that we produce a “right” while staying focused on the assets of the business [AWA 07]. In fact, in a hierarchical modeling process to define system specifications, from most upstream stages to the lowest stages of target code generation, the requirements use must allow expectations of each modeling level and of each actor to be clearly specified. As it is inconsistent to specify requirements without considering the means to validate them, special attention must be paid to define modeling elements of system architecture to ensure the traceability of our requirements throughout the V-cycle.

At the end, this will result in an instrument for systems development to measure process capability of a specific engineering organization that develops safety systems, and that focuses on software certification through both the end-product quality approach and the development process approach.

In general, no process framework tool and no modeling process for guidance exists in practice for the existing model-driven engineering methodologies. It is also overlooked that people must improve their skills through special training and specific assignments, because this can significantly affect working efficiency. A project monitoring process using modeling process languages and standards to guide the various actors involved in the design of these systems will be then be integrated.

1.4 Research questions and success criterion

This thesis is at the crossroads of several concerns: we are dealing with safety which is a common subject to all societal actors working in the embedded domain. We are focusing on requirements engineering that affects a wider audience, as long as that we want to sell a product (physic, service...). We have also seen that model-based engineering is useful to manage different expectations of actors in the domain and respond to tight schedules. The main objective is the following:

> **Define a synthesis between product and process development following a model-based paradigm and to provide integration of standards reference, formulated in terms of requirements.**

This argument may be broken down into the following research questions:

- What properties are necessary in a requirements engineering modeling framework?
- What are the requirements and attributes needed to formalize the development processes with respect to their performance, measurement and monitoring?
- Which methodology is sufficiently effective to merge the certification approaches for both the process development and end-product?

The threefold scientific issue hides an underlying question:

- How can we manage the interface between the model driven engineering paradigm and other techniques used during the development cycle?

The research project is intended to contribute to both academic knowledge and current practice in the automotive domain. The success criterion is therefore that results should contribute to other academic research projects and should be applied in industry.

1.5 Research methodology

Our research goal is to provide a consistent solution that allows the combination of modeling, requirements management at models level and consideration of the automotive standards. We follow a model-driven approach, because it is assumed that modeling is a semi-formal representation that provides the ability to understand, communicate and keep the engineering information in a well-defined structure. More particularly, we use UML (Unified Modeling Language) [OMG 11a, OMG 11b], which is a modeling language widely used in the industry. Furthermore its use has been increased in recent years by standards such as ISO26262.

The proposal aims to provide advanced techniques and notations supporting the full requirements life cycle, from their definition up to their architectural allocation. It integrates the recommendations of ISO26262 and those of the Automotive SPICE standard, for example the requirements properties specification, the requirement allocation to an item or architectural element. It is organized around the followings topics:

- The identification in the ISO26262 and automotive SPICE standards of different needs to be covered and their means of production;
- The consideration through and into models of identified requirements;
- The definition of mechanisms to trace requirements throughout the process development: traceability between requirements; traceability to model elements and to a system architecture which will be defined, traceability to process aspects;
- The consideration of multi-formalisms modeling and interoperability tools;
- The integration of these mechanisms in a tooled platform based on UML modeling;
- The definition of a comprehensive methodology to guide the design of systems in such an environment.

The validation of the theoretical soundness and usefulness of these ideas will be tested on an ongoing automotive application. The methodology will improve understanding of the whole process and cooperation between engineers' teams.

1.6 Thesis outline

This dissertation is divided into six sections:

1. **Introduction.** The thesis is introduced, and different issues in the current context are identified. The research motivation is exposed; the research questions and success criteria are defined and the research methodology is discussed.

2. **Different certification approaches in the automotive domain.** A presentation of the safety standard in the automotive domain is discussed. It describes existing practices relating to hazard identification, risk assessment, and safety requirements management. The existing quality assurance referential is also

described with the definition of maturity models, capability levels, improvements and assessment methods in the domain. Strengths and weaknesses in the application of these standards are highlighted in the conclusion.

3. **Product and process in model driven engineering framework**. A related literature review of product and process development is surveyed. Firstly, it discusses existing practices related to requirements engineering, the commonly used methods for their modeling, the system architecture definition and the traceability in the context of automotive applications as the supported tools. Process development is also introduced, exploring the core assets of the modeling languages and the tooled platforms. The section is concluded with a review of capability and measurement models in product development and quality assessment. It is argued that no single modeling approach provides an adequate description of practice. The need for additional research to identify the most appropriated basis for addressing the research questions is highlighted.

4. **Contribution.** The conceptual basis for this research is detailed in this section by defining a merging approach which embodies a product quality and process quality approach with respect to the ISO26262 and SPICE standards. A metamodel for defining safety assets at product level regarding these automotive standards is then presented. This metamodel defines how the requirements and architecture can be captured in such a way that they can be traceable from each other and from origin specifications documents. A model-based approach to support project management is developed to address requirements identified in the first point, and thereby to provide a context for addressing the research questions. The approach uses process modeling and measurement to improve the control and monitoring of a project and to reduce the cost and frequency of re-planning. The benefits of the contribution are demonstrated through CASE tools where the interaction of process and product models is managed.

5. **Industrial application of the approach.** An evaluation of the main contributions of the research of the previous chapters is presented. This evaluation is carried out by means of tool support and a pilot industrial application, thereby validating the research contributions with respect to the success criterion discussed above.

6. **Conclusions.** The research questions are discussed in light of the dissertation. Research contributions are reviewed, opportunities for further work are highlighted and conclusions are drawn.

2

Certification issues in automotive domain

2 Certification issues in automotive domain

2.1 Automotive standards

Before discussing the contributions of this thesis, the prerequisites of the context need to be explicitly specified, namely the different automotive standards to be met mandatorily as they greatly influence our work.

2.1.1 ISO26262 standard

The ISO26262 [ISO 11] standard is a new standard defined for handling safety purposes. It is the adaptation of IEC 61508 [IEC 10] for road vehicles. The standard, based upon a V-model [INC 07], focuses on the assessment of functional safety proposing a system classification with ASIL (Automotive Safety Integrity Levels) levels, together with appropriate requirements, additional processes, activities, techniques and methods, expected output data etc... The final goal is to provide evidence that all reasonable system safety objectives are satisfied, to justify the acceptability of safety based upon product-specific and finally to highlight evidence to illustrate the competence for managing systems. Hazard analysis, risk assessment and ASIL determination are preliminary tasks used to help determine this evidence. An overview of the latter is then provided before exploring how they contribute to the definition of safe systems.

2.1.1.1 Hazard analysis, risk assessment and functional safety concept

Based on the item definition, i.e. the description of the system, the safety lifecycle is initiated by hazard analysis and risk assessment activity. Firstly, the hazards are identified systematically using adequate techniques: brainstorming, checklists, quality history, FMEA (Failure Modes and Effects Analysis) [LAP 96]; hazardous events which will cause an item's malfunctioning behavior and the consequences are described for relevant combinations of operational situations and operating modes before they are classified [LAD 03, BLA 03, ISO 11, IEC 10]. The classification is done by determining the ASIL for each of them [ISO 11]: ASIL A, ASIL B, ASIL C and ASIL D, where ASIL A is the lowest safety integrity level and ASIL D the highest. In addition to these four ASILs, the class QM (Quality Management) denotes no requirement to comply with ISO26262, i.e. it does not integrate any risk. A top-level safety requirement [MOD 07, CEN 11, IEC 11] also called safety goal [ISO 11] with its ASIL evaluated for each hazardous event is the end result of the hazard analysis and risk assessment. Following this, the functional safety requirements with their associated information are derived from the safety goals and their allocation to architectural elements or external measures [ISO 11, IEC 10] is performed. Lastly, an acceptance criterion for safety validation of the item is specified.

The next tasks focus on system safety activities that typically occur prior to allocating safety requirements to software and hardware items. These are specifications of technical safety requirements and system design.

2.1.1.2 Safety development at system level

Based on the functional safety concept specification, development at system level is started with the specification of technical safety requirements with their ASIL, derived from the functional safety concepts previously identified, considering both the functional requirements and the preliminary architectural assumptions [ISO 11]. The technical safety requirements are the technical requirements necessary to implement the functional safety concept, with the intention being to detail the item-level functional safety requirements into system-level technical safety requirements. This includes the safety mechanism i.e. the response of the system or elements to stimuli that affect the achievement of safety goals. The Figure 2-1 below illustrates the hierarchical approach by which the safety goals and the functional safety requirements are determined as a result of the hazard analysis and risk assessment.

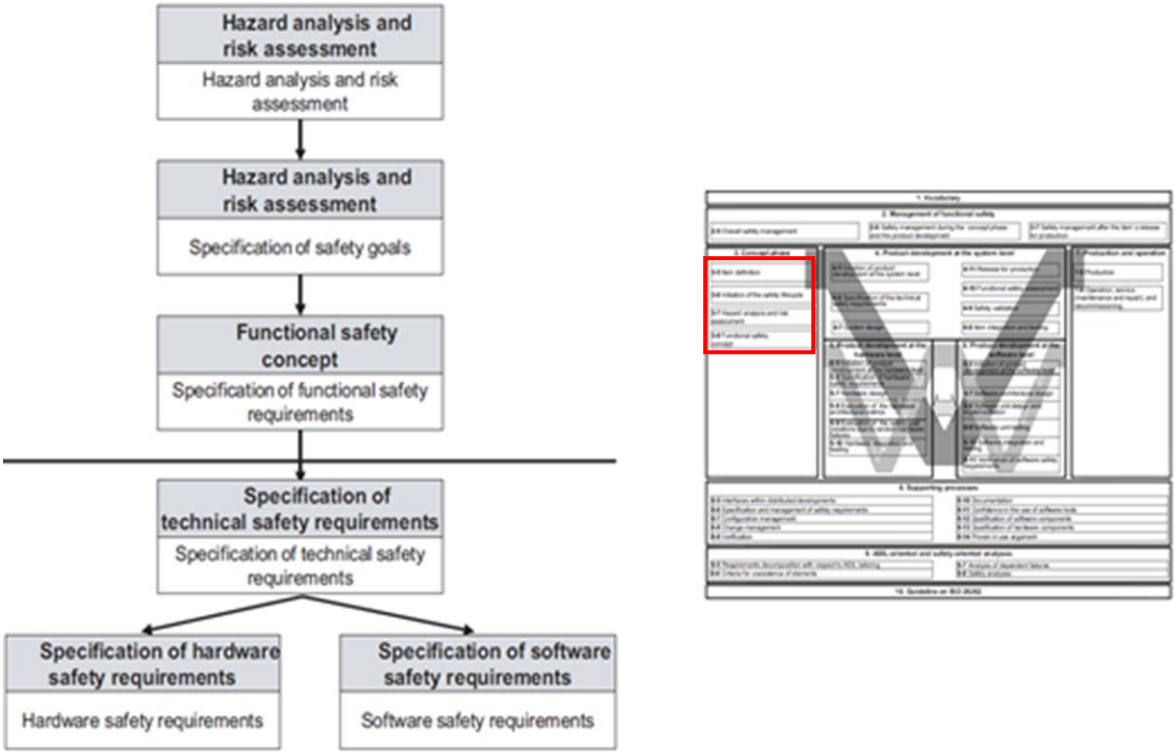


Figure 2-1. From hazard analysis and risk assessment to requirements specification according to ISO26262

After the technical safety requirements have been specified, the system is designed. During system design, the system architecture is established, the technical safety requirements are allocated to hardware and software, and, if applicable, to other technologies. In addition, the technical safety requirements are refined and requirements arising from the system architecture are added, including the hardware-software interface (HSI) [CEN 11, ISO 11, IEC 10, IEC 11]. Depending on the complexity of the architecture, the requirements for subsystems can be iteratively derived. At this level, safety analyses [LAP 96, LAD 03] are also performed in order to identify measures to avoid systematic failures and measures for controlling random hardware failures during operation.
Parallel to the system development and safety assessment activity, a safety case [KEL 03] is created incrementally compiling the work products that are generated during the safety lifecycle.

2.1.1.3 Safety case and safety arguments

The acceptability of system safety, primarily based on the sufficiency, satisfaction and traceability of the safety requirements and the absence of dangerous faults, explicitly requires the submission of a safety case. A safety case is *"a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment*" [MOD 07]. ISO26262 defines it as an *"argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development*" [ISO 11]. Through it, the evidence supports a safety argument which substantiates claims regarding the safety of the system in a given operating environment.

2.1.2 SPICE standard

The Automotive SPICE (Software Process Improvement and Capability Determination) [AUT 10a, AUT 10b] is an international standard based on ISO/IEC 15504 [IEC 11] and used in major worldwide automotive firms, as a *"framework for the assessment of processes*" [IEC 11]. In fact, it was developed in part to provide a basis for linking the results of model evaluation process improvement and assessment methods [HOE 08].The automotive SPICE can be considered as representative of software process assessment models since assessors assign ratings to indicators and metrics that measure the capability of software processes. The standard presents a two-dimensional reference model for the maturity models which specifies requirements for Process Reference Models (PRM) [AUT 10b] and Process Assessment Models (PAM) [AUT 10a] (see Figure 2-2). This reference model consists of some major components, namely: some lifecycle processes from different process categories for the process dimension, and six capability levels for the capability dimension.

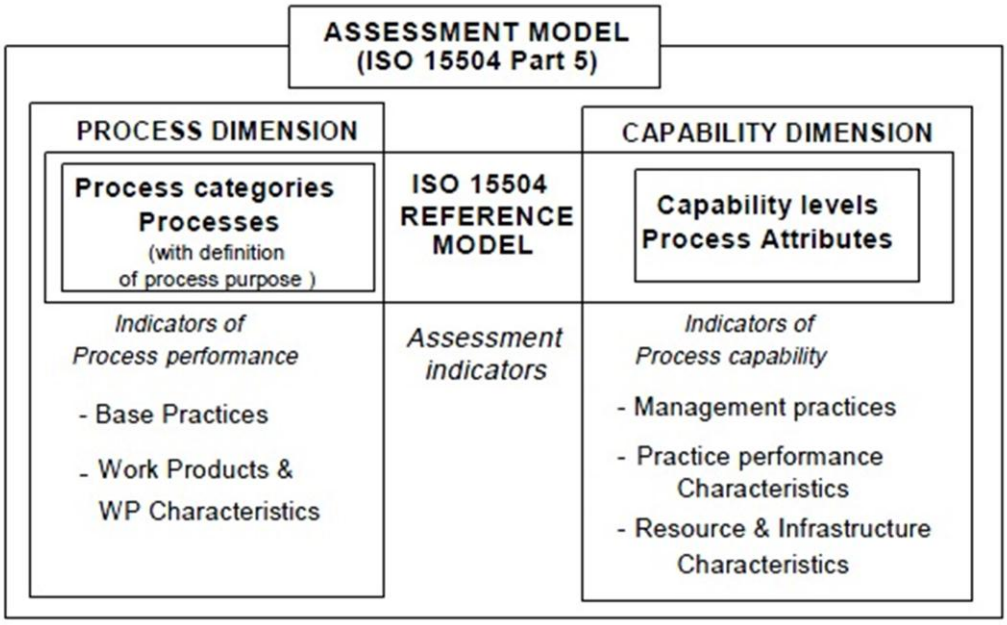


Figure 2-2. ISO 15504 assessment model

2.1.2.1 Reference model

The Automotive SPICE Process Reference Model (PRM) (see Figure 2-3) with the associated process attributes in ISO/IEC 15504-2 is derived from ISO/IEC 12207 AMD1: 2008 [ISO 08] and ISO/IEC 12207 AMD2: 2008 [ISO 08]. It provides a common basis for performing assessments process capability, allowing for the reporting of results using a common rating scale. The set of processes provided are classified into Process Categories at a first level, and within a process category, in Process Groups according to the type of activity they address. The processes included in the same group have a logical relationship as their capabilities are related and contribute to an additional problematic.

There are three Process Categories: Primary Life Cycle Processes, Organizational Life Cycle Processes and Supporting Life Cycle Processes. The Primary life cycle processes consists of *"processes that may be used by the customer when acquiring products from a supplier, and by a supplier when responding and delivering products to the customer including the engineering processes needed for specification, design, development, integration and testing"* [AUT 10b]. Among them, the Engineering process group (ENG) is one that directly elicits and manages the customer's requirements, specifies, implements, or maintains the software product and its relation to the system.

Each process is described in terms of a purpose statement. These statements contain the unique functional objectives of the process when performed in a particular environment. A list of specific outcomes is associated with each of the process purpose statements, as a list of expected positive results of the process performance.

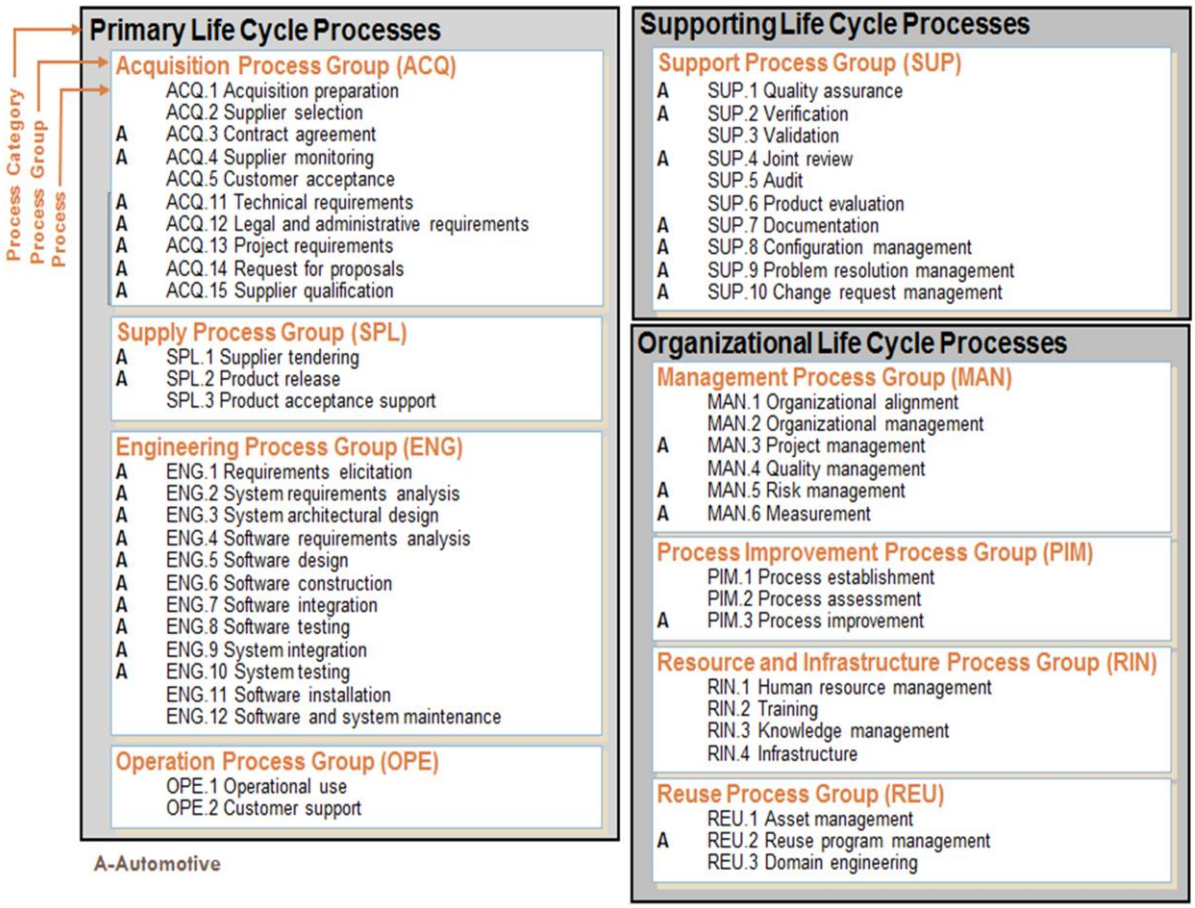


Figure 2-3. SPICE reference model

The Automotive SPICE Process Reference Model (PRM) is used in conjunction with the Automotive SPICE Process Assessment Model (PAM) when performing an assessment [AUT 10a].

2.1.2.2 Assessment model

The Automotive SPICE Process Assessment Model must be considered when interpreting the intent of the Automotive SPICE Process Reference Model. The model is based on the principle that the capability of a process can be assessed by demonstrating the achievement of process attributes on the basis of evidence related to assessment indicators. Assessment is defined as *"a review of a software organization to provide a clear and factual understanding of the organization's state of software practice*" [HUM 89]. The Process Assessment Model indicators are used as a basis for collecting the objective evidence that enables an assessor to assign ratings (see Figure 2-4).

N	**Not achieved**	**0% - 15%**	**There is little or no evidence of achievement of the defined attribute in the assessed process.**
P	**Partially achieved**	**16% - 50%**	**There is some evidence of an approach to, and some achievement of, the defined attribute in the assessed process. Some aspects of achievement of the attribute may be unpredictable.**
L	**Largely achieved**	**51% - 85%**	**There is evidence of a systematic approach to, and significant achievement of, the defined attribute in the assessed process. Some weakness related to this attribute may exist in the assessed process.**
F	**Fully achieved**	**86% - 100%**	**There is evidence of a complete and systematic approach to, and full achievement of, the defined attribute in the assessed process. No significant weaknesses related to this attribute exist in the assessed process.**

Figure 2-4. SPICE rating scale

There are two types of assessment indicators: process capability indicators, which apply to capability levels 0 to 5 and process performance indicators, which apply exclusively to capability level 1.

The six capability levels incorporate nine process attributes. Process attributes, applicable to all processes, are *"features of a process that can be evaluated on a scale of achievement, providing a measure of the capability of the process"* [AUT 10a]. Capabilities associated with their process attributes (see Figure 2-5) refer to the ability of the organization to produce these products predictably and consistently.

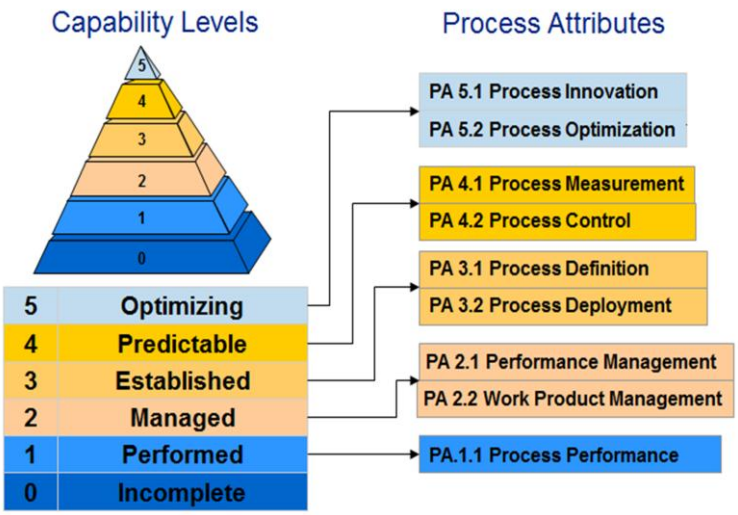


Figure 2-5. SPICE capability levels and Process Attributes (PA)

The process attributes in the capability dimension have a set of process capability indicators that provide an indication of the extent of achievement of the attribute in the instantiated process. These are Generic Practice (GP) and Generic Resource (GR).

As additional indicators for supporting the assessment of a process at Level 1, each process in the process dimension has a set of process performance indicators which is used to measure the degree of achievement of the process performance attribute for the process assessed. The process performance indicators are Base Practice (BP) and Work Product (WP) (see Figure 2-6).

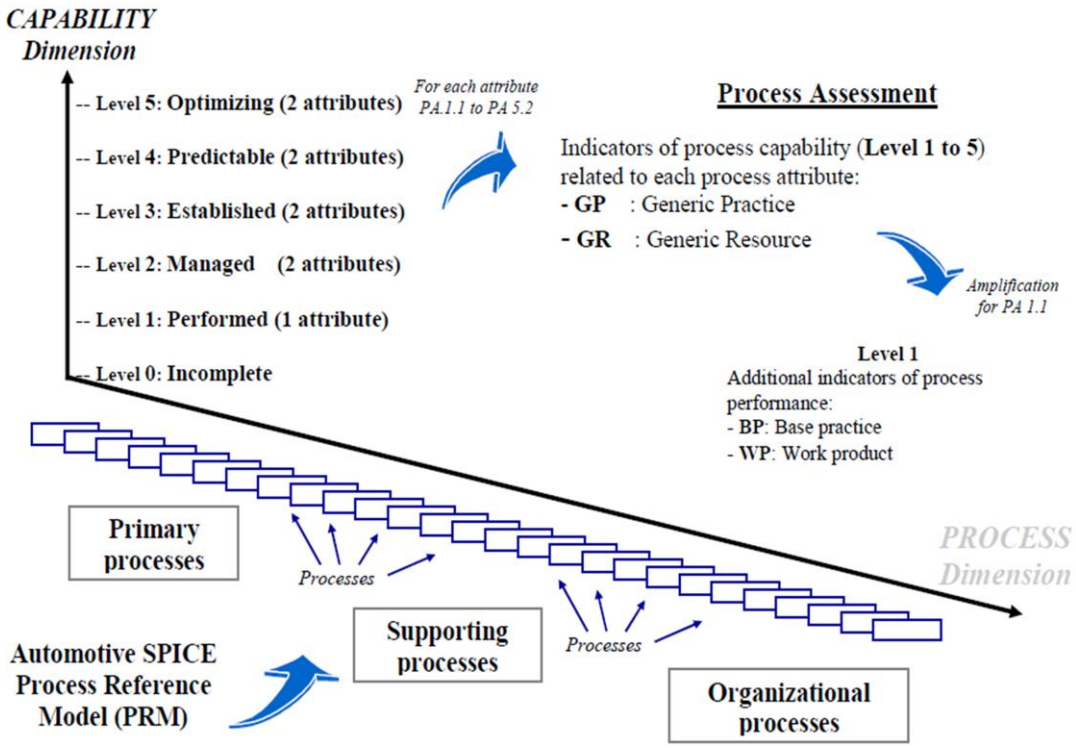


Figure 2-6. SPICE model scope

Evidence of performance of the base practices, and the presence of work products with their expected work product characteristics, provide objective evidence of the achievement of the purpose of the process.
We rely in this thesis on the HIS Automotive SPICE [HIS 12], a basic subset of processes (named HIS Scope), which has been defined on top of Automotive SPICE as a selection of a standardized assessment method mostly appropriated for determining suppliers software process capability and related to the sub models of the V-Model. The HIS scope (see Figure 2-7) uses the same assessment model as Automotive SPICE; only the Process reference model is reduced. Hereafter, we will refer to this subset as SPICE.

Engineering Process Group	
ENG.2	System requirements analysis
ENG.3	System architectural design
ENG.4	Software requirements analysis
ENG.5	Software design
ENG.6	Software construction
ENG.7	Software integration
ENG.8	Software testing
ENG.9	System integration
ENG.10	System testing

Support Process Group	
SUP.1	Quality assurance
SUP.8	Configuration Management
SUP.9	Problem resolution management
SUP.10	Change request management

Management Process Group	
MAN.3	Project management

Acquisition Process Group	
(optional)	
ACQ.4	Supplier Monitoring

Figure 2-7. HIS Automotive SPICE

We have presented the two standards we need to deal with. In next chapters, their impact with respect to certification is defined. Firstly certification shall be defined.

2.2 Certification approaches

Certification has received much attention and has become increasingly popular during these last years [STA 01]. ISO defines certification as *"a procedure by which a third party gives written assurance that a product, process or service conforms to specified characteristics".* In industry, it is a voluntary act that can give companies a competitive advantage. Certification must thus be understood as *"the process of verifying a property value associated with something, and providing a certificate to be used as proof of validity"* [STA 01]. An independent certification body can provide an assurance of product quality by issuing such a certificate to companies and public authorities. For embedded software, two orientations concerning certification [VOA 07] were undertaken in practice to study the issue. The first was a product orientation [DER 11, YAH 10, DEN 05], which focuses on specifying and evaluating the attributes of product quality itself by means of a set of practices (tools, techniques and methods). The second was a process orientation [OJA 04, KNE 06], which focuses on continuously improving the development processes under the assumption that

organizations with more advanced methodologies produce higher quality products as well. Both product orientation and process orientation approaches are explained in more detail in this chapter.

2.2.1 Product and process oriented safety certification approaches in the automotive industry

The software certification by product quality approach is an acceptable approach which determines the quality of software operating in complex environments. It was stated that the ISO26262 standard assumes a product certification approach because it typically mandates the development and management of well-structured and well-reasoned safety arguments following a set of practices preconized. Indeed, for each activity needed to be performed during a system development, the standard explicit the suitable methods and tools to use together with the characteristics to reach in order to achieve a global quality. This information explicitly addresses the product in its specification, its design, its verification and its validation.
In contrast to the product approach to software quality, the emphasis on embedded product quality is often refocused on the development process of these products which is the *"process orientation*". In contrast to product orientation, this approach defines ***what to do***, without leaving full freedom on ***how to do***. The reasoning is based on the assumption that a 'quality process' results in a 'quality product'; in other terms quality is created during a development process. By improving and controlling the quality of the software development process, the quality of the product is expected to become more constant, with higher and better predictability. Creating quality embedded products is therefore often about managing the development process in a suitable way so that the product is improved by the process. The most well-known standards for evaluating the quality of the software development process are the Capability Maturity Model (CMM) [CMM 02] and the SPICE standard (ISO 15504) [IEC 11]. This thesis will focus on the HIS Automotive SPICE [HIS 12], a standard derived from ISO/IEC 15504 to comply more precisely with automotive artifacts.

2.2.2 Gaps in certification approaches

Product quality is a key measurement of the software process. It provides *"a clear record of development progress, a basis for setting objectives, and a framework for current action"* [HUM 89]. It is the basis on which the software organization generates products. Even though industry has reported many successes with the process orientation, traditional process quality approaches do not work overall because questions about the impact of each specific part of the process on product quality remain unanswered. In fact, due to the complexity and abstractness of software, software engineering processes very often cannot be clearly traced to the product; direct links between process and product quality is then hard to prove. The statement: *"a good software development processes guarantees the excellent quality of product"* is mainly an assumption. The focus on the process should be complemented with a focus on the product.
On the other hand, the product oriented approach also has its weaknesses. The main claim of this approach is the specification of product quality. However, this specification of product quality is not always available, nor is it easy to specify: Is it the software code or a set of the workproducts defined during the safety lifecycle? Is it the safety case? The fundamental limitation of such product-based certification nevertheless lies in the observation that good tools, techniques and methods do not necessarily lead to the achievement of a specific level of integrity of an ASIL (e.g. as defined by a target failure rate) [HAB 06, MAI 08]. Another issue for product oriented software

quality creation is that the product quality must be created incrementally during system development and safety assessment. Then, periodically verifications and improvements are necessary along the path of development. This refers to the necessity of a process-oriented approach towards software quality.

2.3 Conclusion

Certification is currently a significant requirement in the embedded products development domain. Another common point in this area is the increasing importance of safety in the development of these systems [MAC 12]. In the automotive industry, the ISO26262 standard reinforces this fact. Stakeholders in the field are now preoccupied with aligning their projects with the recommendations of this standard. They must also deal with the precedent demands regarding deployment of Automotive SPICE in their companies. However, ISO26262 and Automotive SPICE follow two orientations which have proved difficult to manage together: the product quality certification approach and the process quality certification approach. Our conclusion is that it is clearly not a matter of choosing between a product or process orientation [KEL 08]; as both need to be applied together. Thus, an assessment of end product software must be dependent from the development process.

The aim of this thesis is to develop an instrument to measure process capability of a specific engineering organization that develops safety systems and that focuses on software certification by using both the end-product quality approach and the development process approach in the automotive domain. Working within the framework of model-based development is intended, as the industry is increasingly interested in this paradigm. Model-driven engineering is the topic of the next section.

3

Product and process in model driven engineering framework

3 Product and process in model driven engineering framework

3.1 Introduction

The aim of this chapter is to place the objectives of this thesis in the context of existing works. In the automotive field, the safety critical embedded systems definition is becoming increasingly complex, in view of numerous constraints: meeting tight schedules and reducing costs, providing better safety, improving and increasing performance in terms of intelligence, capacities, features, innovations, etc... One of the main issues in this context is the difficulty of handling a multitude of goals (of different actors) during the development of automotive products when not only trying to build a compliant system but also the *"right"* system. Model-driven engineering appears to be a good route for seamlessly integrating different viewpoints [BEZ 02]. Consideration of requirements may ensure that user needs are satisfied... Thus, the first chapters of this section examine the existing approaches of requirement management in a model-driven engineering framework for the automotive industry.

The need to respect maturity models is a further consideration. The assessment of the effective quality must be possible to demonstrate the quality both at product and process levels in a certification goal [DEC 09]. The second part of the section presents a state of the art on process and measurement concepts led by the models for the development of embedded systems. Furthermore, all the literature review is analyzed with the prerequisite to be compliant with ISO26262 standard.

3.2 Requirement engineering in a model driven engineering framework

Interest in using model-driven engineering [JIA 04, CAB 08] (also known as model-based engineering) has been steadily increasing, especially in the transport domain as the computer part of systems is currently progressing and many traditional System Engineering (SE) techniques are no longer adequate. System engineering mainly consists of requirements management and an architecture definition which responds to the requirements specified. This is an important discipline when ensuring that the system developed is not only compliant, but also the well-defined, in particular concerning the requirements engineering phase. This means that one of the prerequisites for successful software development is, the management of the requirements engineering process. In fact, according to Boehm, approximately 60 percent of all errors in system development projects originate during the requirements engineering phase [BOE 81]. Bell and Thayer observed that inadequate, incomplete, inconsistent or ambiguous requirements are numerous and have a critical impact on the quality of the resulting software [BEL 76]. Alford and Lawson state that the major cause of insufficient product quality is in non-stated requirements [ALF 79]. For many years these references have consistently supported the view that poor requirements are the major cause of software problems, resulting in project failures and/or time and budget overruns [VAN 09]. It can therefore be concluded that the accuracy with which a development process is able to create a quality product is thus dependent on the accuracy of the product quality requirements [DAV 88]. Nowadays, this statement is still relevant, reinforced by SPICE as the ISO26262 standards as they address some key process areas for requirements management as a priority of their engineering activities. Requirements engineering therefore deserves to be a working area for the specification of

product quality in our work. Before focusing on model-driven engineering, the outlines of requirements engineering will be specified.

3.2.1 Refined definition of requirements engineering

Requirements Engineering (RE) is concerned with identifying the purpose of a software system, and the contexts in which it will be used. It acts as the bridge between the real-world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies [RE 01]. It is a sub-discipline of system engineering that involves methods, rules and processes, to establish and maintain a single repository that is refined and completed during development and is maintained throughout the system lifecycle. According to [VAN 09]:" *Requirements engineering is the discipline concerned with discovering, understanding, formulating, analyzing and agreeing on what problem should be solved, why such a problem needs to be solved and who should be involved in the responsibility of solving that problem.*" It includes the tasks of determining the requirements for a new or altered system, taking into account the sometimes conflicting demands of various stakeholders, including those of end users. The core activities to meet these ends are *Requirements Elicitation, Requirements Analysis, Requirements Specification, Requirements Verification and Validation, and Requirements Management* [POH 96, POH 11]. Although they appear to be individual tasks, these steps cannot be strictly separated and executed sequentially in a strict linear progression as in the Royce waterfall model [ROY 87, SOM 10]: some requirements are implicit in the work practices, while others can only arise when design solutions are proposed. Often, the elicitation intersects and iterates with the analysis, specification and validation steps. This may reflect more a spiral model [BOE 78]. The process is repeated cyclically, because, even after the requirements are assessed at baseline, it is probable that they will change at least once during development. It is also highly likely that they will change immediately after development.

In practice, in software development, the RE is commonly still based on the V-model [MCD 84, INC 07]. In fact, it is not just for developing software, but for developing systems that include both software and hardware. The V-model can be considered as an extension of the waterfall created to overcome its reactivity problem. It is also inspired by the spiral model because it demonstrates the relationships between each phase of the development life cycle and its associated testing phase. This places the requirements at the beginning next to the product's operation at the end, and the design next to verification [MOO 01, FOR 05].

Much literature is available on RE and many have been defined according to different objectives to be reached.

3.2.2 Requirements Engineering Processes

Among existing works concerning RE which are in accordance with our goal, we can cite goal-oriented RE. Goal orientation in RE is motivated by the need to clarify requirements when elicited from users, especially since at the early stages of system conception requirements are unclear or unknown. Goal construction, decomposition and refinement offers a natural mechanism for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying different levels of requirement concerns and the corresponding satisfaction arguments [VAN 08]. Lamsweerde through the KAOS methodology [VAN 08, VAN 09] prescribes a detailed process for goal-orientation in RE, which assists in building a multi-viewpoint model. The methodology uses a multi-view graphical language for system modeling [SCH 02], a lightweight formalism for model

specification [MAD 09, OMG 11a, OMG 11b], an optional real-time temporal logic for model analysis [DEL 04], a systematic method for model elaboration [LET 05], and various dedicated techniques for goal refinement and operationalization [TRA 04], conflict management [LET 04], hazard analysis [PON 07], agent responsibility assignment, goal mining from scenarios [DAM 05], including guidelines, etc… Such use is based on multi-view activities showing how goals, objects, agents, operations, domain properties, and scenarios are inter-related in the *"system-as-is*" and the *"system-to-be*". Requirements Engineering with Scenarios for a User-centered Environment (RESCUE) process [MAI 96, MAI 04] may be used complementarily with goals-based requirements engineering. It is a process for requirements development and management which includes several activities such as system goal modeling, activity modeling, creative design workshops, use-case modeling, systematic scenario walkthroughs and impact analysis.

As we are interested in safety management, the Cleanroom software engineering reference process can also be considered. It provides a theory-based, team-oriented process for development and certification of high-reliability software systems under statistical quality control [PRO 99]. One of the principal objectives of the Cleanroom method is to develop software so that exhibit zeros failures in use; the emphasis is on rigorous engineering and defect prevention, rather than defect removal. The overall approach combines mathematically-based methods for (software) requirements specification, design, verification together with statistically-based methods for usage-based testing to certify software fitness for use [LIN 96].

LaQuSo[1] (Laboratory for Quality Software) defines another certification model which is the Software Product Certification Model [HEC 08]. This is a product certification methodology defined as a model which can be used to certify products after their creation. The model also allows certification of software products in all life cycle stages as it indicates which elements and relations should be present in the product when it is being created, with the applicable rigor for the criticality of the software, up to formal verification for the most critical products [HEC 06]. The model concepts are defined through six Product Areas with each area that can be further divided into subparts known as elements [HEC 10].

3.2.3 Discussion about requirements engineering processes

Being a linear model, the waterfall model is very simple to implement. The project requires the fulfillment of one phase, before proceeding to the next. This is also its biggest disadvantage however, because in each stage every phase requirement is frozen and impossible to modify after. Consequently, small changes or errors that arise in the completed software may cause major problems. The spiral model overcomes this drawback. Adaptability in the design of this model in software engineering accommodates any number of changes that may occur during any phase of the project. Nevertheless, the model is quite complex and there is no proper control to move from one cycle to another cycle which can have an impact on the schedule and cost. Furthermore, as it is highly customized for every project, reusability is limited.

Goal oriented and RESCUE processes are interesting in the sense that they fit into the model-driven engineering paradigm. In this work, we do not wish to deviate from business practices and it is difficult to relate these two approaches to the classical V-cycle.

The Cleanroom and LaQuSo models deal with the certification issue which concerns this thesis. The Cleanroom model is based on formal methods which mean however, that it can be rejected because its implementation in real projects is not simple. The LaQuSo process model thus becomes the

1 http://www.laquso.com/

closest to our needs although it does not meet them completely. It is based on CMMI, and we prefer the SPICE standard which is similar. It checks some safety properties and our aim is to comply with the ISO26262 standard. It allows UML diagrams to be used [OMG 11a, OMG 11b] and our aim is to evolve in a model-driven engineering framework. We will therefore mainly be inspired by it, but also by others, to a lesser extent, to build our solution. In the subsequent chapters, some product and process modeling languages are described following this paradigm in the field of embedded systems development.

3.3 Product and process modeling languages

Companies which produce complex safety-critical systems are increasingly adopting model-based approaches to the development, assessment and assurance of E/E systems [HAB 10]. Because a major preoccupation in the automotive field involves certification support, we are particularly interested in how to use the modeling concepts in a system development environment to define safe and compliant systems. This implies concepts for product modeling and also for process modeling.

3.3.1 Product oriented modeling languages

The purpose of the product model is to support a more efficient and sound development process by providing a domain-specific level of development. It consists of the description of those aspects, concepts and their relations to a system under development explicitly dealt with during the development process and handled by the development tool [SCH 02]. Thus, it supplies the *"language"* to describe a product. Language is defined as a system of signs and rules used to express thoughts and to communicate. Each sign has a symbol and a meaning in itself and in relation to other signs [MAD 09]. When attached to the semantics of language, signs do not establish exclusive correspondence with external reality: we can use language in different contexts. To obtain models that can be interpreted, formalizing their expression is required, i.e. to define an *"expression language"*. In computer science, graphic representations generally using diagrams and symbols to represent common concepts and lines to their relationship [MAD 09] have reached a certain level of maturity. Furthermore, they also help to improve understanding of a complex problem, and communicate a system's function and the requirements. UML [OMG 11a, OMG 11b] and some of its profiles can be considered as an example of these languages and it has been used in many fields of expertise.

3.3.1.1 Unified Modeling Language

In the automotive industry, the OMG™ Unified Modeling Language (UML) [OMG 11a, OMG 11b] has long been used as the de-facto standardized general-purpose modeling language. Throughout the development cycle, it allows software abstract models to be created to specify, visualize and document facts related to the development of an object-oriented system, through two broad graph categories: structural diagrams and behavioral diagrams [OMG 11a] (see Figure 3-1). Nevertheless, although the language can effectively describe the structure, the behavior and the interactions of a system, it has the disadvantage of being too general. In fact, some researchers have shown that using only UML as a requirement specification language [KON 06] presents deficiencies [GLI 00]. To model an engineering system, a modeling language should be able to represent different facets and

describe all the specific techniques that it does not allow. It is not possible, for example, to distinguish the different types of traceability links between requirements.

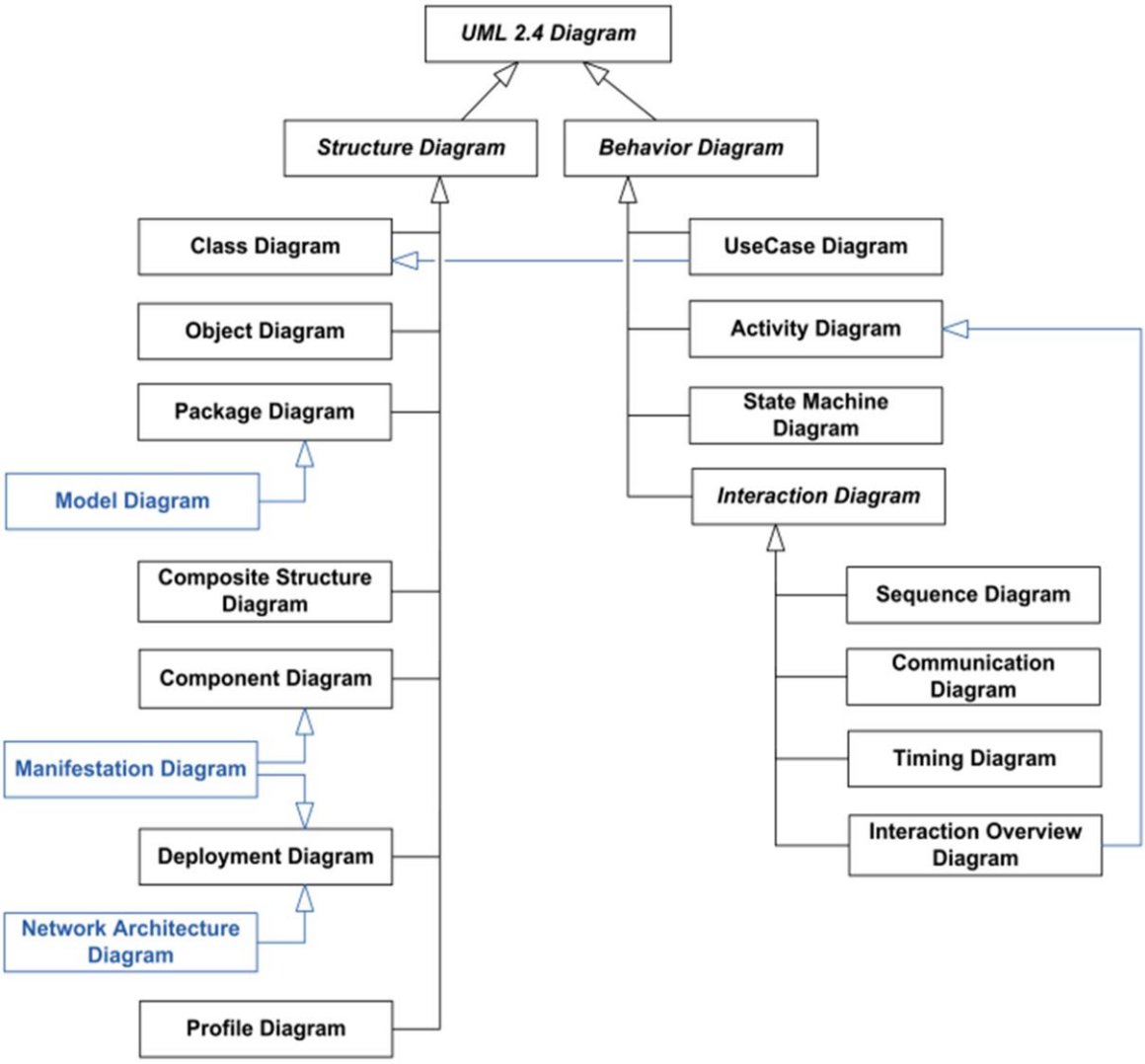


Figure 3-8. UML diagrams

This has resulted in many attempts to develop other modeling languages which can be more specific and which are designed to be completely adapted to a particular area. Those adapted to requirements engineering usually focus on a single point of discipline, either the early stages e.g. [GOL 07, YU 97, HEA 04], or traceability [LET 02], or only non-functional type requirements e.g. [ROD 06, ZHU 07]. SysML itself is such developed extension of this type.

3.3.1.2 SysML

To overcome the difficulty of general purpose of UML, the OMG™ System Modeling Language (SysML) [INC 07, OMG 10a] has been developed. It was quickly and widely adopted by the industry. This is a domain specific language that can support the analysis, design, verification and validation steps. SysML is defined as an extension of UML for systems engineering in the form of a UML profile [SEL 07] to benefit the user experience, and also for reasons concerning the reusability of tools. As an UML2.0 profile for systems engineering, some diagrams were removed and others changed in

order to focus on specific objectives in the field. In fact, SysML uses slightly modified versions of the class diagram and composite structure diagram that become the block diagram and internal block diagram respectively [OMG 10a] (see Figure 3-2). The advantage of working with SysML blocks instead of classes is that they have a vision centered on the idea of software, while blocks may represent different entities, as the software of hardware, data, people or facilities. Two new charts have also been introduced: the parametric diagram that can be used for performance analysis and quantitative analysis; and the diagram for requirements management purposes. The main particularity of the latter is the effective specification of requirements, thus offering the same fundamental concepts and information capabilities for requirements management as independent commercial tools. The specified requirements can therefore be directly allocated to system architecture designed with block diagrams and the same modeling language as in [BAL 06] used. Another advantage is that SysML provides tables of allowances that support flexible allocation of requirements, functional and structural allocation. UML in contrast, provides only limited support for tabular notations.

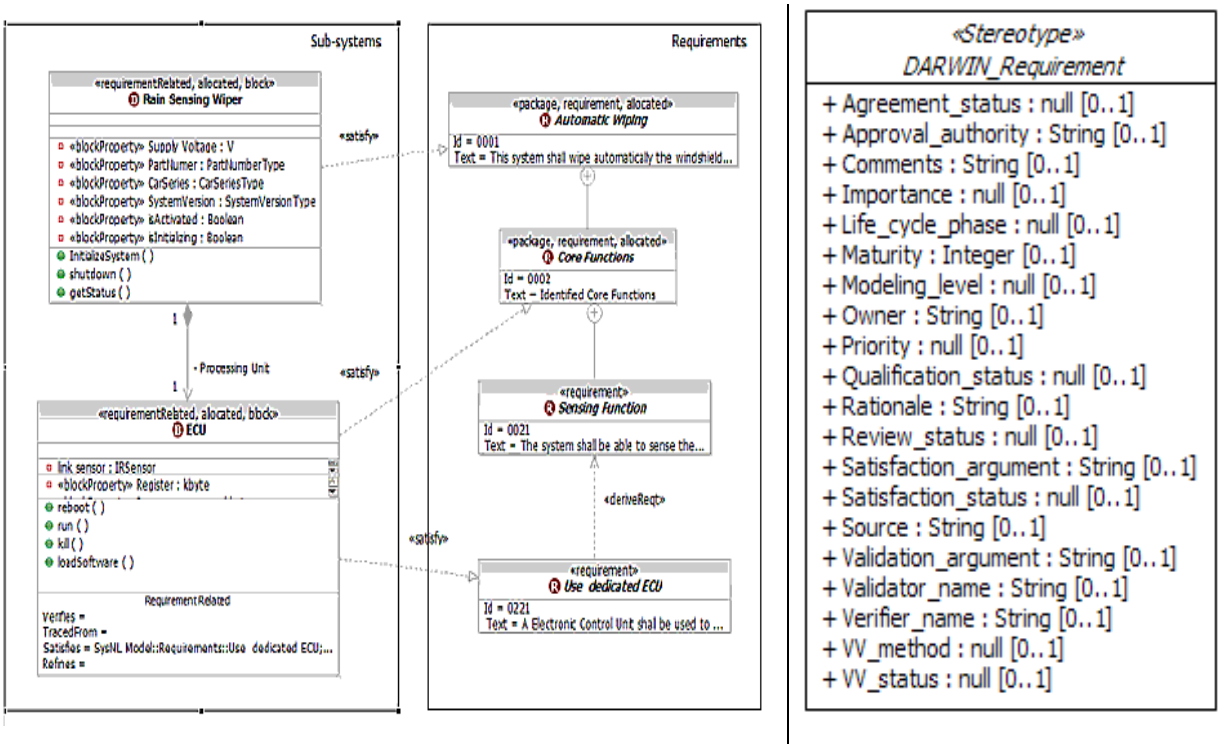


Figure 3-9. SysML example (left) and DARWIN requirement stereotype specification (right)

Recent modeling improvements and extensions have been made with the SysML UML-profile so that requirements can be better considered. The DARWIN approach for example [DUB 10, ALB 08], is a specific SysML-profile strictly focused on requirement management and its traceability (see Figure 3-2).

In the automotive engineering field, as in other fields, considerable effort has been made to develop domain-specific approaches. Existing approaches and accomplishments in the development of automotive embedded systems are AUTOSAR [AUT 12, SAN 08], EAST-ADL2 [ATT 10a, ATT 10b]. AUTOSAR is a referential used to create the E/E system architecture starting from the design-model. The objective is to create a basis for industry collaboration on basic functions while providing a platform which continues to encourage competition in innovative functions [AUT 12]. EAST-ADL2 is, for its part, a modeling language that supports all aspects of an engineering project for the

specification of systems or a system of systems. It is described in greater detail in the following chapter.

3.3.1.3 EAST-ADL metamodel

EAST-ADL is an initiative from the automotive domain defined as an Architecture Description Language (ADL) [FAU 08] to treat all information engineering techniques needed to support the development of electronics in automobiles [CUE 08, SER 08]. EAST-ADL is also a UML profile [OMG 11a, OMG 11b]. The basic concepts such as classes, compositions and associations are used to define the domain model, with more stereotypes, properties and constraints. The language requires system descriptions on several abstraction levels, from top-level user features to tasks and communication frames. As with all ADLs, it uses a precise and common vocabulary for actors to work around the specification relating to architecture. For instance, it specifies the components of the automotive architecture in an abstract way without going into implementation details, explicitly defines the interactions between system components and provides modeling support to help designers to structure and compose the elements [DUS 02]. The language is expressed through five abstraction levels: *Vehicle Level, Analysis Level, Implementation Level, Operational Level*. These levels represent the different phases of an engineering process. The Figure 3-3 below is an example of a requirement hierarchy and its allocation in the architecture on the three first abstraction levels.

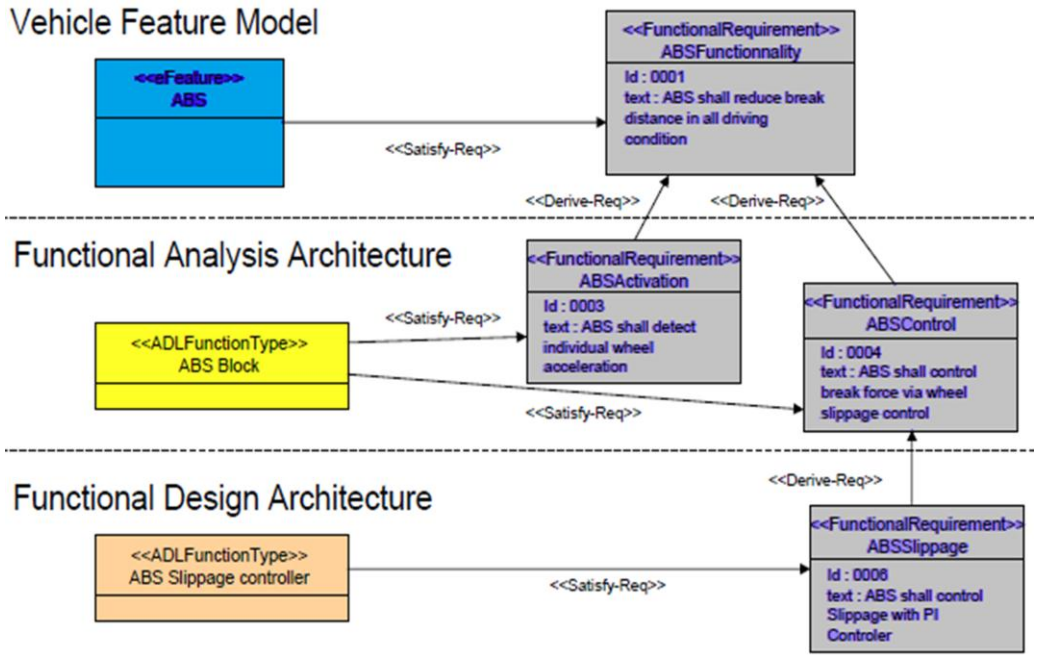


Figure 3-10. EAST-ADL example on three abstractions levels: vehicle, analysis and design

In the Vehicle Level, the electronic characteristics of the vehicle such as brakes, wiper, collision prevention, etc. are organized. A variability mechanism supports the definition of rules to include different vehicles, allowing product line architecture [POH 05]. At Analysis Level, the *Functional Analysis Architecture* layer describes the features that realize the functionalities. Functionalities are related to the environment through sensors and actuators. At Design Level, parts of the design are shown in the *Functional Design Architecture* layer, where the functions defined at Analysis Level are decomposed and allocated. A *Middleware abstraction*, allows interfaces to be defined. The Design

Level also includes a *Hardware Architecture Design* layer. Interfaces and interactions can be defined and simulated in existing tools, such as Matlab Simulink [MAT 12]. In fact, this level of abstraction is strongly influenced by the modeling in Simulink. It allows such architectures to be analyzed with respect to completeness, consistency, ambiguity, and performance for the generation of software systems [ABD 00]. At Implementation and Operational Level, the implementation architecture, reusable code and software compatible with AUTOSAR are defined and deployed at Operational Level.
EAST-ADL2 also involves the expression of non-structural aspects of the system, such as requirements based on SysML concepts.

3.3.1.4 Limitations of product modeling languages

Some modeling languages described above have some drawbacks that already allow us to dismiss them in our analysis, despite their undoubted potential.
UML is a modeling language which is widely used today. It is a very accomplished formalization and non-proprietary modeling language, based on a graphical annotation. It also describes the structural integration of different views of a product as the behavioral interactions within the system. However, semantical relations exceeding the structural relations of the conceptual model are missing. Since UML is focused on the conceptual product model, it must be related to development activities. In our case, it may also be too general for an optimal and appropriate usage because it we want to take into account the specificities of the automotive world.
Similarly, SysML is not designed for modeling domain-specific applications unlike Escalona et al [ESC 06] for example, which focuses on the web systems domain, although it already specializes in the concept and requirements of the system with the definition of requirements and parametric diagrams. The DARWIN [ALB 08] profile seems closer to our needs for this purpose in the sense that it already incorporates several recommendations of the ISO26262 in the properties of a requirement. One example is the requirement *"status*" which can have the values *"proposed, assumed, agreed or reviewed*" and which does not appear. But, it is not sufficient because, following the safety standard: *"Safety requirements shall have the following attributes: a) unique identification remaining unchanged throughout the safety lifecycle; b) status; and c) ASIL*". As this profile does not consider the system architecture, its usage must be combined with other specific profiles and languages dedicated to system architecture. Furthermore, requirement engineering must also allocate requirements to design elements.
EAST-ADL2 appears to be the best candidate so far. Its greatest advantage is that it offers an open architecture dedicated to the development of automotive systems. It can automatically generate AUTOSAR components. Furthermore, it has the advantages of the languages above. At UML level, it integrates its graphical notation and completeness in terms of possible architectural views (structure, behavior, interaction) and sets the system for different levels of abstraction. At the behavioral level, it can integrate Simulink models. Variability is evenly presented in this profile. It also incorporates the notion of requirements specific to SysML. EAST-ADL2 tends to meet, although differently from the DARWIN profile, the ISO26262 standard for the automotive domain by covering a safety aspect. Its drawback is that does not satisfy the requirement to be compliant with standards in the automotive systems field. For example, some necessary quality characteristics and attributes needed in requirement specification stipulated in the ISO26262 are lacking.
Some other research and industrial initiatives for assessing RE are also being developed in the model-driven community. Each initiative focuses on part of our problem. In the CESAR project [CES 09] for example, one requirements management objective is to improve the current processes

dealing with requirements management and engineering in order to favor interchange in the development and supply chain and to facilitate the definition and identification of safety critical requirements in line with safety standards.

Inspiration is drawn from these different approaches to build a solution which is adapted to automotive needs and compliant with standards. Instead of trying to redefine a new profile from scratch, and thanks to the capability of profiles [SEL 07] to be composed and extended into a new profile in UML, we have chosen to reuse, integrate in a new profile and complete two complementary profiles: DARWIN for the requirement management and EAST-ADL2 for the system architecture design. This profile will be detailed in the next section.

As with other common weaknesses, these technical solutions cannot be easily associated with widely used RE techniques and tools that may exist and that offer very rich mechanisms of elicitation, refinement or decomposition of requirements, and that may be based on dependability techniques or fault management, for example. In fact, major actors in the automotive domain have a fairly efficient tool-based methodology to define and manage their requirements. Most of them use general office automation tools (like MS Excel™ or MS Word™) or commercial tools such as DOORS[2] [DIC 03] and Reqtify[3] [BUR 06, GEE 07]. These tools ensure the requirements development, i.e. their elicitation, their analysis, their specification and their validation. These tools also manage the traceability between requirements, but they do not manage elements of system architecture. Consequently, part of the traceability of architectural elements is still often hesitant or even nonexistent. The system representations are thus either just informal or ad hoc (e.g. the whiteboard approach); and, therefore completely uncorrelated to requirements activities. The documentation of the customer needs from the source specifications in a modeling environment is not ensured. Incomplete traceability of information between the source documentation and the modeling environment is then obvious. This is insufficient and a reason for change because satisfactory and complete requirement engineering must also allocate requirements to design elements according to automotive standards like Automotive SPICE *("The purpose of the System architectural design process is to identify which system requirements are to be allocated to which elements of the system."*) and ISO26262 (*"The implementation of the technical safety requirements shall be specified in the technical safety concept and the system design specification").* Even if tools like Reqtify offer some kind of traceability between different elements (requirements from DOORS and blocks for SysML models in ARTiSAN [CAS 10], for instance), these commercial tools do not manage elements from an arbitrary UML modeler. Given the large number of requirements that may be represented, the challenge is to implement simple process which easily documents the customer needs from source specifications in the required environment as well as requirements models without specifying them manually. Another downside is that the customer needs are likely to evolve over time, and consequently this task would need to be repeated many times. Therefore, the challenge is being able to import and export the requirements list from native customer specification documents without writing them manually as a model in a modeler. The Requirements Interchange Format, namely ReqIF [OMG 11c], can help to address this issue [CAS 10]. It is a standardized open, generic, non-proprietary and tool-independent format to support requirements exchange processes between different requirements management tools (see Figure 3-4).

2 DOORS, http://www.ibm.com/developerworks/offers/lp/demos/summary/r-telelogicdoors.html
3 Reqtify, http://www.geensoft.com/en/article/reqtify/

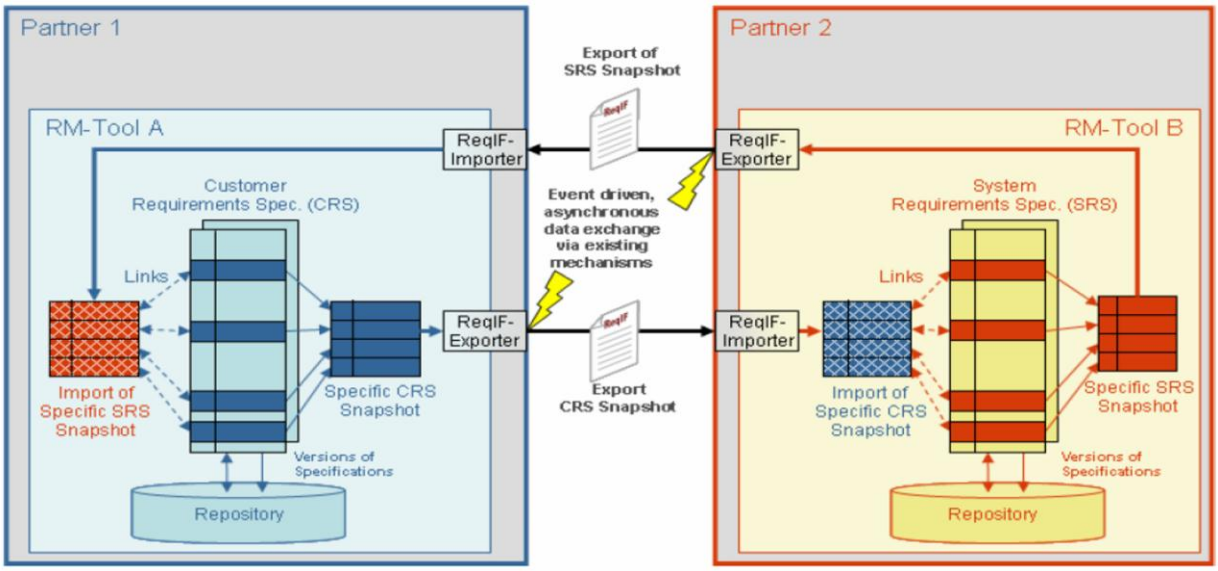


Figure 3-4. Example ReqIF exchange scenario

The contribution is implemented in Papyrus MDT[4] (Model Development Tools), a graphical dedicated tool developed in the Eclipse environment for modeling embedded systems. This tool can be used for many different applications. First, Papyrus provides a full implementation, efficient, robust and methodologically agnostic of UML2. Secondly, Papyrus is an open and flexible tool for defining and using domain-specific modeling Languages thanks to a very advanced implementation of the concept of UML profiles [SEL 07]. It supports not only SysML, and EAST-ADL2 profiles among others [LOR 11], but it also allows a new profile to be defined which corresponds to user needs in a friendly way.

3.3.2 Process oriented modeling languages

The vision process, which has been gradually built by different theories of organizations over the past half-century, is playing an increasingly important role within companies. In the field of embedded systems, the issues of reusability, autonomy and collaboration between software components have been extensively researched and led to a request for these processes to be formalized. [SCH 02] says that *"the justification of the product model is its application in the definition of a process model"*. In fact, today process modeling is often considered as a prerequisite to the design of dynamic systems. Specifically, process modeling means describing activities of a development process and their relations [SCH 02]. More formally, we can distinguish three abstraction layers in the process modeling domain: a metamodel level, a model level and an endeavor level [IS0 07a] (see Figure 3-5).

The metamodel includes the generic concepts to describe the processes in a model, i.e. the relationships and rules of concepts that are used to define a process model in a lower level of abstraction [IS0 07]. The process model, also known as methodology is *"The specification of the process to follow together with the work products to be used and generated, plus the consideration of the people and tools involved during a development effort"* [ISO 07a]. It is thus a description of a process which is specialized in a field or for an organization, as confirmed by [EAS 04]:"*A Process*

4 Papyrus MDT, http://www.eclipse.org/modeling/mdt/downloads/?project=papyrus

Model is an abstract description of how a particular organization normally conducts a collection of activities, focusing on resource usage and dependencies between activities".

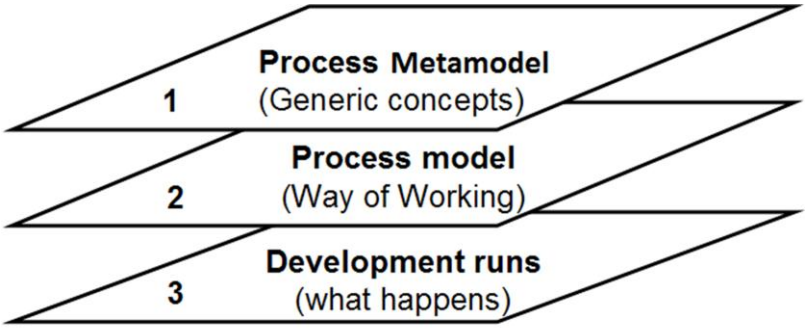


Figure 3-5. The process modeling abstractions layers

Method and Process must not be confused. In fact, methods focus on technical activities (i.e. the content of activities), whereas process models concentrate on management of activities (i.e. how activities can be measured and improved)."*A method is a prescription for how to perform a related collection of technical activities, especially where a set of techniques and notations are needed. A method offers guidance for how to use the notations and techniques together to achieve a particular goal. Methods tend to be specific in their applicability*" [EAS 04].

The lowest level is where the process model is instanced or enacted given that the enactment is the act of applying a methodology for some particular purpose [ISO 07a]. It is formalized by the term *process* as "*an enactment of a process model, describing the behavior of one or more agents and their management of resources*" [EAS 04]. The process model describes what should happen; a process is what actually happens in a particular project through the application of a process model [ISO 07a]. The process engineering domain emphasizes therefore, the idea of defining the metamodel as a set of classes from which process model chunks can be generated and then composed into a usable process [HEN 03]. This domain often resembles Method Engineering [RAL 01, BRI 96].

Much literature exists on process engineering. We are specifically interested in process language with graphical notation. These languages use technical diagrams with symbols associated with names that represent concepts and lines that connect symbols and represent relations and various other graphical annotations to represent constraints. By examining the metamodels associated with major languages of process modeling, we represent the key concepts in the Figure 3-6 below.

One of these process modeling languages, the Business Process Modeling Notation (BPMN) [OMG 09a, OMG 09b] is considered as the major credit rating business process, which means that the processes described in BPMN can be shared and easily understood by everyone. Nevertheless, as its main purpose is the modeling of business processes as its definition says, it is intended more for business analysts. More accurate languages related to embedded systems development are SPEM [OMG 08] and SEMDM [IS0 07a].

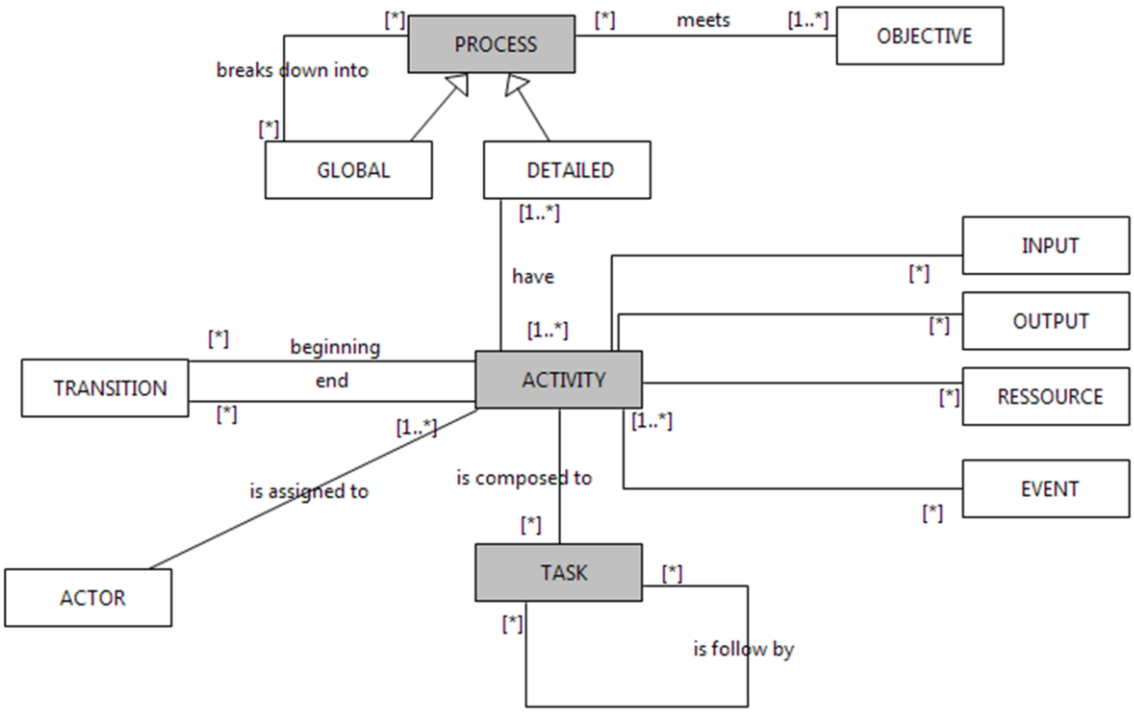


Figure 3-6. Process modeling language metamodel

3.3.2.1 SPEM

SPEM (Software Process Engineering Metamodel Specification) [OMG 08] has been proposed as a reference language for process modeling used to describe the process of software production and facilitate sharing of information. It is both a metamodel design process and a conceptual framework that provides tools and concepts for modeling, documenting, presenting, managing and making concrete development [OMG 08]. It is derived from UML aspects specialized to meet the process modeling goal.

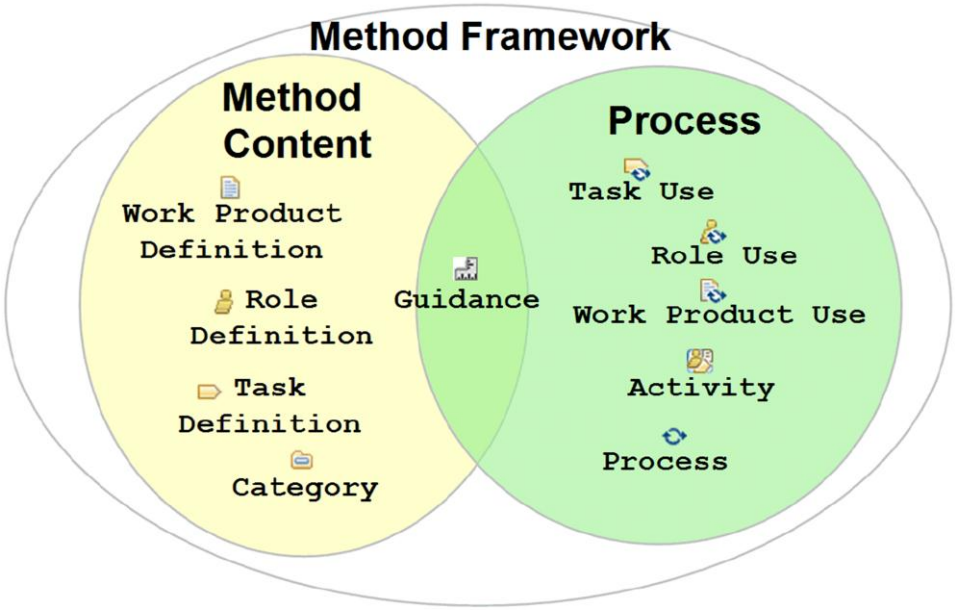


Figure 3-7. SPEM method framework

The Figure 3-7 above provides an overview of the key concepts defined in SPEM and how they are positioned to represent method content and process. *Method Content* is primarily expressed using *Work product definitions, Role definitions, Task definitions*, and *Guidance*. Guidance is defined in the intersection of Method Content and Process, because it can be defined to provide background for method content as well as for specific processes. On the Process side, these elements are used to represent processes in SPEM 2.0. The main element is the *Activity* which is used to define processes. Activities also manage references to method content. Other concepts exist, such as Lifecycle (Life Cycle) which defines and organizes the phases and iterations, etc...

3.3.2.2 SEMDM

SEMDM (Software Engineering Metamodel for Development Methodologies, ISO/IEC 24744) [IS0 07a] uses a new approach to defining methodologies based on the concept of powertype [ODE 94]. It establishes a formal framework for defining and extending development methodologies for software, business or systems, including three major aspects: the process to follow, the work products to use and generate, and the people and tools involved [IS0 07]. Its major aim is to deliver a highly generic metamodel (see Figure 3-8) that does not unnecessarily constrain the resulting methodologies, while providing for the creation of rich and expressive instances. It can be used to generate and substantiate a unique methodology or used in a method engineering context to create endeavor-specific methodologies. In using powertypes [GON 06a, GON 06b] as its substantiating conceptual architecture, it deviates from the common (for instance from OMG, Object Management Group) instantiation-linked multi-layer architecture replacing this with an architecture in three layers: endeavor (where people work; i.e. process level), method (where practices are determined, i.e. process model level), and metamodel (where practices are formally defined) [HEN 05].

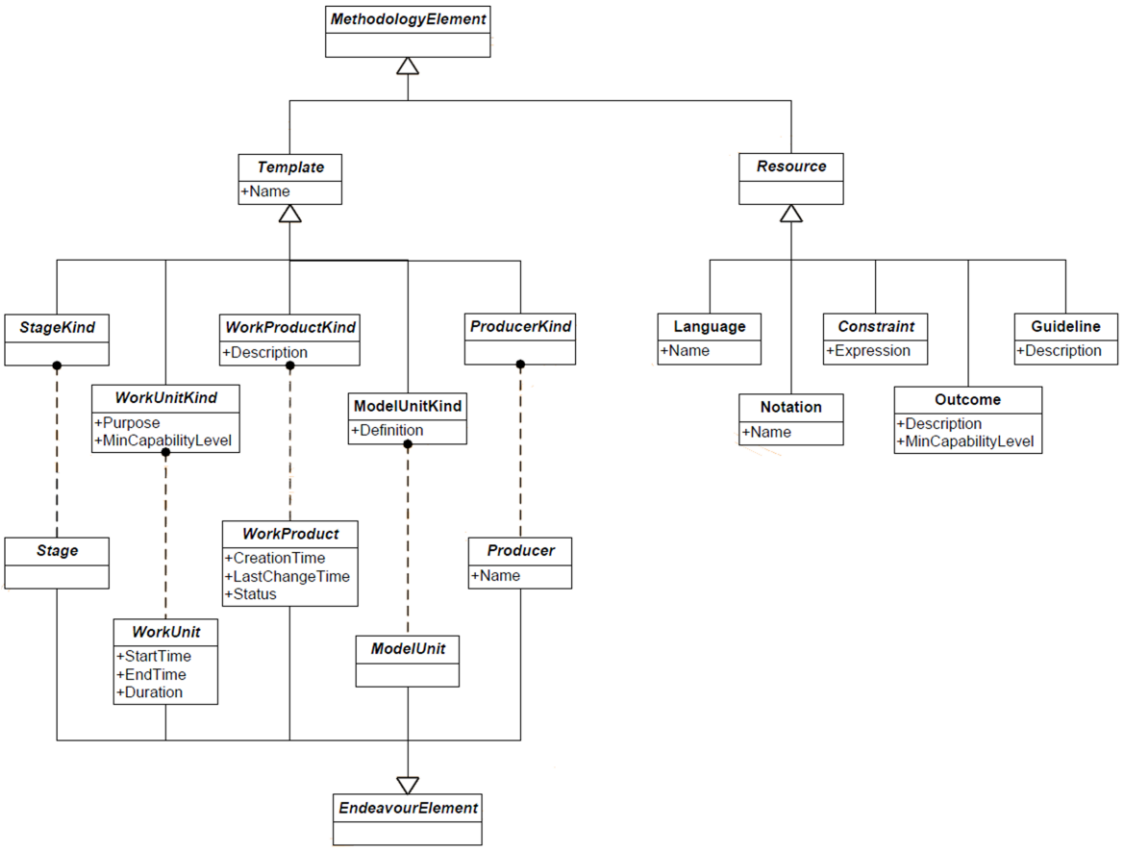


Figure 3-8. Abstract view SEMDM metamodel showing the core classes in the metamodel

3.3.2.3 Limitations of process modeling languages

Against the diversity of existing process modeling proposals, a reference process metamodel becomes necessary. Regarding our analysis, we opt for SPEM, primarily because it offers a graphical notation, which is easy to understand and very close to UML formalism. SPEM's goal is not only to support the representation of one specific development process or the maintenance of several unrelated processes, but to provide process engineers with mechanisms to consistently and effectively manage whole families of related processes. In contrast to SEMDM, SPEM is based on the UML Infrastructure [OMG 11b], focused on embedded and software-intensive systems and is able to communicate with a product modeling language. It is also possible to generate UML models from SPEM.

A further SEMDM disadvantage is that it is not supported by a tool. According to [ELL 10], this is because it does not provide a standardized notation, which clearly contradicts an easy to digest formalism. SPEM does not have this drawback. Eclipse Process Framework[5] (EPF) is the major example of its application. It is a rich application built entirely on an SPEM framework and a basic process. It describes modeling complex processes, of any kind, with the ultimate aim of disseminating examples of processes that support scalable iterative development practices. Methods and processes are structured according to the metamodel specifications using SPEM with UML diagrams and an associated XML schema. This functionality is its main advantage because it is useful to connect the process domain to the product model through UML.

3.3.3 Modeling of relationships between product and process

As the atom of a process, an activity describes how a product is changed; in this sense, relationships between process actions and product quality should be made explicit. Therefore, since process activities are defined as changes of product model instances, this means a process model can only be defined at the top of a product model [SCH 02]. The granularity of a product model also defines the expressiveness and thus the quality of the process model. Using both models, detailed development processes can be described: it assists in making a specific improvement program for specific projects. Nowadays, any modeling language supports the relationships between both models in a convenient manner. Although SPEM allows UML models to be generated from these processes, it is just another way to represent the process in a graphical way. This interpretation does not semantically link to the product. To fill this gap so as to integrate the process with the product perspective, an extension is made to the work products definition in SPEM [OMG 08] in the CESAR project [CES 09]. Looking at the conceptual system metamodel, the *WorkproductDefintion* and *WorkproductKind* concepts are specialized with an extension mechanism to introduce more artifact types through domain models (see Figure 3-9). The extension is implemented through the Permeter tool [CES 10a] and the editor allows the user to enter a work product definition based on an EMF metamodel [STE 08].

In the tool, an extension to the definition of the measurement framework for project monitoring is implemented. This forms the subject of the next chapter.

5 http://www.eclipse.org/epf/

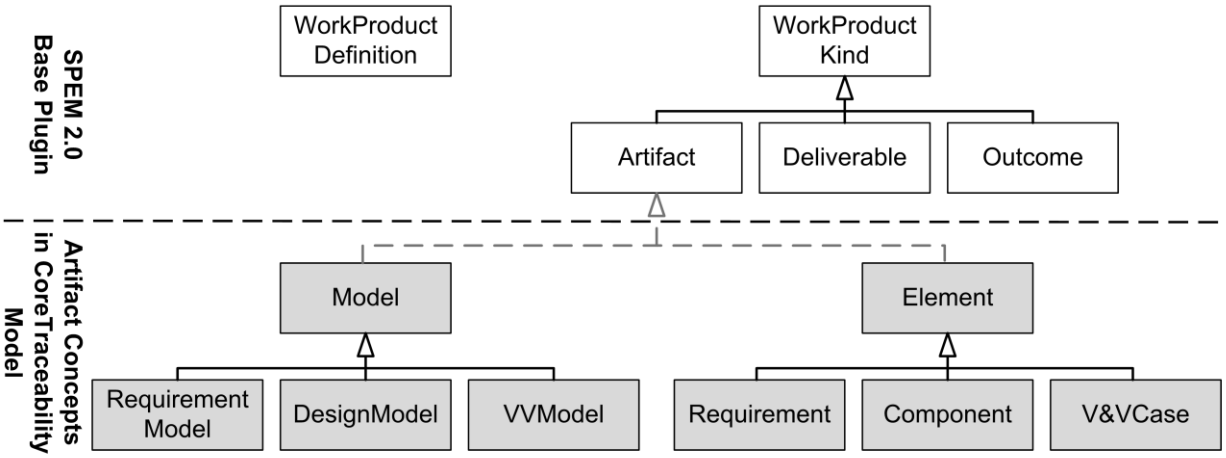


Figure 3-9. Integration of Process and Product Model Work Product Concepts

3.4 Quality of products and processes

A partial solution to the unknown relationship between process and product quality is provided by software process measurement, as it helps to determine the impact of process actions on product quality in a specific context. Measurement should be used for three purposes:

- Evaluating process-product relationships;
- Evaluating compliance of a product with the product quality specification;
- Evaluating the performance of a development process.

The process models constitute the starting point to analyze, enact and improve the processes both in their appearance and human application. Through measurement and analysis of the development process and its success in achieving the intended product qualities, an evaluation instrument becomes available. Next, background information is given about standards and research for measurement in embedded systems domains.

3.4.1 Literature review of the measurement domain

Measurement can be used for both a development process and a product. According to [ISO 07d],"*Software measurement is the continuous process of defining, collecting, and analyzing data on the software development process and its products in order to understand, control and optimize that process and its products*". Satisfying quality requirements entails software processes which must produce the expected results and be correctly defined. Any improvements made should be in accordance with the objectives of the enterprise [GAR 06b]. Successfully managing software processes is thus one of the main goals for software organizations in order to improve product quality, given the existing correlation between process and product quality [FUG 00]. The SPICE standard [AUT 10a, HIS 12, HOE 08] is in accordance with this statement.

3.4.1.1 Measurements of software processes

Measurement is an excellent mechanism for *controlling* a software process [FEN 97]. According to [ISO 07d], the measurement process is an adaptation of the Plan-Do-Check-Act cycle [DEM 86] commonly used as the basis for quality improvement. Two key aspects to be considered as the core in the software processes activities are *Plan* the Measurement Process, and *Perform* the Measurement Process.

According to [MCG 02], *measurement planning* begins with recognizing that a manager or engineer has a specific *information need*. Data that helps to satisfy the defined information need can be obtained by measuring many different software process elements and product characteristics, known as software *entities or attributes*. The *measurable concept* is an idea concerning the entities that should be measured in order to satisfy an information need. The measurable concept will be formalized as a measurement construct that specifies exactly what will be measured and how the data will be combined to produce results that satisfy the information need. A *measurement procedure* defines the mechanics of collecting and organizing the data required to instantiate a measurement construct. All of the applicable information needs, measurement constructs, and measurement procedures are combined into a *measurement plan* (see Figure 3-10).

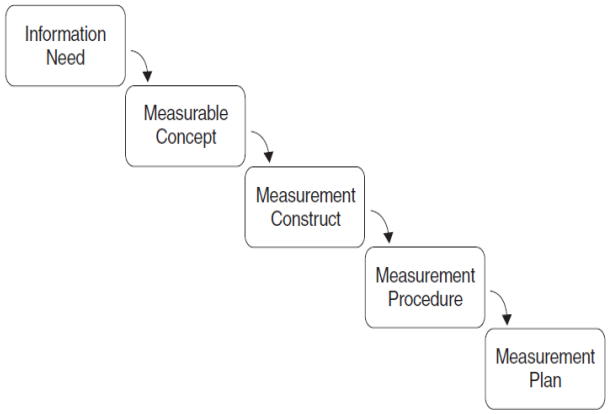


Figure 3-10. Evolution of an information need into a measurement plan

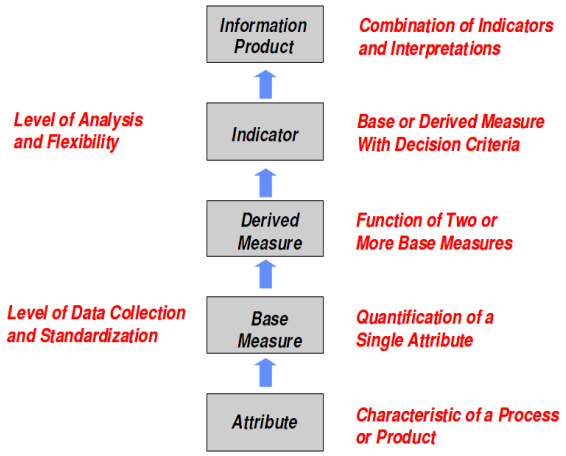


Figure 3-11. Levels of a measurement construct

Several methods are available which describe how to carry out process measurement planning. The GQM (Goal-Question-Metric) [BAS 90] paradigm is the most widely used paradigm in this field [PAR 96, VAN 02, BER 06] and its key concepts are:

- Processes (e.g., requirements development and management) have associated goals.
- Each goal leads to one or more questions regarding the accomplishment of the goal.
- Each question leads to one or more metrics needed to answer the question.
- Each metric requires two or more measurements to produce the metric.

Measurements are selected to provide the data that will accurately produce the metric necessary to answer the questions that determine goal accomplishment. Metrics are goals, indicators and measure [CES 10a]. According to [ISO 07d], to m*easure* is *"to make a set of operations having the object of determining a value of a quantitative or categorical representation of one or more attributes but measurements should have a clearly defined purpose"*. The standard classifies

measures as follows: *"base measure"*, *"derived measure"* and *"indicators"*. Figure 3-11 illustrates the different measurement levels [CAR 06, MCG 02, ISO 07d]. The *information products* are the outputs of the execution of the measurement plan which respond to the project information needs [MCG 02].

Perform the measurement consists in collecting and analyzing data. At each of the three levels of measures, additional information content is added in the form of rules, models, and decision criteria. Specific rules for assigning values, and defining the measurement methods, measurement functions, and analysis models associated with each level of measure are the activities conducted in this phase (see Figure 3-12).

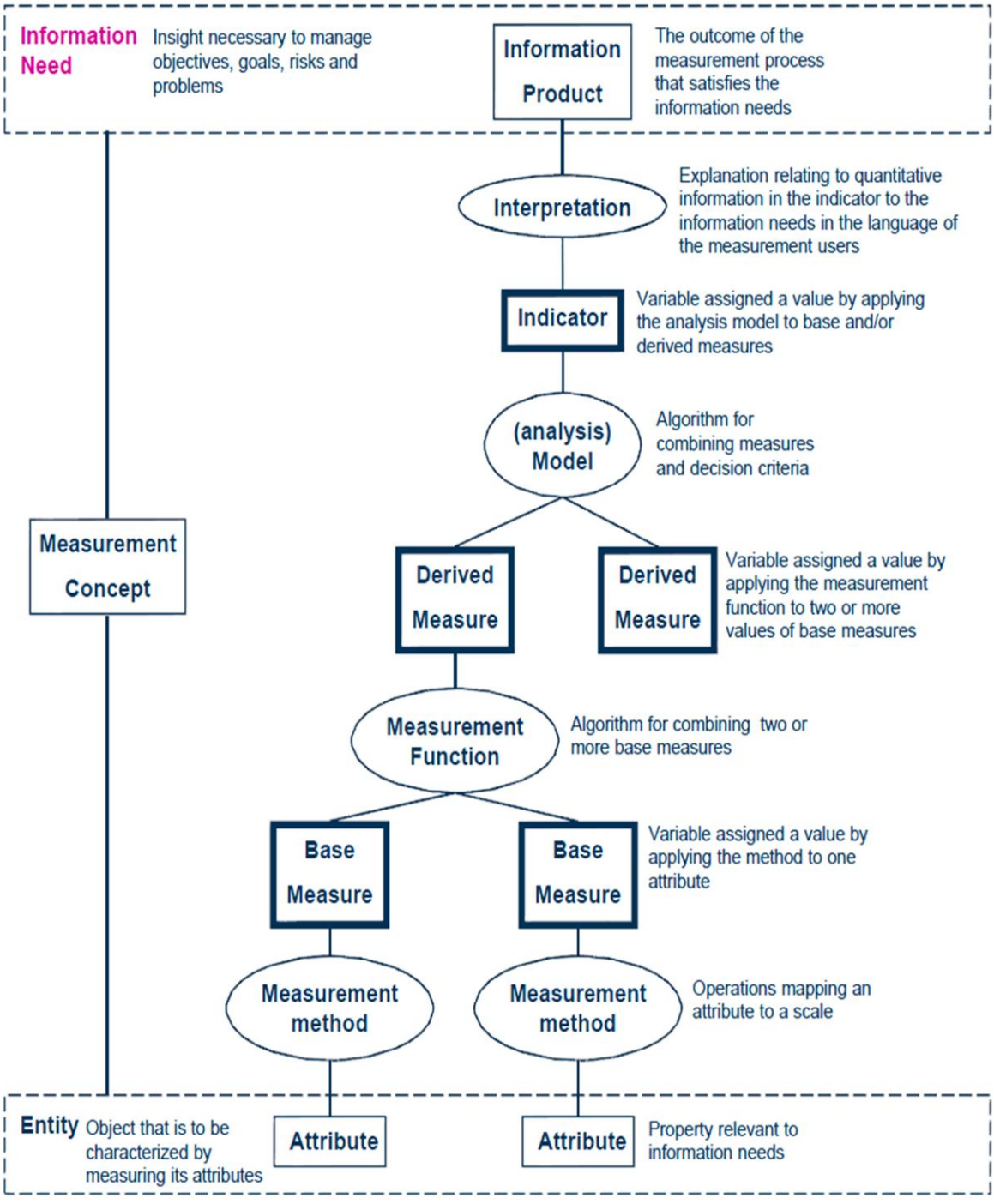


Figure 3-12. Creating an efficient measurement program [MED 09]

In addition, the measurement process defined in [ISO 07d] includes an evaluation activity [MCG 02, MED 09].

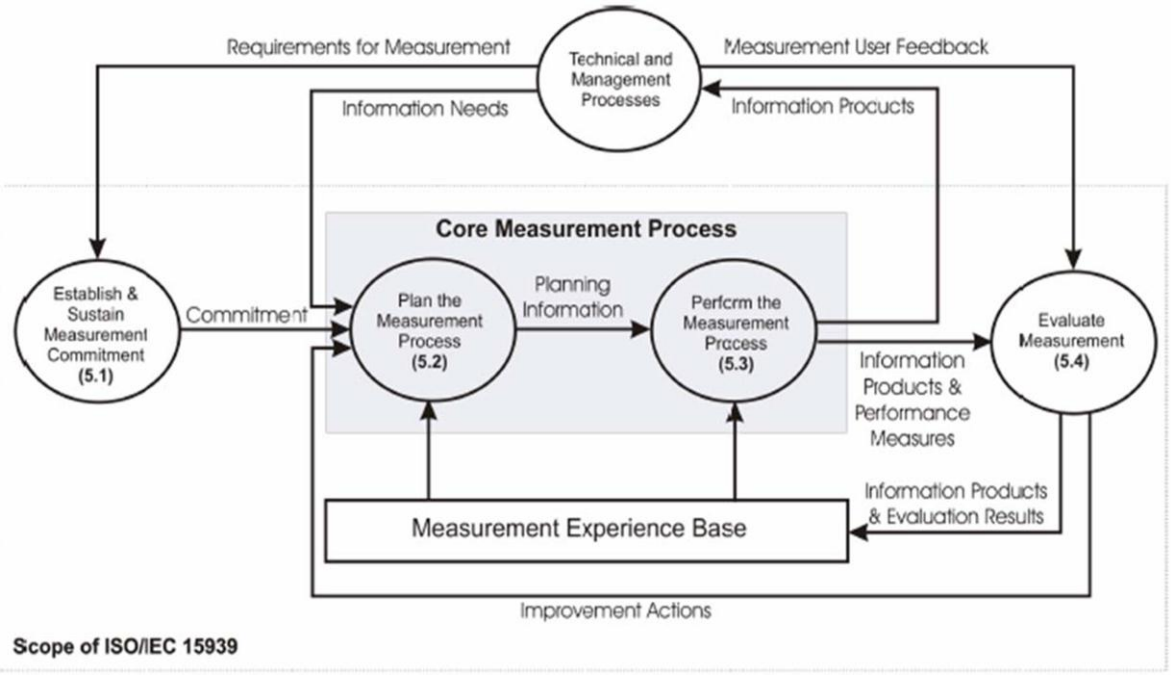


Figure 3-13. Measurement process procedure [MED 09]

The INCOSE Systems Engineering Measurement Primer [INC 98] points out that measurement needs to be viewed as a process for obtaining the vital insight into the progress, products, and/or processes of the project or system being developed. The process includes the activities for selecting and specifying the measures, establishing a measurement plan, planning and executing the data collection and storage, analyzing the data, reporting the results (see Figure 3-13), and most importantly, taking action [CAR 98, WA 99]. The intent is to emphasize that evaluation and feedback are an essential component of the measurement process, and should lead to improvements of the measurement process and measures. A measurement process is applicable to system and software engineering and management disciplines.

3.4.1.2 Measurements of software products

System and software measurement is a key discipline in evaluating the quality of products and the capability of organizational processes. It is becoming increasingly important in two-party business agreements, where it provides a basis for specification, management, and acceptance criteria [ISO 07d]. Measures for the early development phases would be the most useful ones, since the early phases and the artifacts they produce are believed to have the largest impact on the entire software development process and the final product [MOR 03]. As a first step, the product assessment will aim principally in assessing the software product quality characteristics as closely as possible [APR 04]. Available methods for product measurement can be used to evaluate compliance of a product to user needs. Examples of such methods are presented in [ISO 05, ISO 07b, ISO 07c]. The methods are based on the specification of product quality and evaluating the end product with respect to this

specification. Research [KAT 06] and standards [ISO 10, ISO 11b] divide requirement quality into quality attributes for single requirements and requirement sets. Then a requirement has a set of quality attributes such as unambiguous, atomic correct, complete, necessary, verifiable, feasible… [ISO 10, ISO 11b]. Based on this definition, quality of a requirement set shall have the following properties [GLI 07, ISO 10, ISO 11b]: hierarchical structure, completeness, organizational structure according to an appropriate grouping scheme, external consistency, no duplication of information within any level of the hierarchical structure, maintainability… the requirements must be also traceable [ISO 11, CES 10a]. To evaluate the fulfillment of a requirement, there are various techniques that attempt to assess their different quality characteristics, in this instance we have inspection, simulation and formal verification, amongst others [ISO 11]. They generally depend on a specific development phase, a specific requirement representation or are applicable for specific requirements only (e.g., Safety) [ISO 11].

During the collection of product quality requirements and the development of the product, architectural issues deserve special attention [VAN 08, VAN 09]. This is based on the notion that an architecture permits or precludes the achievement of a system's targeted quality attributes [ISO 11, CES 10b, CES 10c]. Producing *"good"* architecture is the first order approach to achieving product quality attributes. The importance of good architecture is, however, underlined [IEC 10, IEC 11, INC 07, HIS 12]. Design quality is calculated based on fulfillment of linked requirements.

Once the basis for requirement quality and fulfillment is defined, a framework is needed to allow monitoring and optimization of development projects.

3.4.2 *Assessment of product measurements through process measurements*

Nowadays, organizations are faced with strong competition and for this reason they have to continuously improve their processes. By monitoring the performance of the software development process, it is possible to provide an overview of actual results of that process, and to take corrective action based on these results [INC 98, CES 10a, FEN 97, CAR 98, WA 99]. The importance of using measurements in software engineering is further supported by the fact that several quality models and process improvement methodologies, for example SPICE [IEC 11, HIS 12] and CMMI [CMM 02, KNE 06], place emphasis on measurements [BER 06]. To realize an integrated view of the current quality status of a project and support project planning and monitoring across different levels of the whole v-cycle, integrating the process with the product perspective is necessary to define and execute these metrics against concrete development data. The previous step of the software processes improvement is their evaluation and this goal requires the definition of measures related to the different elements involved in software processes [GAR 06b]. Using this system related information, it is possible to assess requirements, their quality and fulfillment and to derive assessments of process specific progress metrics, milestone fulfillments, and risk indicators from them [GAR 06a, GAR 09]. Due to the subjective character of data quality, it is important to stress the difference between the concepts of measurement (*"measurement is the act of assigning a number to an attribute of an object being observed*") against assessment ("*the classification of someone or something with respect to its worth*") [CAB 07]. Some frameworks have been proposed in this sense [CAN 06, GAR 06b, STA 09, CES 10a]. Most of them rely on the definition of a metamodel based on the ontology [GAR 06a, CES 10a] of key concepts of a process measurement [INC 98, ISO 07d, MCG 02] (see Figure 3-14).

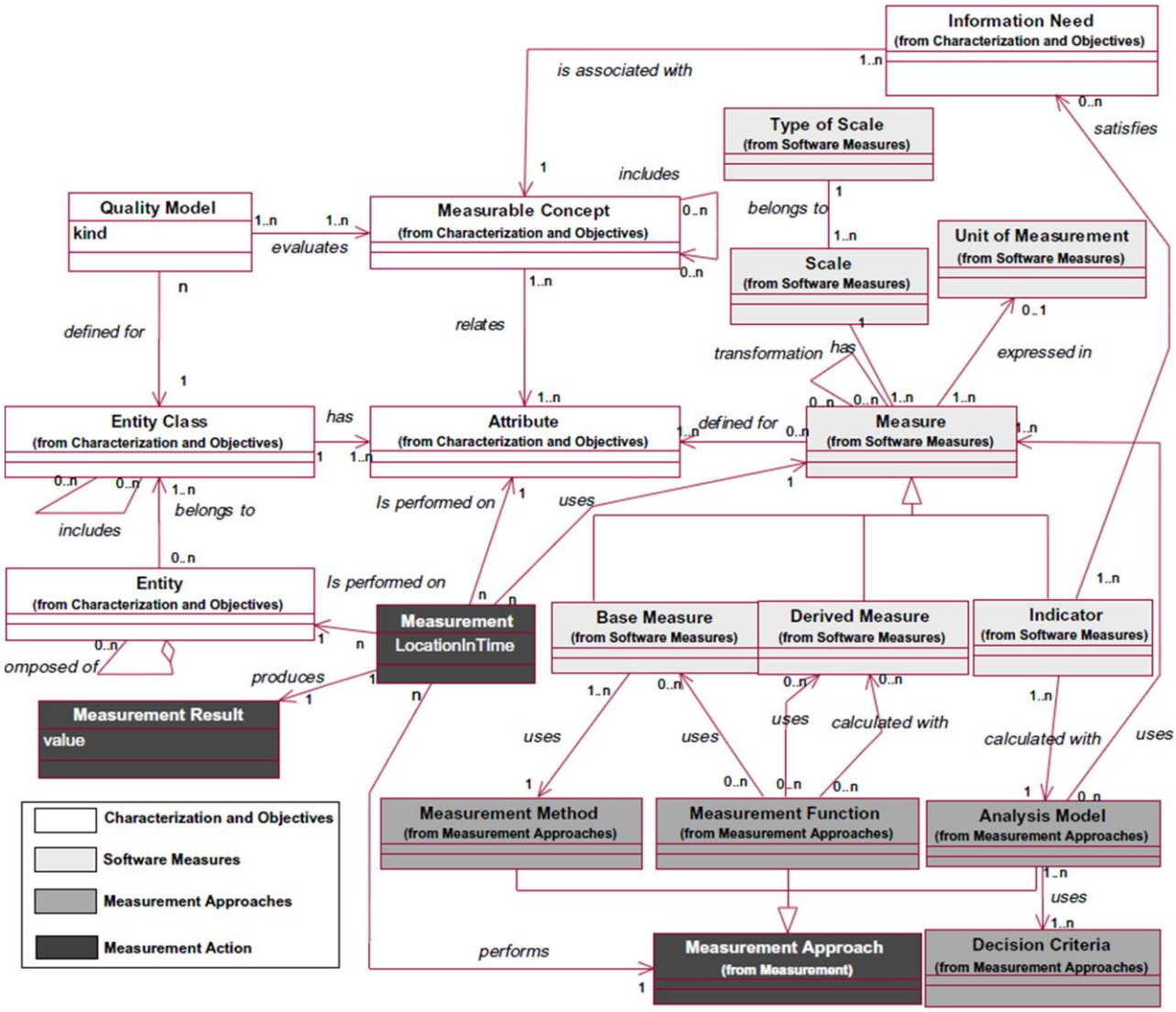


Figure 3-14. Software measurement ontology

In [CES 10a], it is proposed as an extension of SPEM with generic measurement concepts compared to the existing metric concept in SPEM (see Figure 3-15). By defining these extensions in SPEM packages, reusable measures can be defined, as well as companywide definitions of quality metrics and a measurement plan. The environment is implemented through the Permeter tool.

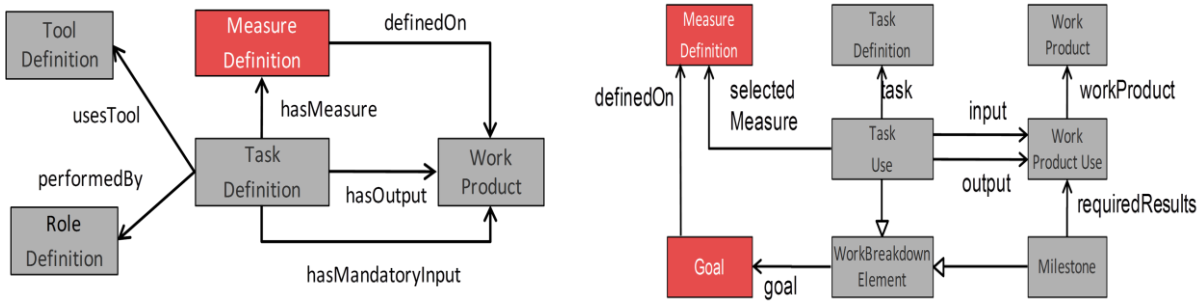


Figure 3-15. SPEM extension for Measure definition

3.5 Literature review summary

Modern day systems, specifically embedded systems are gaining in overall interactive complexity with increasing constraints (functional, real-time, safety, etc.), coupled with the pressures of tight schedules. These complexities are concerned by technologies that can help manage these concerns in the system engineering community. Using model-driven engineering is a way to do that.

3.5.1 *Research objectives review*

The relevant goal in this thesis is to take into account the requirements management in the context of model driven engineering with the objective of considering the needs identified in the certification standards and, more specifically, in the automotive domain with the ISO26262 standard and the HIS Automotive SPICE referential. In the introduction, we have defined some research questions regarding this objective, namely:

- What properties are necessary in a requirements engineering modeling framework?
- What are the requirements and attributes to formalize the development processes regarding their enactment, their measurement and the monitoring?
- What methodology is efficient enough to merge the certification approaches for both process development and the end-product?

Obviously, these questions must be all taken into account in the context of our subject with the consideration of automotive standards.
Regarding the first question, we have discussed languages such as SysML and DARWIN for requirement representation, and EAST-ADL2 for architecture description. It was noted however, that sometimes they fail to meet the ISO26262 and SPICE recommendations. For example, ISO26262 states that *“Safety requirements shall have the following attributes: a) unique identification remaining unchanged throughout the safety lifecycle; b) status; and c) ASIL"*. SysML does not comply with this statement. The traceability with architectural elements is missing in DARWIN while satisfactory and complete requirement engineering must also allocate requirements to design elements according to the standards. Although EAST-ADL2 meets this last requirement, many others attributes quality and characteristics about requirements [ISO 07b] summarized in ISO26262-8 chapter 6 ”Specification and management of safety requirements” are not taken into account, incomplete or poorly documented. Another weakness in these languages is their requirement classification which is not compliant with the hierarchical and organizational structure imposed by the ISO26262. They are generally based on the breakdown proposed by Glinz in [GLI 00]. The latter classify *“safety requirements*” like quality requirements while in the ISO26262, any requirement (even a functional one) can be considered as a ”safety requirement”. The following table 3-1 summarizes which standards expectations are not met by the languages.

Table 3-1. SPICE and ISO26262 standards criteria at product level not met by existing modeling languages in literature

Standards criteria	SysML	DARWIN	EAST-ADL2
C1. Requirement characteristics quality and attributes	✗	Based on ISO9126 [ISO 07b] ✓	N/A
C2. Requirement allocation on architectural elements	With allocation and block diagram ✓	N/A	Different architecture views ✓
C3. Requirements hierarchy and structure	✗	✗	✗

In response to the second question, SPEM was established as the best solution for process modeling. It aims to improve not only the *product*, but also the *process* that leads to the product, making high-quality software development more affordable [SCH 02]. ISO26262 in the way that it is intended can be used as a tool to assess the product quality, and in the same way, SPICE could be used to measure the process quality. Product quality can be done through the use of standards involving product quality terminology and specifying product quality in measurable terms. It must also take into account the quality rules of ISO26262, the determination of the ASIL for example, which is not the case in SPEM. Process quality can be done through modeling and assessment of established software development processes. Different SPICE engineering steps for system activities which include requirements elicitation (ENG1), system requirements analysis (ENG2) and system architecture design (ENG3) must be documented, as specific safety activities. But once again, the classification or hierarchy of methods, tools and properties according to severity levels as defined in ISO26262 cannot be combined as there is no equivalent concept in the language.

Both the software development process and the software product can be measured. Increased control and early feedback from results are the outcomes of measurement. Application of measurement is therefore a prerequisite and an excellent tool to guide process improvement because it provides feedback on the effects of process actions on product quality. SPEM does not integrate the endeavor layer where this measurement and control is possible when a process is running. Permeter tool, through its extension for EPF (Eclipse Process Framework) tries to fill this gap. A summary of SPEM weaknesses is provided in the table 3-2 below.

Table 3-2. SPICE and ISO26262 standards criteria at process level not met by SPEM language

Standards criteria	SPEM
C1. Process modeling	Classification or hierarchy of methods, tools and properties according to severity levels does not withstand by any concept ✗
C2. Process measurement	Quality rules of ISO26262, for example the determination of the ASIL is not taken into account in SPEM ✗
C3. Specific process configuration	From a generic process, it is not possible to generate automatically a specific process for a project ✗
C4. Process monitoring	Process control is not possible as long as SPEM does not provide a running layer ✗

The underlying issue behind the third question is the interfacing between process and product modeling languages together with other techniques currently used during the development cycle like Matlab Simulink models, for example. In the state of the art, that case has no solution. Papyrus MDT is useful for system modeling; EPF with the Permeter extensions is able to manage process modeling and assessment in a limited manner. Neither tool can provide the features proposed by the other. Furthermore, they are unable to communicate with each other. However, since benefits of model-based engineering come also from the interaction of the process and product models and their realization in a CASE tool, the close integration of the process and product properties must be managed.

3.5.2 *Research approach*

This thesis intends to provide a consistent solution which allows the combination of modeling and requirements management at models level and the consideration of the automotive standards within a model-driven approach. It is organized around the followings topics:

- The identification in the ISO26262 and automotive SPICE Standards of different needs to be covered and their means of production in a single manner. It results in a definition of a merging approach which embodies a product quality and process quality approaches;
- The consideration through and in models of requirements identified together with safety assets through advanced techniques and notations supporting the full requirements life cycle, from their definition up to their architectural allocation. Advanced techniques include mechanisms to trace requirements, such as traceability between requirements, traceability to model elements and to a system architecture which will be defined, traceability from origin specifications documents, etc.;
- The definition of a comprehensive methodology to guide the design of systems in such an environment. The approach uses process modeling and measurement to improve the control and the monitoring of a project and to reduce the cost and frequency of re-planning.

The integration of these mechanisms in a tooled platform based on the consideration of multi-formalisms modeling will be considered in the solutions.

3.6 Conclusion

In this section, we have presented existing works in our context for development of safety critical embedded systems. We have discussed the problems inherent in these works and we have confronted them with our research questions defined in the previous section. This led us to propose a new model driven approach for developing embedded systems that takes into account automotive standards and constraints that are not yet resolved in the state of practice. This reasoning about the gaps in the literature which led to these different research questions and this proposed research approach to resolve them was validated through some publications [4, 5]. The next section will present these contributions in detail.

4

Model-Driven Requirements Engineering process according to automotive standards

4 Model-driven requirements engineering process according to automotive standards

4.1 Introduction

As exposed in the introduction section, the aim of the thesis is to use the requirements in a modeling framework to define systems safe and compliant with automotive standards, following a hierarchical modeling process specifically covering the system engineering activities. The demands set in the previous section indicate three major expansions of current state of practices for reach this goal:

- Merging of ISO26262 and SPICE standards following a unique approach
- Specification of main requirements for engineering activities in a model-based environment considering safety aspects
- Modeling and measuring of product and process depending on the product quality specification

The following chapters detail the contributions brought regarding each of these expansions.

4.2 A merging of product oriented approach and process oriented approach

This chapter presents a framework where a SPICE assessment and a functional safety audit are simultaneously performed in a certification perspective (see Figure 4-1).

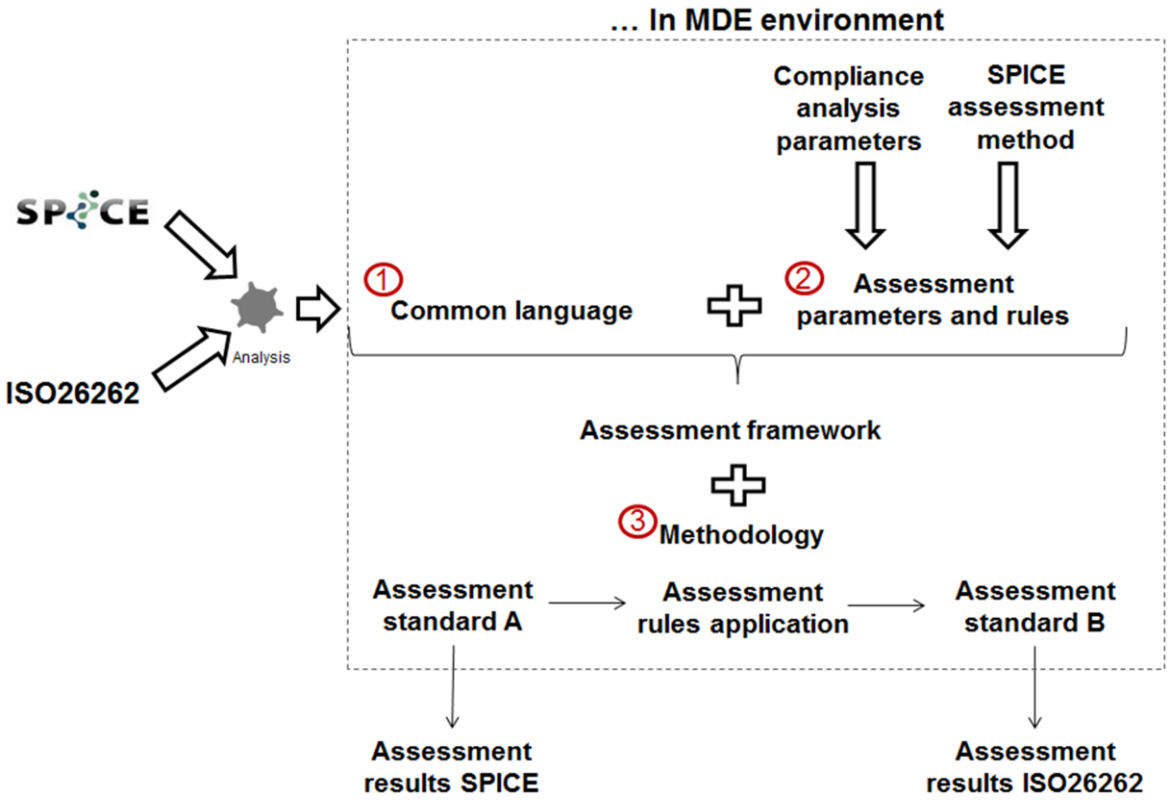


Figure 4-11. Compliance need between engineering process compliant to SPICE and ISO26262 standard

Firstly, we present an analysis of the overlapping between the two standards and of the identified gaps. Then, the considered methodology envisaged to unify them in a single process which allows a full assessment for compliance with both SPICE and ISO26262 standards is presented. An assessment framework to measure the process capability of a specific engineering organization that develops safety systems is discussed. Lastly, an example of use of the global certification framework is given.

4.2.1 Process based quality assurance and process based safety assurance

For a long time now there have been demands for process-oriented developments in the automotive sector. Many companies have already set themselves up here, or aligned their improvement projects accordingly. Indeed, in the current situation, suppliers have to prove process capabilities to the OEM (Original Equipment Manufacturers) through maturity models, and standards such CMMI (Capability Maturity Model Integration) or SPICE etc… These processes are described as being *"process-based*", in that they define a set of practices to be adhered to during the development of software. They provide good strategies to assess an organization's software development capability and, based on the resulting assessment, they allow identification of the process strengths, weaknesses and risks of preventing them. Unfortunately, because the latter do not comprise safety aspects, they do not satisfy the requirements for consistent safety management.

The new ISO26262 safety-related standard was released for the automotive industry with more stringent requirements on the development of a product, specifically to handle this purpose. It proposes a certification system that focuses on an end-product quality approach based on the construction of well-structured and reasoned safety arguments.

Regarding that, the current problem for suppliers, with the aim of mastering their processes, is to check whether they have to completely adapt their current projects now; or to check what has already been achieved according to Automotive SPICE, what has effectively been used and integrated to meet the new ISO26262 requirements as well (see Figure 4-2).

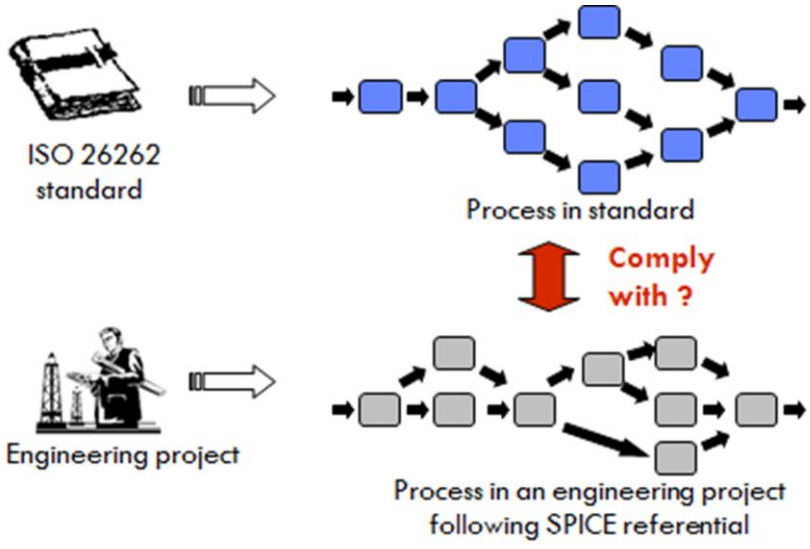


Figure 4-12. Compliance need between engineering process compliant to SPICE and ISO26262 standard

Several researches have already addressed this issue of how to incorporate the ISO26262 in the current practices. Indeed, many researches are looking for a mapping between SPICE and ISO26262 [HOE 08, LAM 11a, LAM 11b]. General evidence is that there is a high coverage of the SPICE scope by the ISO26262 standard, but a low coverage of the safety standard by SPICE instead. This is in particular because, in addition to the requirements defined at process level as it is the case in SPICE, the ISO26262 standard also includes specific requirements to be considered at product level. According to our study, we reached the same result as in [PET 09, PET 10]. For instance, we note that for SPICE (HIS scope for our concern), all processes are fully supported by ISO26262, failing processes support SUP8 and SUP9 (respectively configuration and problem resolution management, see Figure 4-3) that are only partially considered. Conversely, the processes of ISO26262 are only partially covered by the SPICE or not at all for many central activities (see Figure 4-4).

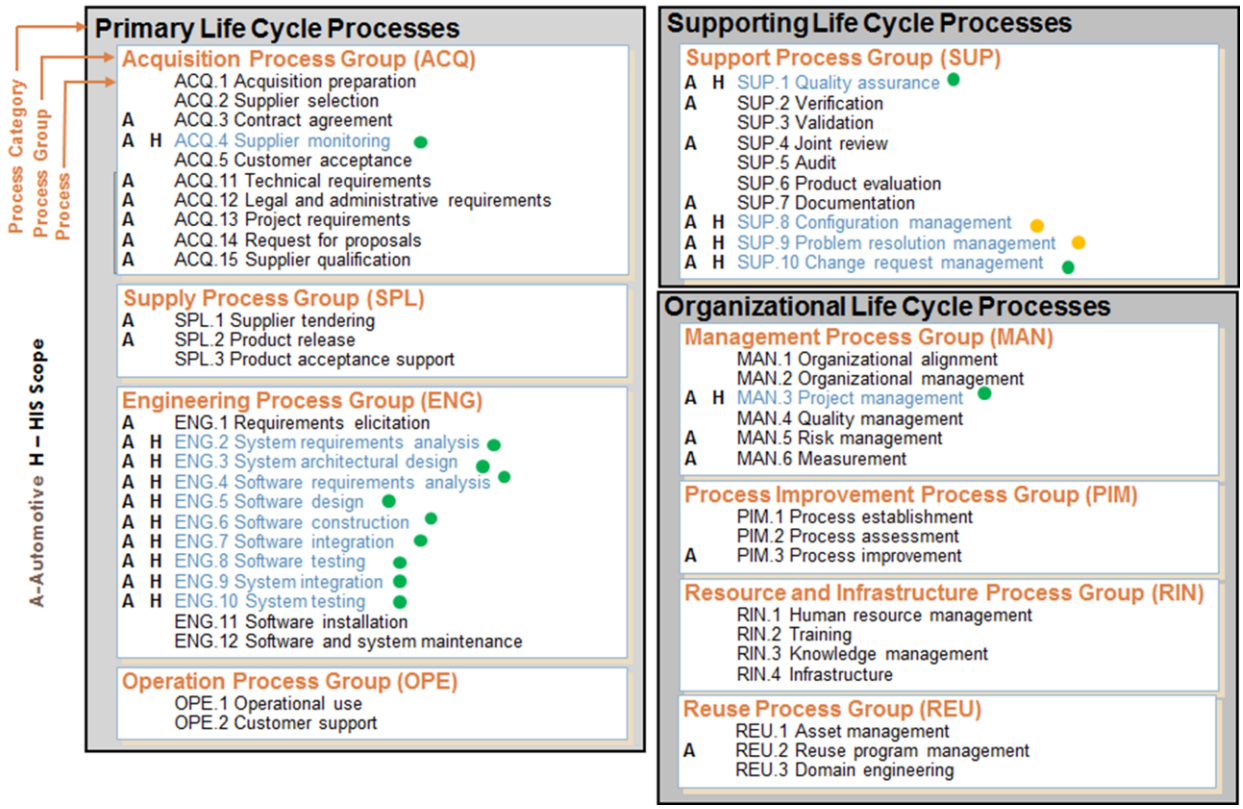


Figure 4-13. Supporting HIS automotive SPICE processes by ISO26262 standard

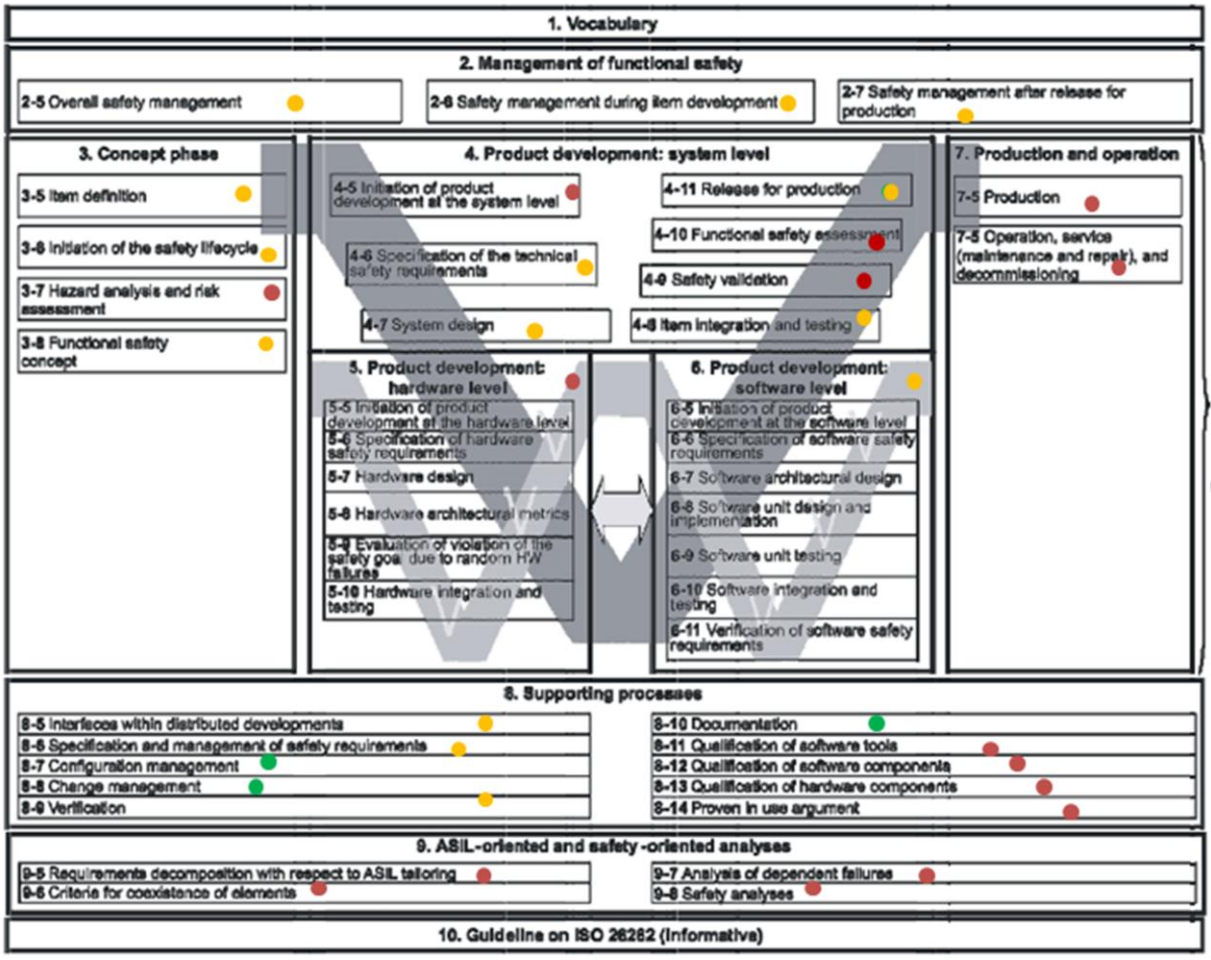


Figure 4-14. Supporting ISO26262 processes by HIS automotive SPICE

This is particularly true for the safety management, hazard analysis and risk assessment, safety concept definition, safety validation, production and operation, safety qualification, safety analyses processes. In Figure 4-5 below, we present this analysis following the logical sequence of development activities induced by the standard.

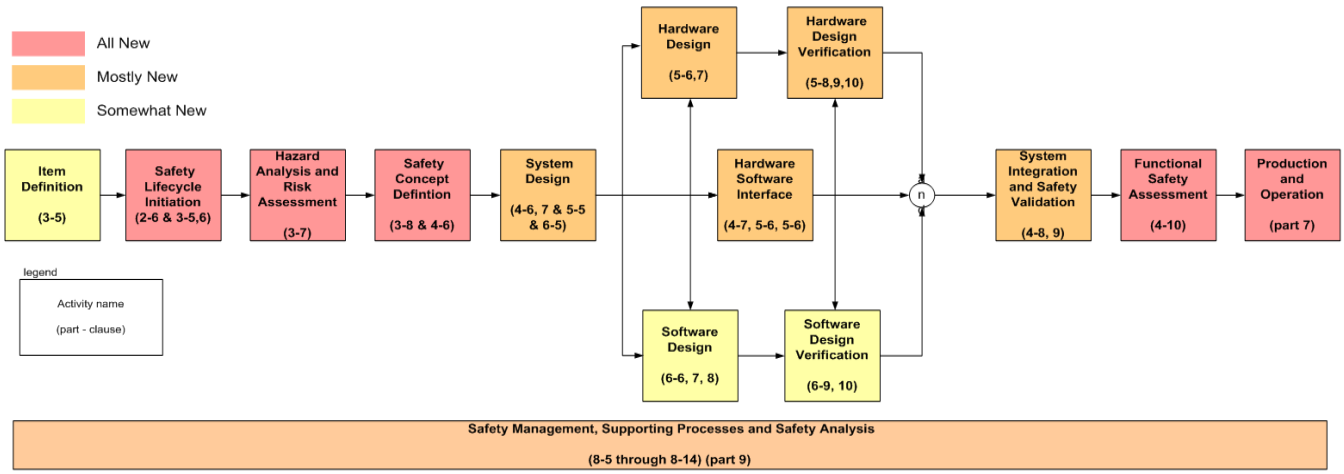


Figure 4-15. SPICE support in ISO26262

Some solutions are already emerging to fill this gap. But few have worked on how to assess the completeness during an audit to both standards together. Next, our approach presents how to enable the assessment of both standards through a unique framework, taking into account this finding.

Our work develops an assessment instrument for a specific automotive engineering organization that develops safety systems and that is aimed at supporting software certification by both end-product quality approach and development process approach.

4.2.2 Common product based and process based metamodel

In [ISO 11], an interesting note states that *"an organization's process definitions must address multiple standards at the same time. If a SPICE assessment is performed, then this SPICE assessment and a functional safety audit can be simultaneously performed. There is sufficient commonality in content that can help to avoid duplication of work or process between both standards and to allow synchronization of the planning*". For having these coordinated processes, we want to provide specific process cross references to ISO26262 requirements and SPICE. Among the research that has investigated the comparison between SPICE and ISO26262, some of it has opted to extend the SPICE standard to ensure compliance [PET 10, LAM 11]. Specifically, these approaches update SPICE processes according to some ISO26262 processes that are already partially covered. In addition, they add, at the appropriate level, processes purely dedicated to safety as the identification of hazards, the safety case creation, the classification of safety requirements and so on.

We take another position that we believe more appropriate in a certification context. Indeed, how can we ensure good compliance to a standard that has been modified if the modification or the extension has not been approved by the certification body that published the original standard? Thus, in our compliance study, we have chosen rather not to modify either of the two standards, but to allow nevertheless a combined assessment method which corresponds to a full compliance for these two automotive standards (see Figure 4-6). The enhanced model obtained is an integrated model which focuses on certification and assessment of software, based on both product quality and process development approaches in a wider scope of requirements.

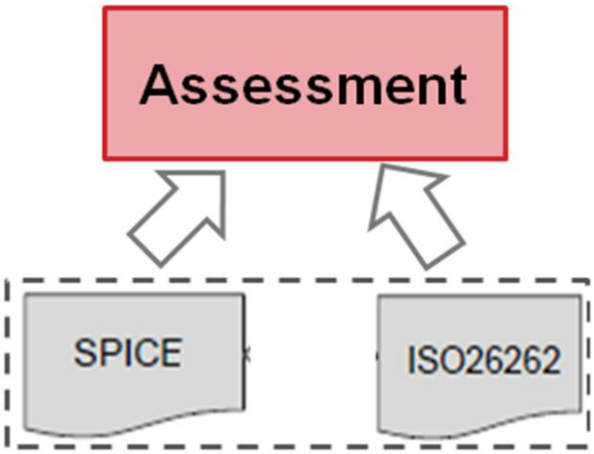


Figure 4-16. Create a standard process from SPICE and ISO26262 standards

4.2.2.1 ISO26262 domain model and SPICE domain model

The methodology used for that purpose is inspired by the model and metamodel composition paradigms (also called model combination or weaving) from the model-driven engineering community [BAR 07]. It is a structured approach that relies on the ontologies. Thus, the respective semantics of each standard are considered as isomorph graphs which need to be matched to provide a single one that embodies all the different concepts [KOL 09].

First of all, we defined a structured set of terms and concepts for each standard as well as semantic relations as ontology. The ontology was enriched by adding to each of the different concepts, some attributes in order to form a domain model. For using this analysis and modeling of the standard, we defined a metamodel: all elements of the ontology, together with their attributes have been used.

In ISO26262 (see Figure 4-7), There are many *Safety lifecycles* which are composed of *Parts*. A Part is composed of *Clauses*. For each *Clause* is defined an *Objective*, a *General* purpose, some *Workproducts* as *Input* (external to the projects, mandatory or optional) and some others as *Output*. Output Workproducts are results of some *Requirements and Recommendations*: individual *Requirement* or a set of requirements as a *Requirement group*. A Requirement (or Requirement group) can be applicable only for specific *ASIL*. *Examples* and *Notes* sometimes give additional information to better understand the requirements. A requirement can also have a reference in form of *Annex* or *Table* of *Property* or *Method*. The Table's elements must be used alternatively or consecutively and are subject to different *recommendation levels* according to an *ASIL*. They are explained further through *Note*s when it is necessary.

For SPICE (see Figure 4-8), the standard is composed of *Process Category* which includes *Process Groups* where we find individual *Process*. A *Process* has a *Purpose*, some *Outcomes*, *Base Practices* and *Workproducts*. *Base Practice* and *Workproduct* meet one or several *Outcomes*. For the process evaluation, *Base Practices* are used to determine *Capability Level* 1. For other *Capability Levels*, the evaluation is based on the *Process Attributes*. *Process Attribute* includes capability indicators such as *Generic Practice* and *Generic Resources*. Resources are either *Tool, Infrastructure*, *Method* or *Human Resources*. *Notes* for additional information can be attached to *Outcome, Base Practice* and *Process Attribute*.

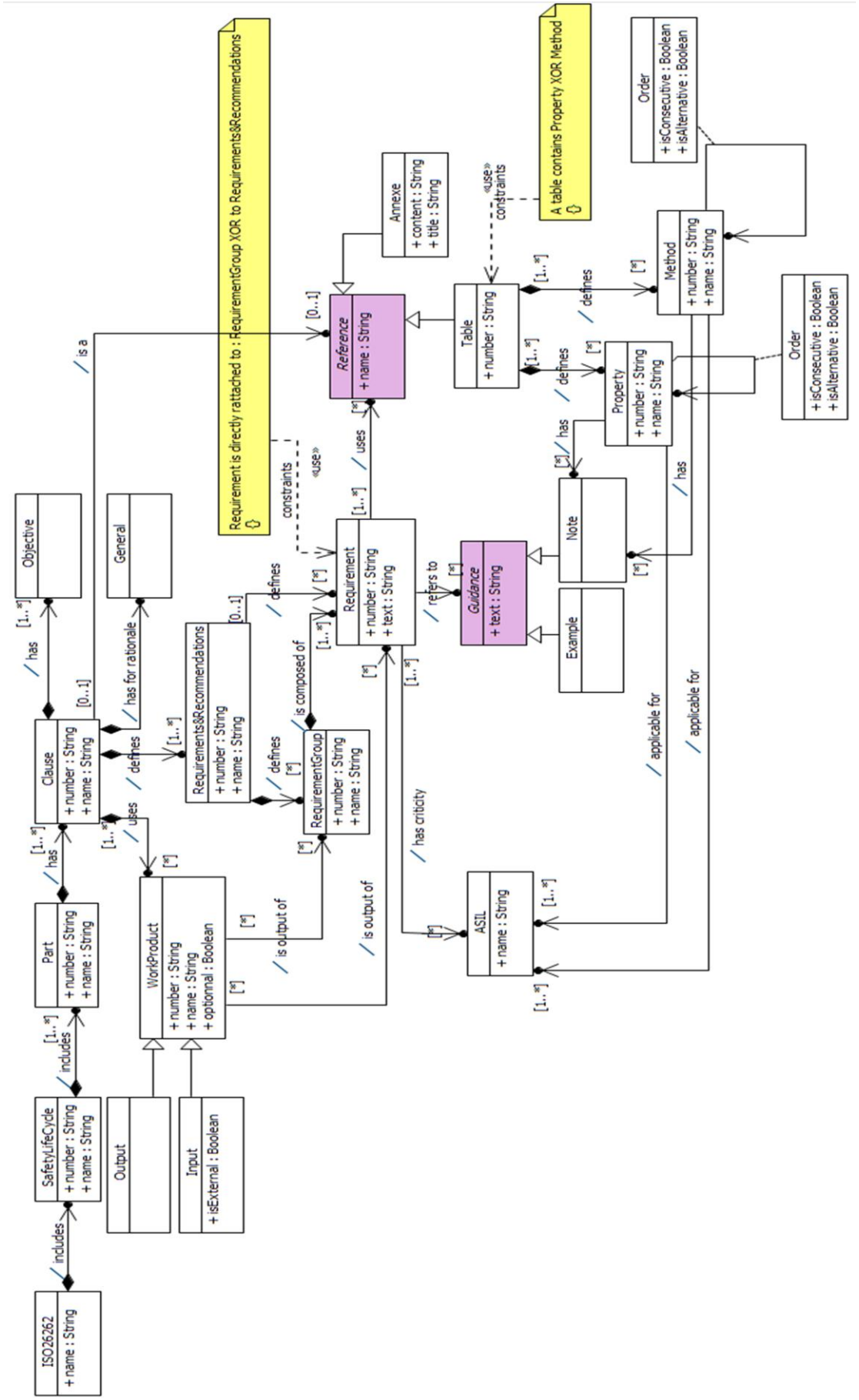


Figure 4-17. ISO26262 domain model

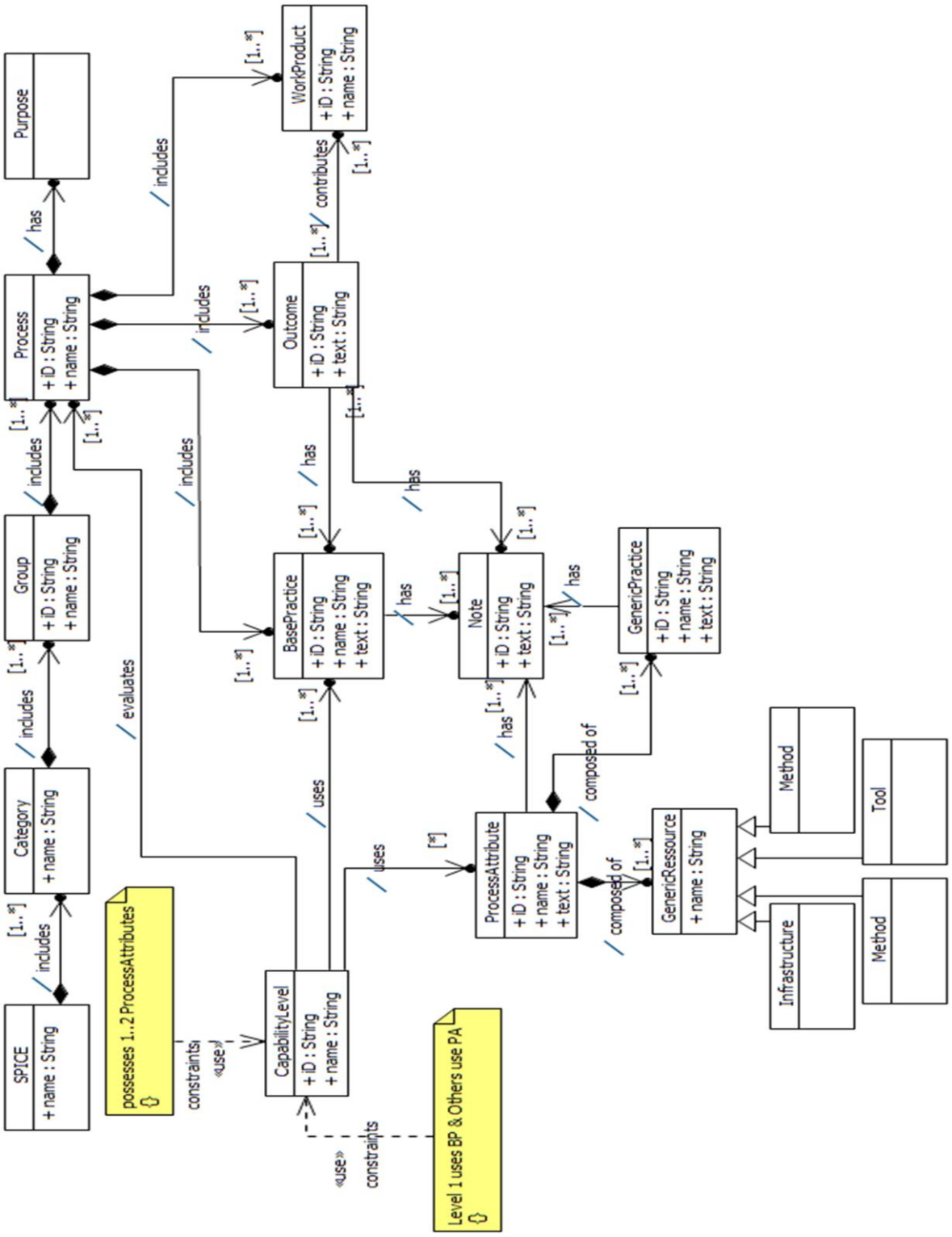


Figure 4-18. SPICE domain model

4.2.2.2 Algorithm of Metamodel Composition and Extension

For the composition of the two standards, we used a generic framework inspired from [FRA 07a]. The composition model proposed follows two steps (see Figure 4-9):

(1) The matching step that identifies the model elements (nodes or edges) that describe the same concepts in different models and that have to be composed;

(2) The merging step where the matched model elements are merged (to form a single class) to create new model elements that represent an integrated view of the concepts in the composed model.

The equivalence between two concepts is based on the operation *€* from [BAR 07]. The operation € specifies that, if a node from a model M_i is equivalent (=) with a node from a model M_j, a new node similar to both will be created in a model result M_r. If a node from M_j is not equivalent with any node from M_i and vice versa, a new node will be created in M_r. The equivalence (=) defined like that is imprecise because it can have different equivalence degrees. We replace it with a notion of similarity measure, continuous $\in$ [0, 1] (chosen by an expert) proposed by [CHI 11] and also applicable to edge.

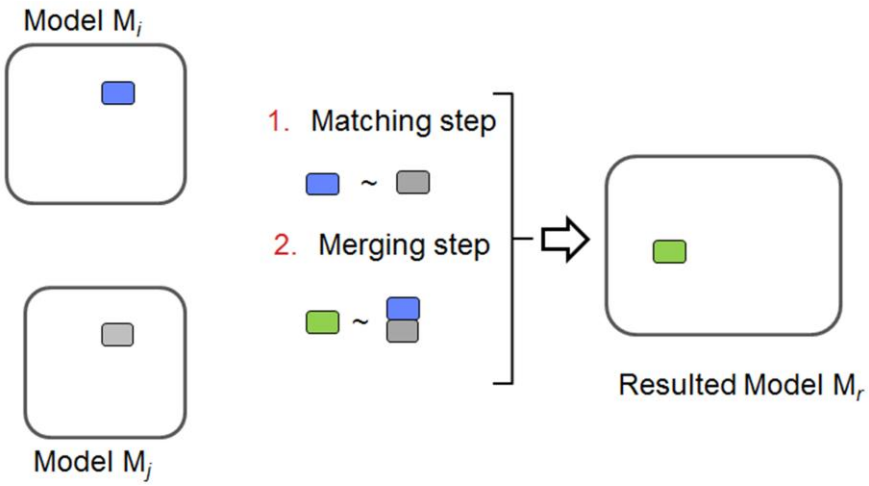


Figure 4-19. Create a standard process from SPICE and ISO26262 standards

Furthermore, global semantics are more important than semantics of individual nodes. The composition method is then improved by the application of the pattern matching principle proposed also by [CHI 11]: let's consider several nodes from two models that are similar. If they form a connected graph in M_j, the edges connecting the nodes in M_f have to be similar with the edges connecting the nodes in M_i (see Figure 4-10).

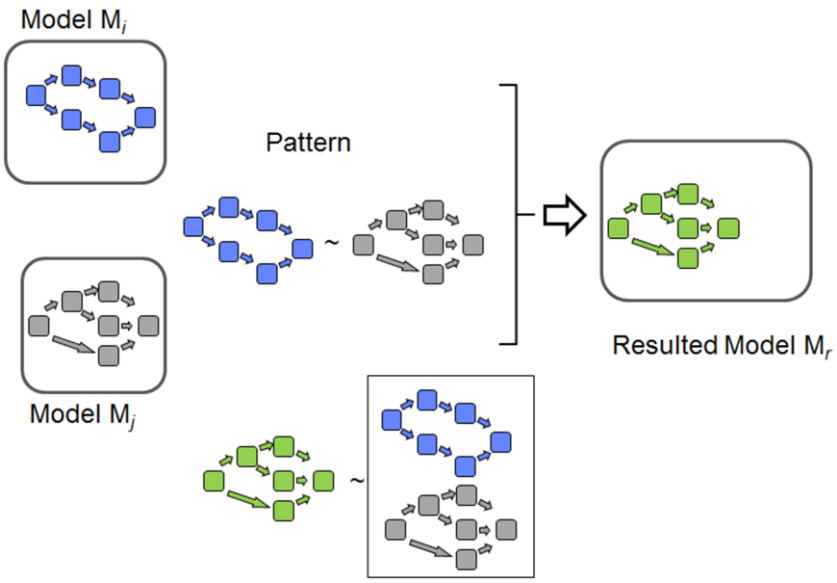


Figure 4-20. Create a standard process from SPICE and ISO26262 standards

4.2.2.3 Extended metamodel for SPICE and IS026262 processes

We apply the full resulting algorithm for identifying the extended metamodel corresponding to the ontology of both SPICE (for which we listed 16 concepts) and ISO26262 (for which we listed 21 concepts) standards. The common resulting metamodel is composed of 27 concepts in total (see Figure 4-13).

Let's us describe the algorithm application from the extract examples of the standards in Figure 4-11 and Figure 4-12.

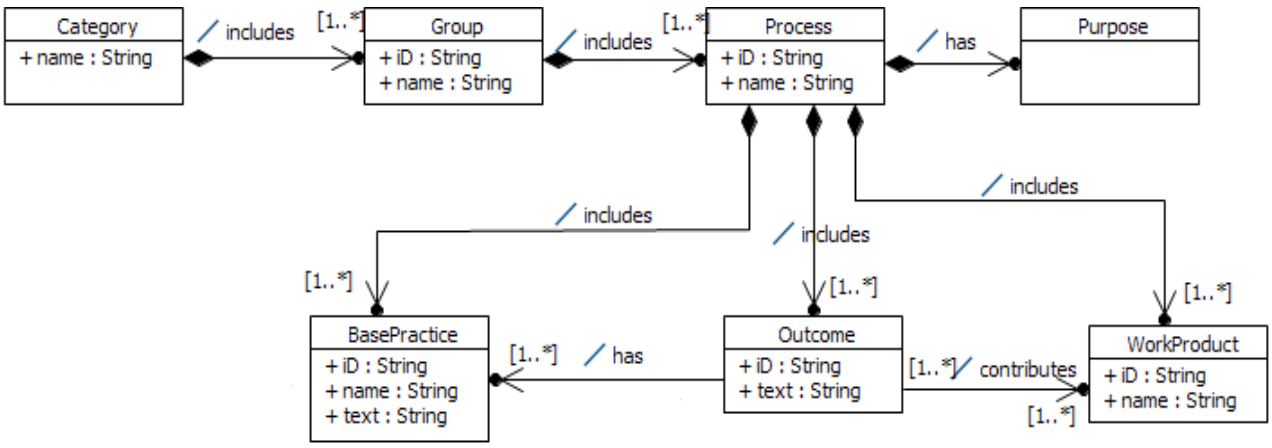


Figure 4-21. Extract of the SPICE domain model

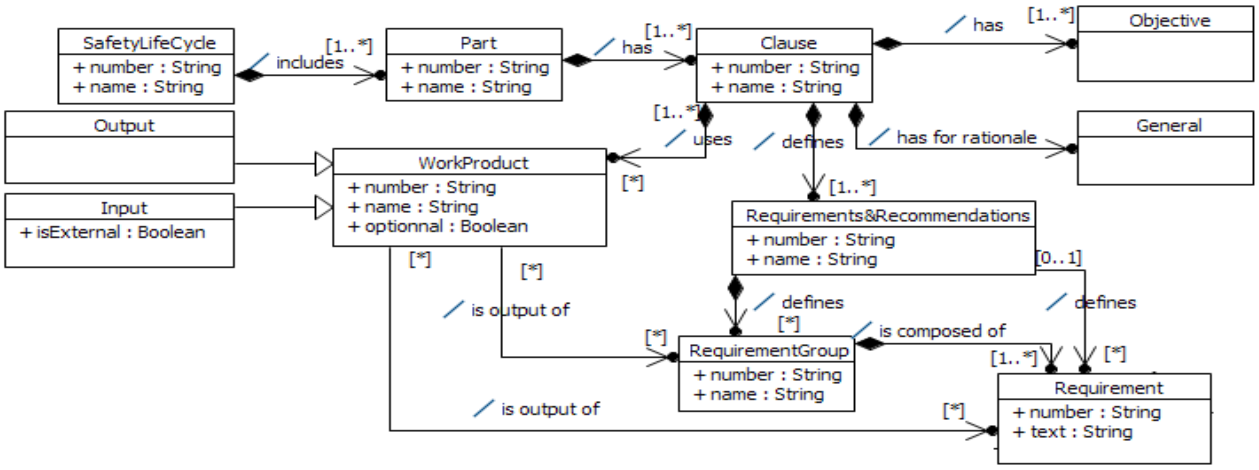


Figure 4-22. Extract of ISO26262 domain model

We start by determining the similarity of *Category, Group, Process* and *Purpose* concepts in SPICE that can be considered similar respectively with *Safety Lifecycle, Part, Clause* and *Objective* concepts in ISO26262, based on our understanding of their semantic definitions. We apply the pattern matching principle: as they form a connected graph in ISO26262, the edges connecting them are similar to the edges connecting the concepts in SPICE.

With regard to their semantics, we establish that *Outcome* and *Requirement* concepts are similar. Because *RequirementGroup* concept has an inheritance hierarchy with *Requirement* concept, we then stated that *Outcome* concept is also similar to *RequirementGroup*. In the same way, a degree of similarity is found between *Outcome* and *Requirements& Recommendations*. Outcome is then similar to several nodes.

Outcome concept is connected with *Workproduct* and *Process* concepts through edges. By transitivity of similarity of edges, we analyze that *Clause* concept is similar to *Process* concept whereas *Workproduct* is a similar concept in both standards.

Output, Input, BasePractice and *General* concepts do not have matching concepts… whereas the *ASIL* concept which is only present in ISO26262: they are copied in the common metamodel as such (see Table 4-1).

Table 4-1. Example of some matching concepts

SPICE concept	ISO26262 concept	Final Concept
Category	Safety Lifecycle	Category
Workproduct	Workproduct	Workproduct
Outcome	Requirement	Requirement
N/A	ASIL	ASIL
Process	Clause	Clause
N/A	Input	Input
Base Practice	N/A	Base Practice

The resulting metamodel that we obtain allows to describe the two standards in a single way (see Figure 4-13). It also provides an assessment framework able to measure both process capability and product quality for the safety systems development that will be in the following.

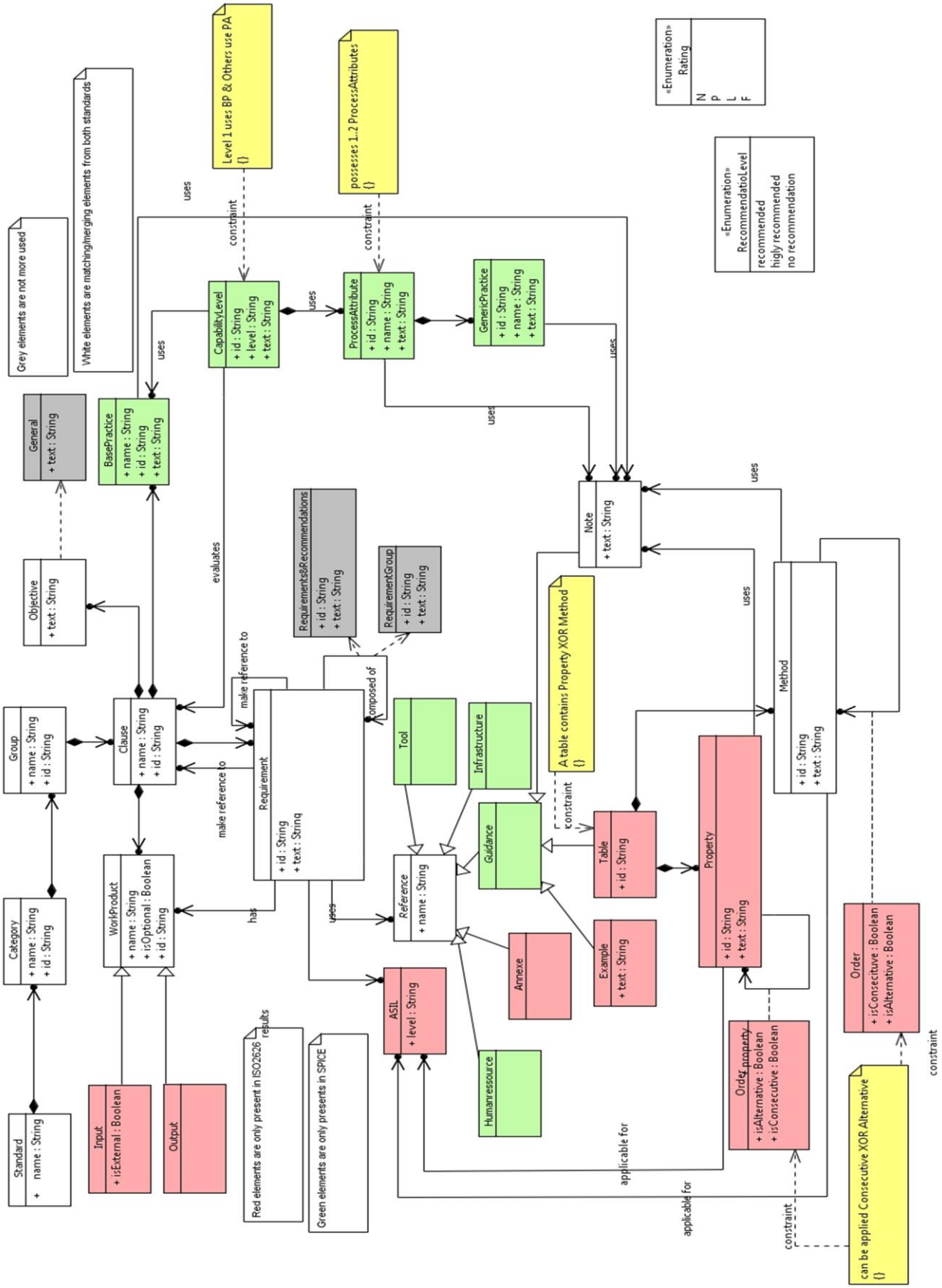


Figure 4-23. Common metamodel from ISO26262 and SPICE. The concepts only present in ISO26262 are in red. Respectively, the concepts only present in SPICE are in green. White ones are those common.

4.2.3 Assessment framework

To implement our metamodel, we decided to use an Excel™ framework. As in [MES 09], we chose this tool because, first of all, it is widely used in the automotive industry as it is quite simple to manipulate; furthermore, it served as an experimental tool for interpreting the suitability of our metamodel before we translated it in a more formal way.

4.2.3.1 Boundaries of the context evaluation

Before beginning this audit, it is necessary to specify the boundaries of the assessment. Indeed, the users have an opportunity to select their interesting quality factors to be applied in the certification and assessment exercise depending on the organization's requirements:
- What is the system (subsystem) under evaluation?
- Given that each SPICE process may be audited individually, and that it can achieve different maturity levels, it is necessary to identify the specific processes that will be subject to evaluation.
- Also for ISO26262, given that the number of requirements to be covered increases according to the higher severity to be achieved (around 1300 requirements for *ASIL A* and more than 1450 requirements for *ASIL D* for instance, see Figure 4-14), it is necessary to define the ASIL referred to as the system (or subsystem). Moreover, the processes (parts) integrated as part of the assessment will be specified later if the entire standard has to be uncovered. For instance, if it is the *ASIL A* that is referred to as the system, all requirements that are specifically valid for the other severities (*B, C, or D*) are hidden by a filter defined in the framework.

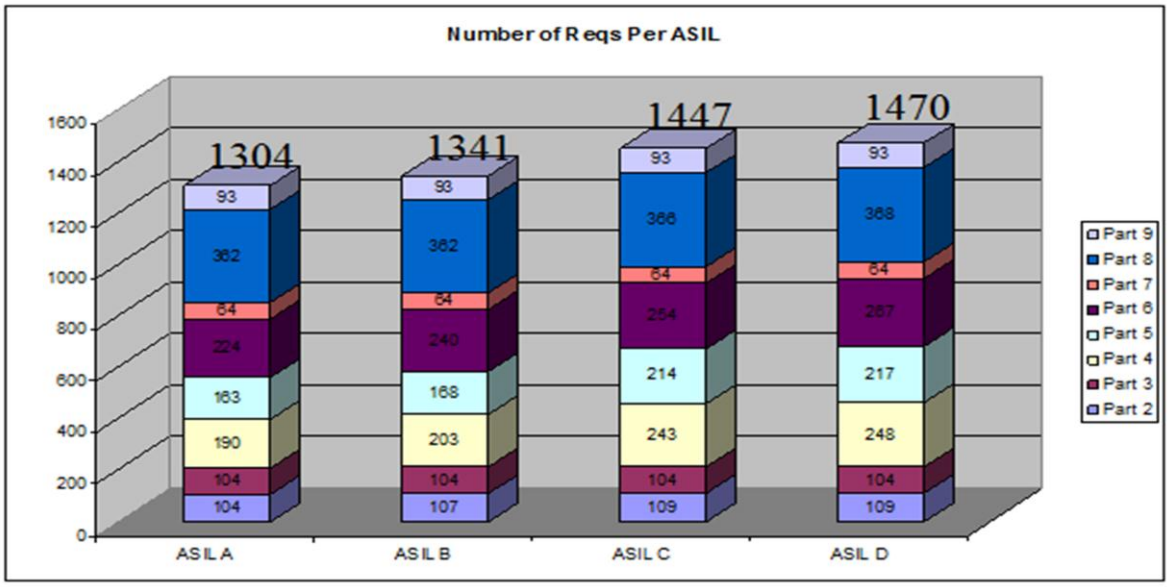


Figure 4-24. Intern study realized by DELPHI on number of Requirements as per ASIL in ISO26262 (DIS)

It is the same for the methods and properties tables recommended by the standard that can also be filtered as, depending on the ASIL classification, the recommendation to apply a certain method or property in the development process differs (see Table 4-2).

Table 4-2. System Design Verification methods table from ISO26262

Methods	ASIL			
	A	B	C	D
System design inspection	+	++	++	++
System design walkthrough	++	+	o	o
System design simulation	+	+	++	++
System design prototyping and vehicle tests	+	+	++	++
Deductive analysis	o	o	+	++
Inductive analysis	++	++	++	++

For generalization, all the SPICE requirements have their ASIL set to *"All"*, i.e. they are to be considered for all ASIL levels.

- Since the certification includes three pillars (product, process and people), we also consider a fourth one: the human resources dimension. We have identified a role responsible for each Workproduct. If desired, the assessment can also be filtered by role.

These different settings can be parameterized in our framework. After having fixed the quality factors, we can know exactly how many requirements must be met in total for the two standards, and also the number necessary to be covered for each standard. An overview of different activities to perform for set the context boundaries is summarized in Figure 4-15 below.

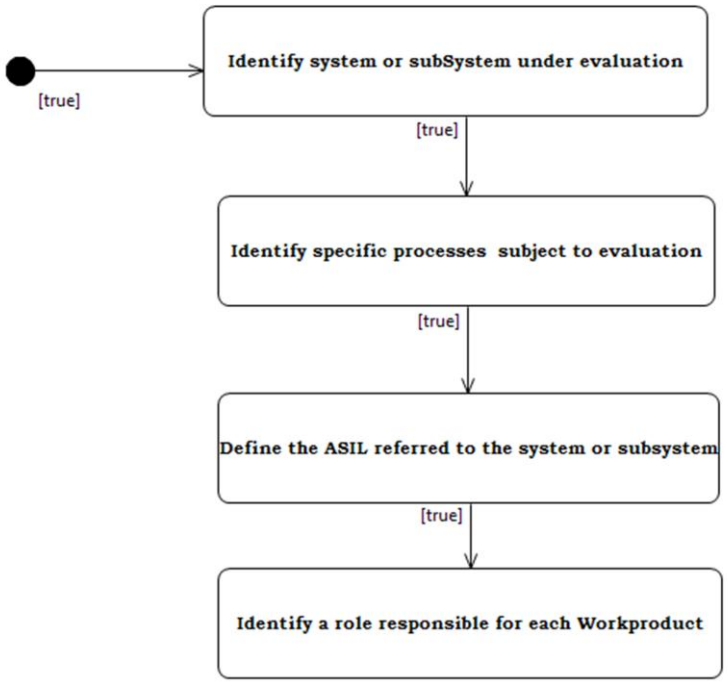


Figure 4-25. Activities to perform to fix the context boundaries for an assessment

4.2.3.2 Assessment Control

We may consider two different cases for the assessment. The first one is to identify which requirements of the safety standard ISO26262 already have a good support if we suppose a SPICE compliance process which is ready. The second one is to identify which requirements of the safety

standard ISO26262 will not be fulfilled assuming the same prerequisite. We first present our analysis of the overlap and the gaps existing between the SPICE and ISO26262 standards; this analysis allows to identify how SPICE requirements are covered in the safety standard. Regarding that, for each requirement, some parameters have been added. This results in the addition of new columns in the framework used for implementation, i.e. an extension of the common metamodel's core (see Figure 4-16).

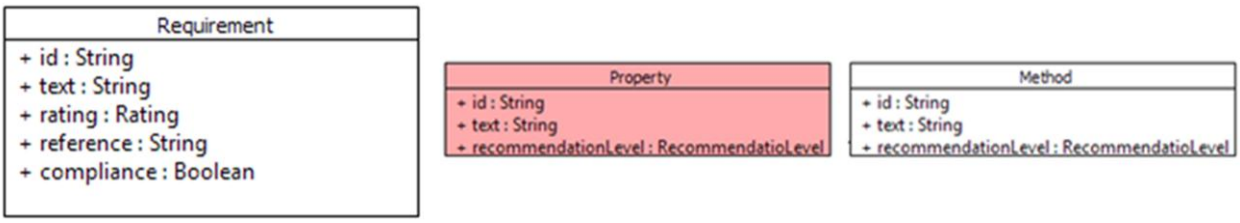


Figure 4-26: Metamodel modifications to handle the boundaries concepts

Let's consider the example of a requirement *ReqA* of the standard A (see Figure 4-17):
- The *Compliance* column indicates the level of coverage of *ReqA* in the standard B. Three values are possible: "*OK*" (*ReqA* is completely covered);"*Partially*" (*ReqA* is partially covered),"*NOK*" (*ReqA* is not covered at all).
- The R*eferences* column indicates the clause(s) reference(s) of the requirement(s) corresponding in standard B.
- The *Rating* column to assign a rating value to the requirement.
- *Recommendation level* is an attribute attached to Table and Property concepts following the ASIL value. Its value can be "highly recommended", "recommended" and "no recommendation".

The Excel framework reflects the implementation of which extended metamodel where we find the elements of a common metamodel of standards plus the new ones defined above to help for assessment.

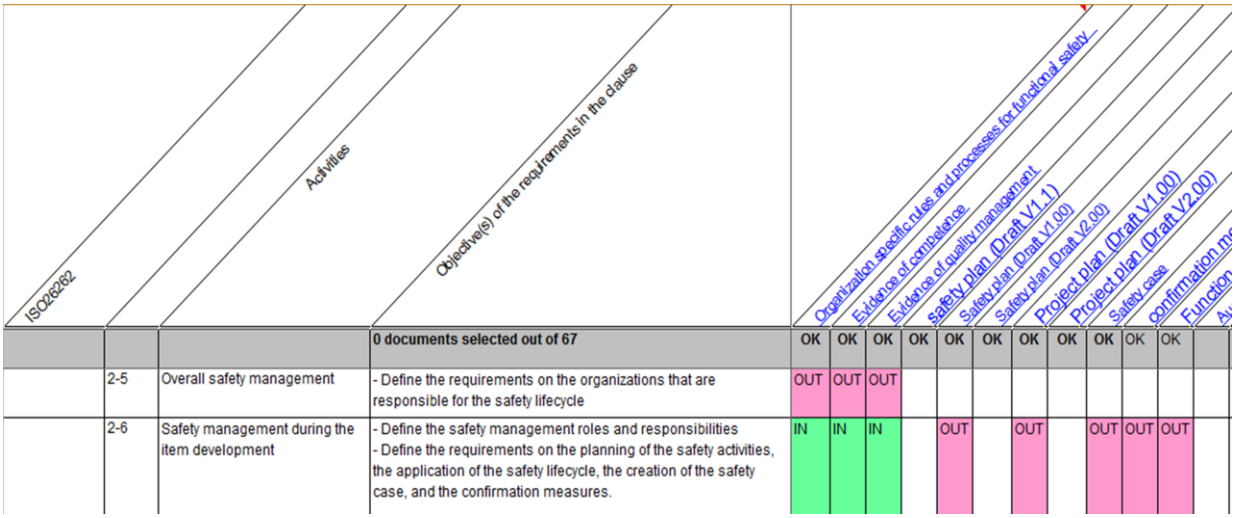
WorkPorduct	Item Definition	Selected ASIL = D	Role	System engineer					
Responsible		Level	1						

ID	Requirement text	ASIL	Recommendation level	Note	Example	Rating	Finding	improvement	Compliance	Reference
3-5.4.1	functional and non-functional requirements of the item as well as the dependencies between the item and its environment shall be made available. This information includes	All		This can include known safety-related incidents on similar items. -Requirements can be classified as safety-related after safety goals and their respective ASIL have been defined. -The required information is a necessary input for the item definition although it is not safety-related. If not already available its generation can be triggered by the requirements of this clause.		N			OK	ENG.2 BP1 ENG. 2- BP.2
a	functional concept, describing the purpose and functionality, including the operating modes and states of the item;	All				N			OK	ENG.2 BP1 ENG.2- BP.2
b	operational and environmental constraints	All				N			OK	ENG.2 BP1 ENG. 2- BP.3
c	legal requirements (especially laws and regulations), national and international standards	All				N			OK	ENG.2 BP1 ENG.2- BP.3
d	behaviour achieved by similar functions, items or elements, if any	All				N			OK	ENG.2 BP1 ENG. 2- BP.4
e	assumptions on behaviour expected from the item	All				N			OK	ENG.2 BP1 ENG.2- BP.4
f	potential consequences of behaviour shortfalls including known failure modes and hazards	All		This can include known safety-related incidents on similar items. Requirements can be classified as safety-related after safety goals and their respective ASIL have been defined.The required information is a necessary input for the item definition although it is not safety-related. If not already available its generation can be triggered by the requirements of this clause.		N			OK	ENG.2 BP1 ENG. 2- BP.5
3-5.4.2	Define boundary of the item, its interfaces, and the assumptions concerning its interaction with other items and elements	All				N			OK	ENG.2 BP1 ENG.2- BP.5
a	elements of the item	All		The elements could be also based on other technology		N			OK	ENG.2 BP1 ENG. 2- BP.6
b	assumptions concerning the effects of the item's behaviour on other items or elements, that is the environment of the item	All				N			OK	ENG.2 BP1 ENG.2- BP.6
c	interactions of the item with other items or elements					N			OK	ENG.2 BP1 ENG. 2- BP.7
d	functionality required by other items, elements and the environment	All				N			OK	ENG.2 BP1 ENG.2- BP.7
e	functionality required from other items, elements and the environment	All				N			OK	ENG.2 BP1 ENG. 2- BP.8
f	allocation and distribution of functions among the involved systems and elements	All				N			OK	ENG.2 BP1 ENG.2- BP.8
g	operating scenarios which impact the functionality of the item	All				N			OK	ENG.2 BP1 ENG. 2- BP.9

8-10 (1) 8-10 (2) SUP.1 SUP.7 SUP.8 SUP.9 SUP.10 ACQ.4 ENG.1 ENG.2 ENG.3 ENG.9 ENG.10 MAN.3 Feuil2 3-5 (2)

Figure 4-27. Extract of a set of ISO26262 requirements presented following the common metamodel. A requirement has typically some attributes: Clauses, Requirements, ASIL, Recommendation level, Notes, Examples and Rating. For a SPICE requirement, the ASIL attributes is defined at value “All”

Note that we rely on the availability of deliverables to assess the level of maturity because, in principle, a process is validated only when all its output workproducts are available.
Our resulting metamodel allows us to get the information about the (required or optional) output and input workproducts of each process. In Figure 4-18 below, we show how this information is represented. For all the activities or processes of a standard, we define these objectives and these workproducts specifying whether they are input or output objectives. The output workproducts of one activity are in general inputs for others.



			0 documents selected out of 67	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	
	2-5	Overall safety management	- Define the requirements on the organizations that are responsible for the safety lifecycle	OUT	OUT	OUT									
	2-6	Safety management during the item development	- Define the safety management roles and responsibilities - Define the requirements on the planning of the safety activities, the application of the safety lifecycle, the creation of the safety case, and the confirmation measures.	IN	IN	IN		OUT		OUT		OUT	OUT	OUT	

Figure 4-28. The input and output workproducts per workproduct and per activity (processes). Workproducts are identifies horizontally while activities are presented vertically. Between them, the information of which workproducts are input or outputs are defined.

Similarly, a deliverable is available only if all requirements to which it refers are satisfied (see Figure 4-19).

$$WP_i.\,rating = \Sigma_S\{Req_i.\,rating\} \mid \forall a \in \{Req_i\},\ WP_i \in \{a.isSatisfied\}$$

$$Process_i.\,rating = \Sigma_S\{WP_i.\,rating\} \mid \forall a \in \{WP_i\},\ Process_i \in \{a.isSatisfied\}$$

Figure 4-29. Assessment algorithm rules

We believe that focusing the assessment on requirement level, and not on base practice for example as being commonly done, it ensures a more detailed assessment. Remain that the base practice is defined as an activity that addresses the purpose of a particular process, identifying, at an abstract level, *"what*" should be done without specifying *"how*" [AUT 10a].

Table 4-3. Example of workproduct declaration in ISO26262 and SPICE. For ISO26262, the safety case ID is 6.5.3 and it results in an output workproduct from the clause 6.4.6. In SPICE, Contract ID is 02-00 and it results in an output workproduct from Outcomes 1 to 7

	ISO26262	SPICE
Workproduct	6.5.3 Safety case, resulting from 6.4.6.	02-00 Contract [Outcome 1-7]

We have defined an Excel spreadsheet for each workproduct with all requirements relating to it. Then, we have as many Excel spreadsheets as available workproducts (Table 4-3). We use the SPICE rating scale [AUT 10a, HOE 08] generically to assess the satisfaction status of a requirement in the framework, that means the values *"N", "P", "L", "F"* (Table 4-4) *and we add the value "N/A" (Not Applicable)* for follow-up questions.

In SPICE, as well as in ISO26262, a requirement may contribute to multiple workproducts. Our method avoids redundant work because once a requirement is validated, it will also be validated wherever it is specified.

In addition, each validated requirement automatically validates all relevant requirements whose references are in the *"Reference*" column. Several cases are possible to assign the rating in this case. For instance, let us consider the requirement *ReqA* from standard A which already has its rating value and its reference requirement *ReqB from standard B*:

- If requirement *ReqB* is completely covered by requirement *ReqA* (equivalent to*k*" value), then the rating assigned to requirement *ReqA* is automatically carried over to requirement *ReqB* (see Figure 4-20).

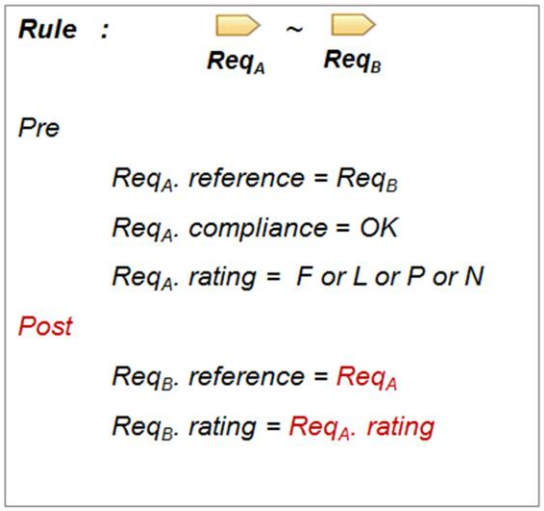


Figure 4-30. Assessment rule when two requirements in the different standards or completely compliant

- It requirement *ReqB* is partially covered by requirement *ReqA* (equivalent to "Partially" value), then the rating assigned to requirement *ReqB* is directly below that of requirement ReqA, when this is possible (see Figure 4-21).

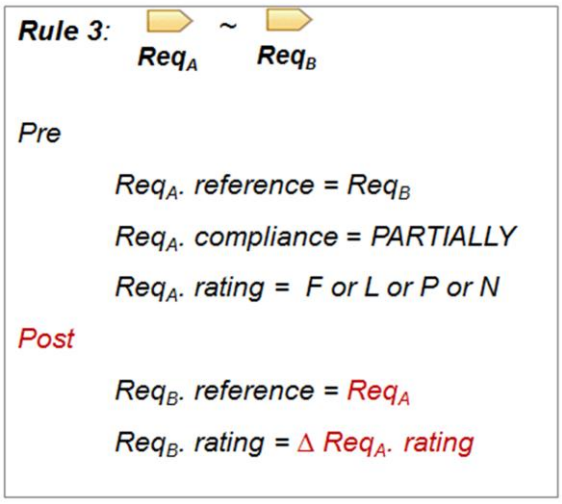


Figure 4-31. Assessment rule when two requirements in the different standards or partially compliant

The table below (Table 4-4) summarizes the different rating values applied to *ReqB* according to the *ReqA* rating in case of partial coverage.

Table 4-4. ReqB rating values applied following the *ReqA* rating value in case of partial coverage of a requirement *ReqB* by a requirement *ReqA*. Coverage of a requirement *ReqB* by a requirement *ReqA*

ReqA rating		ReqB rating
F		L
L		P
P		N
N		N
N/A		N/A

If requirement *ReqA* has no corresponding reference in standard B ("*NOK*" value), then obviously nothing is postponed (see Figure 4-22).

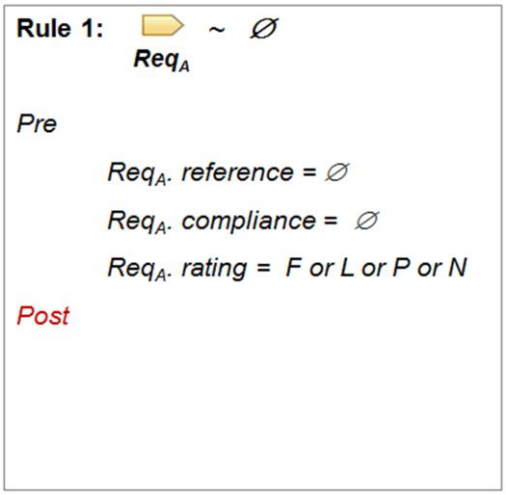


Figure 4-32. Assessment rule no corresponding reference exist for a given requirement

The staining colored tab gives a quick indication that a workproduct is available. The colors follow the same associated with the SPICE rating scale, i.e. green when all clauses are fully achieved (*F*), Yellow when they are largely achieved (*L*), orange when they are partially achieved (*P*), or red when they are not at all achieved (*N*) (see Figure 4-23), which by transitivity indicates the maturity level for each process.

Figure 4-33. Staining tab code color following the readiness of the deliverable. Each tab represents the deliverable of a certain process with its rank. For example 4-6 (1) is the first output deliverable of the subprocess 4-6 (clause 6 of part 4 of ISO26262)

4.2.4 Case study

We applied the framework to a trivial industrial case study. The process can be started indifferently with the SPICE requirements or the ISO26262 requirements, as the work done on one affects the other. Nevertheless, it is better to begin with the ISO26262 requirements because it has a wider spectrum and some studies have concluded that covering the ISO26262 standard (regardless of ASIL level) corresponds to covering capability level 2 of the SPICE standard at least. The opposite is not true.

We considered a subset of ISO26262, only activities associated with the specification of functional and technical requirements together with the system design (this corresponds to 9 clauses, 138 requirements and 30 workproducts). After performing the audit of all the requirements of a given standard, it is possible to verify, through a summary sheet, the maturity level for each process being evaluated, derived from a Workproducts rating (see Figure 4-24, see Figure 4-25).

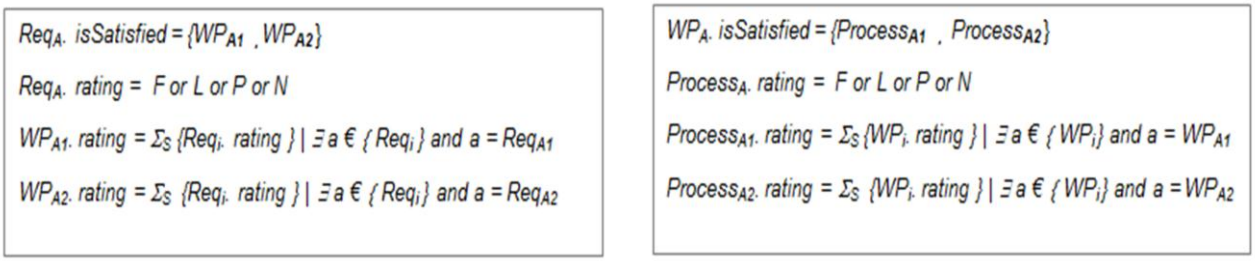


Figure 4-34. Method for deriving maturity level of ISO26262 processes

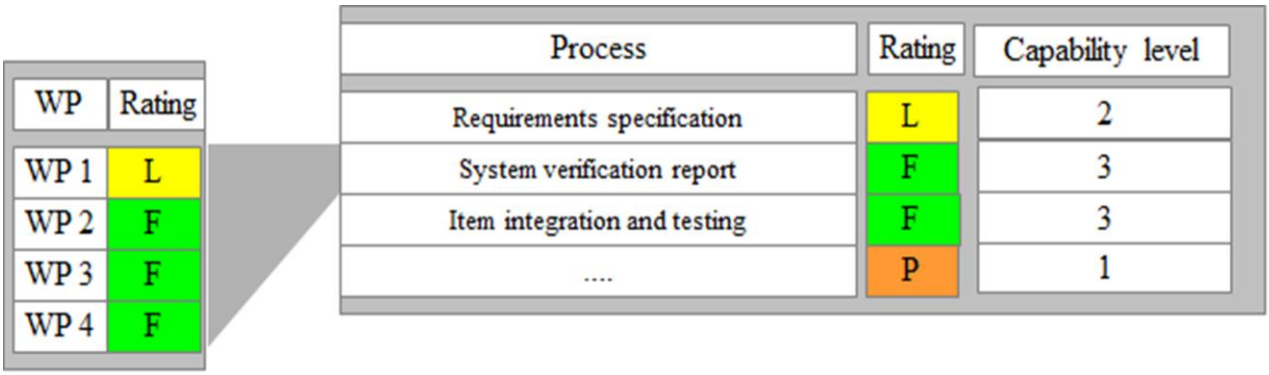


Figure 4-35. Method for deriving maturity level of ISO26262 processes

Furthermore, it helps to already have a partial evaluation of the other standard (see Figure 4-26). Having a partial evaluation of the other standard greatly helps since, to perform a complete assessment, it is only necessary to review, for each deliverable of this standard, the requirements that have not been automatically validated. This would be those whose reference column contains the *NOK* on the color tab in red.

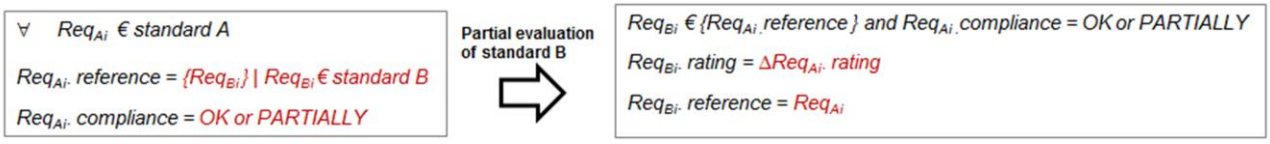


Figure 4-36. Method for deriving maturity level of ISO26262 processes

For the SPICE audit, we obviously have to switch to the terminology given in the standard (that means *Base Practice*, *Process Attribute, Generic Practice*, etc…, see Figure 4-27) that we can find again regarding the matching concepts (see Table 4-1). In our case study, 5 clauses (i.e. Processes), 42 requirements (i.e. outcomes), 30 workproducts and 30 base practices were concerned.

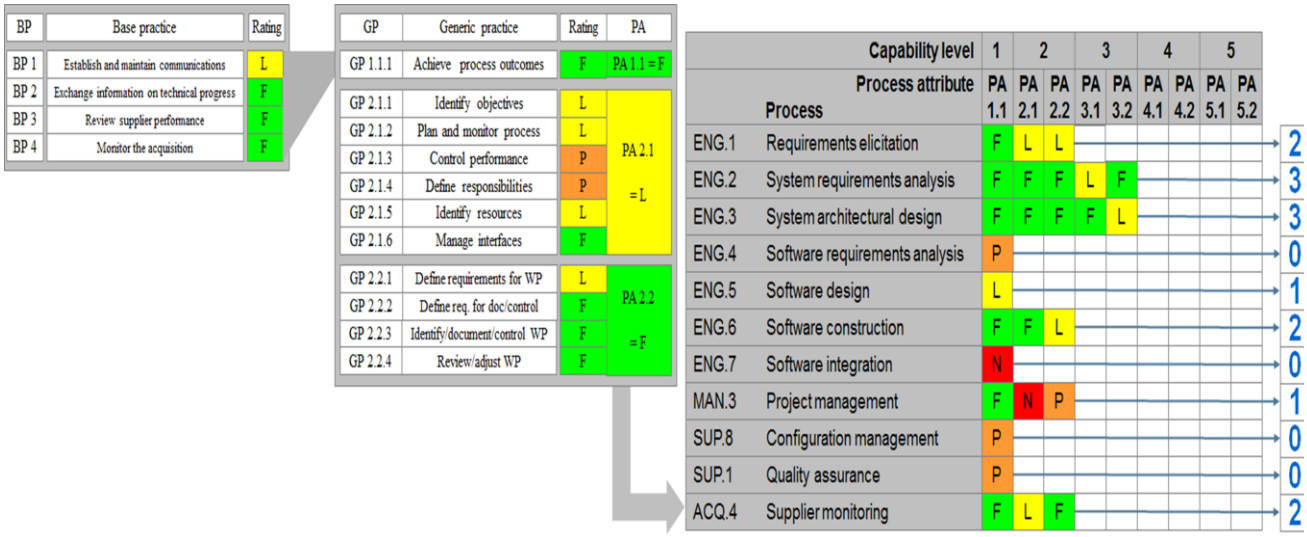


Figure 4-37. Maturity level Calcul of SPICE processes. The number in blue (from 0 to 3) represents the final maturity level achieved by the process derived from rating on their process attributes

The time and effort saving is undeniable as we avoid some redundancy in the requirements verification. Nevertheless, it should be used carefully in general, only in the scope of an audit-line evaluation for one project. Indeed, assessing processes in an organizational unit of a certain capability level means much more than requirements conformity: for instance, level 3 in SPICE means having the processes institutionalized in the organization and a simple requirements conformity is not sufficient enough to judge this fact. It is also a first solution to evaluate the feasibility, costs and additional efforts that the full deployment of the ISO26262 standard would require on large scale projects within the organization.

4.2.5 Summary

Our work is an attempt to develop an instrument to measure the quality of products of an engineering organization that develops automotive safety systems. Inspired by the SPICE process assessment model which has already proved its worth in the automotive industry, we adopt the notion of the SPICE rating scale to determine the maturity of a product which would be fully compliant with both automotive standards. We propose a SPICE-based model approach for assessment of safety engineering regarding the ISO26262 automotive standard. The approach has been experimented on a subset of the standards. The major benefit of our proposal of such an integrated assessment process is that it reuses the practices already present in the industry thus reducing the efforts of introducing the new standard. Preliminary presentations of these results were besides published in [7, 8]. We nevertheless remind that the matching between the ISO26262 requirements and SPICE which is the foundation of some features for the assessment requires a significant review by the certification experts, although this does not undermine the proposed methodology.

The technology selected for the implementation of the framework is Excel. Nevertheless, given the amount of data and the algorithms implemented, we meet difficulties in maintaining or adding other features. It would be wise to find a more appropriate format to ensure an efficient and effective assessment as in [MES 09]. Then, inspired also by the tool-based approach for managing systematic compliance in multi-standards context in [KOW 12], we extend our work by equipping our assessment approach in a model-based framework. A comparison of our common metamodel with the SPEM metamodel suggests to us that it would be possible to translate it in this process language with some extensions to be developed to cover all our concepts. This part will be presented in the last chapter of this section. Before that, let's us explicit how some artifacts requested by the standards can be provided in a model-driven engineering framework. Indeed, we have defined an approach which allows fulfilling the recommendations of SPICE and ISO26262. The fulfillment of these recommendations generally manifest themselves either through artifacts like documents, statements or models representing different aspects of the system under development (requirements, architectures, test cases...). In the next section, we present our work to provide these models.

4.3 Safety profile for automotive product

In this chapter, we present a model-driven approach to take into account the needs defined in standards concerning all activities involved at requirements engineering level [POH 96, POH 11] in the development cycle of automotive systems. We noted in the previous section that some contributions were very close to our approach, but the bibliography denotes that no current work is able to tackle every part of our problem: we can cite the SYSML profile, DARWIN profile and EAST-ADL2 profile. The idea is not to redefine the wheel. For that, we draw our inspiration from these different approaches and we define an UML-based profile named Requirement Management engIneering according to Automotive Standards (ReMIAS). The ReMIAS approach aims to implement the normative recommendations of the field in models. The profile integrates the recommendations of ISO26262 and those of automotive SPICE standard like for example the requirements properties specification, the requirement allocation on an item or architectural element.
Furthermore, we must consider the practices and ensure the connectivity and the interoperability of our approach with the tools currently used in industry. We propose therefore to integrate all of these mechanisms, both our approach and existing ones, in a tooled platform for definition of secure and automotive compliant embedded systems, based on a modeling environment.
Thereafter, we explain how the Requirement Management engIneering according to Automotive Standards (ReMIAS) approach allows efficient modeling of requirements engineering for E/E automotive embedded systems. This work resulted in a publication in [3].

4.3.1 Requirements management

For the requirements modeling part, we propose to handle normative expectations in a semi-formal representation. We remind the reader that, in this thesis, we are only interested in the system engineering activities.
In the description of the successive Engineering Process Groups (called ENG) of SPICE which consist of processes that directly elicit and manage the customer's requirements, specify, implement, or maintain the software product and its relation to the system, we are concerned by only the first parts: *the requirements elicitation* (ENG1) part, *the system requirements analysis* (ENG2) part, and *the system architectural design* (ENG3) whose objective is to identify which system requirements

have to be allocated to which elements of the system. The ISO26262 standard also goes in this direction as it recommends that, in parallel with requirement capturing, the development of an architecture concept where the requirements will be allocated (see Figure 4-28).

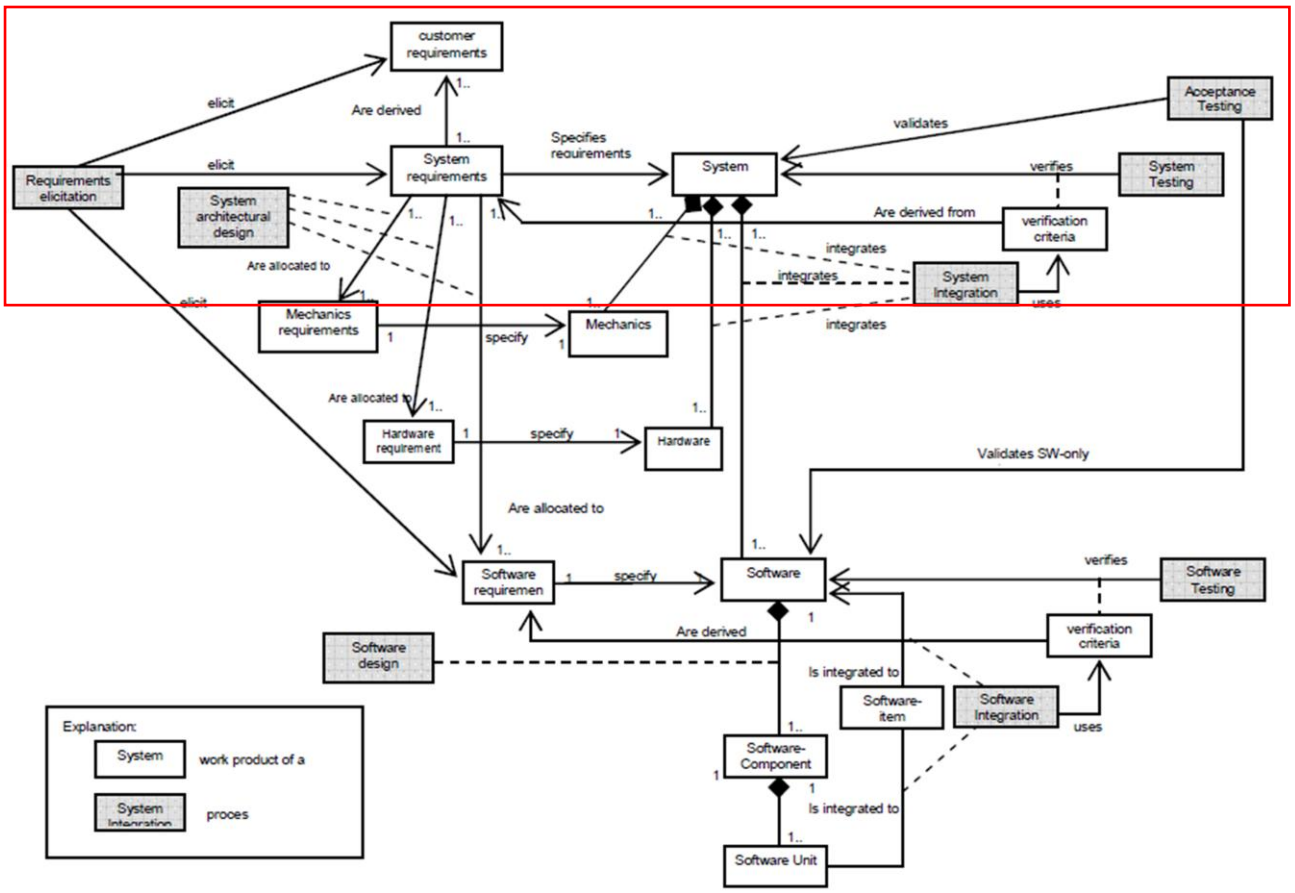


Figure 4-38. Key concepts schematic in terms of processes and work products through the engineering processes

We define The Requirement Management Engineering according to Automotive Standards profile based on the DARWIN profile for the requirement specification part and EAST-ADL2 profile for the architecture description part. Other references are done on SysML and MARTE[6] (Modeling and Analysis of Real-time and Embedded systems Object Management Group) [OMG 11d] about his library VSL for especially taking into account time aspects. In addition, to consider the normative recommendations of SPICE and ISO26262, the profile also manages traceability: traceability between requirements such as proposed by commercial tools, plus traceability between requirements and model elements like the SysML profile does.

4.3.1.1 Requirement specification

The requirement specification is tackled in the same way by both the automotive SPICE and the ISO26262 standards as they use more or less the same processes. During the elicitation and analysis activities [SOM 10], once the system requirements have been identified, analyzed in terms of technical feasibility, prioritized and categorized, they can be evaluated. The consistency of customer requirements with respect to system requirements ensured by establishing and maintaining

6 OMG MARTE, http://www.omgmarte.org/

bilateral traceability (including verification criteria) is also performed during the elicitation and analysis activities. According to SPICE, *verification criteria* define the qualitative and quantitative criteria for verification of a requirement. During these first steps, the main properties of the requirements such as classification, quality and characteristics are documented. ISO26262 summarizes them in the *"Specification and management of safety requirements"* chapter (ISO26262-8 chap. 6, see Figure 4-29**)**.

SysML and EAST-ADL2 implement some standard directives such as the formalism notation, the unique *ID* and the *ASIL* level for instance; but many other attributes are not taken into account, incomplete or poorly defined. One example is the *"status"* requirement which can have the *"proposed, assumed, agreed or reviewed*" values and which does not appear.

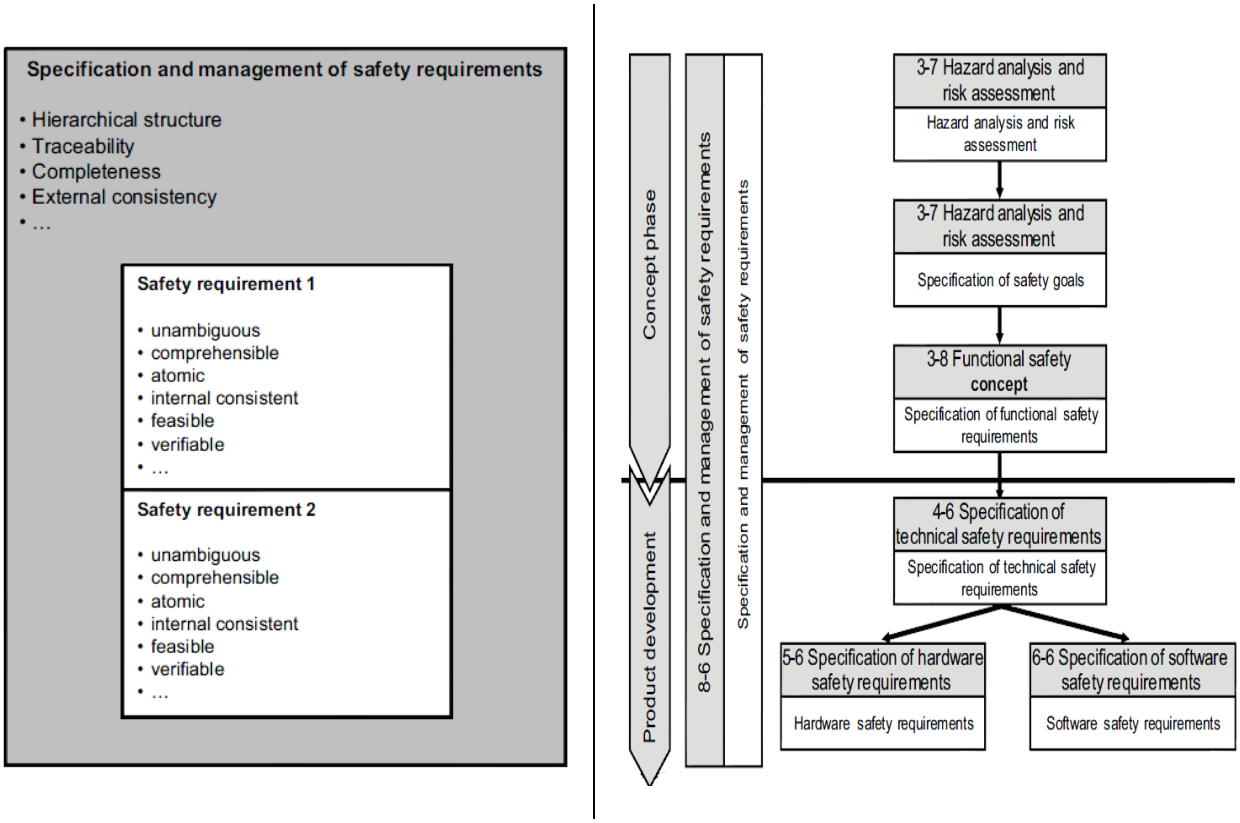


Figure 4-39. Safety requirements specification according to ISO26262

DARWIN somewhat mitigates this defect by proposing more attributes [ISO 10] such as *the author, the status, the verification result* which matches the verification criteria of automotive SPICE… Its traceability model extends the one proposed by SysML, so a requirement has a reference made to its realization in the design. Nevertheless, it always lacks a few quality characteristics such as specified by the ISO26262 standard. We extend DARWIN by adding the following properties in Figure 4-30 below.

Another weakness in many of the existing approaches is the requirement classification: the classification is not compliant with the hierarchical and organizational structure imposed by the ISO26262. DARWIN is based on the breakdown proposed by Glinz in [GLI 07]. The latter classify *"safety requirements"* such as quality requirements while in the ISO26262, any requirement (even a functional one) can be considered as a *"safety requirement"*.

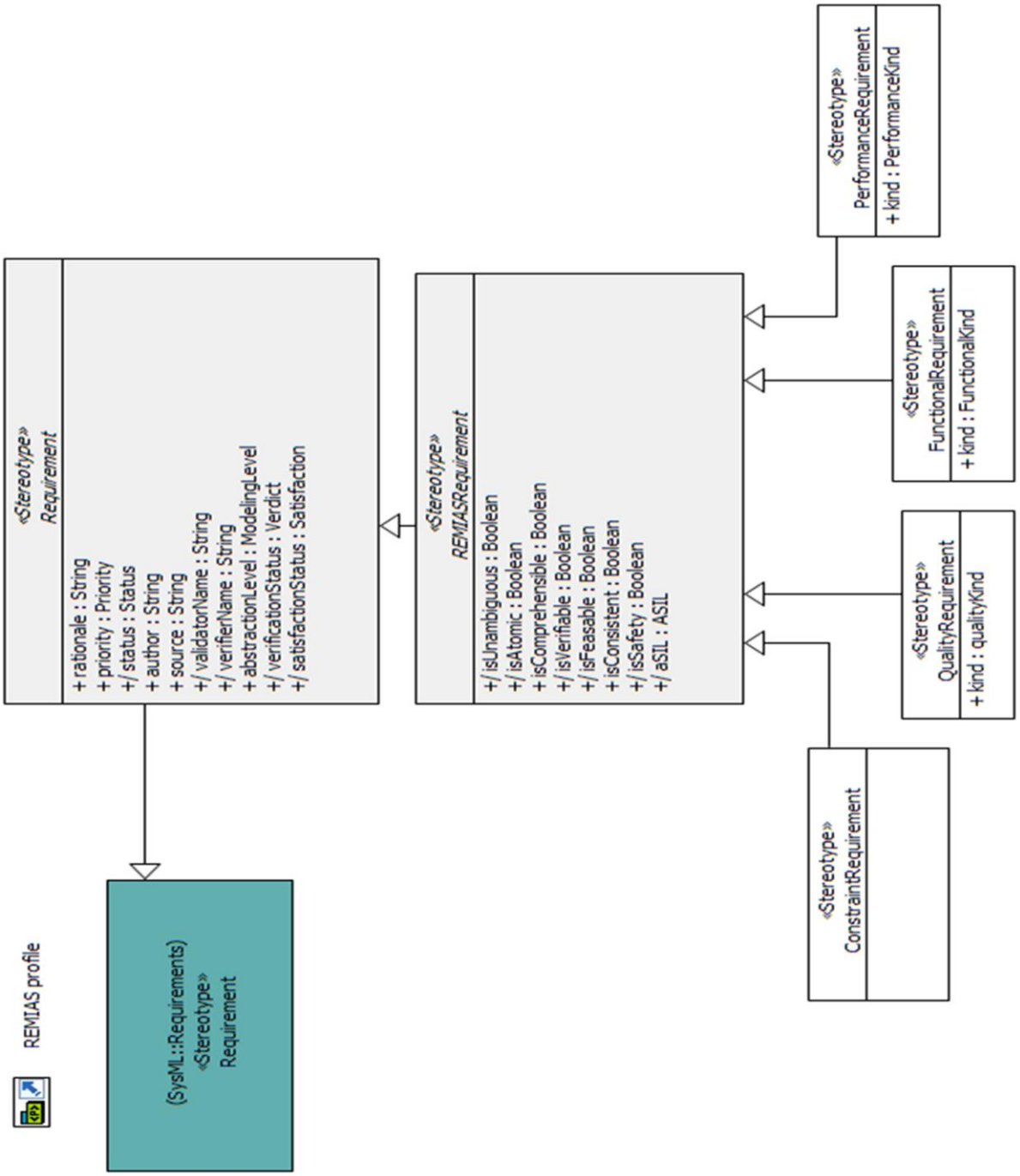


Figure 4-40. Extract of ReMIAS requirement profile package

properties, and the concepts that lead to their definition (see Figure 4-31); then, any requirement can be defined as a *"safety requirement*". By concepts that lead to requirement definition, we undermine mainly those relevant to the *Hazard analysis and risk assessment* (ISO26262-3 chap. 7) part of the safety standard. Indeed, as said *"Requirements can be classified as safety-related after safety goals and their respective ASIL have been defined"*. An ASIL is determined using the "*severity*", "*probability of exposure*" and "*controllability*" parameters for each hazardous event identified for the list of operational situations of the item under evaluation.

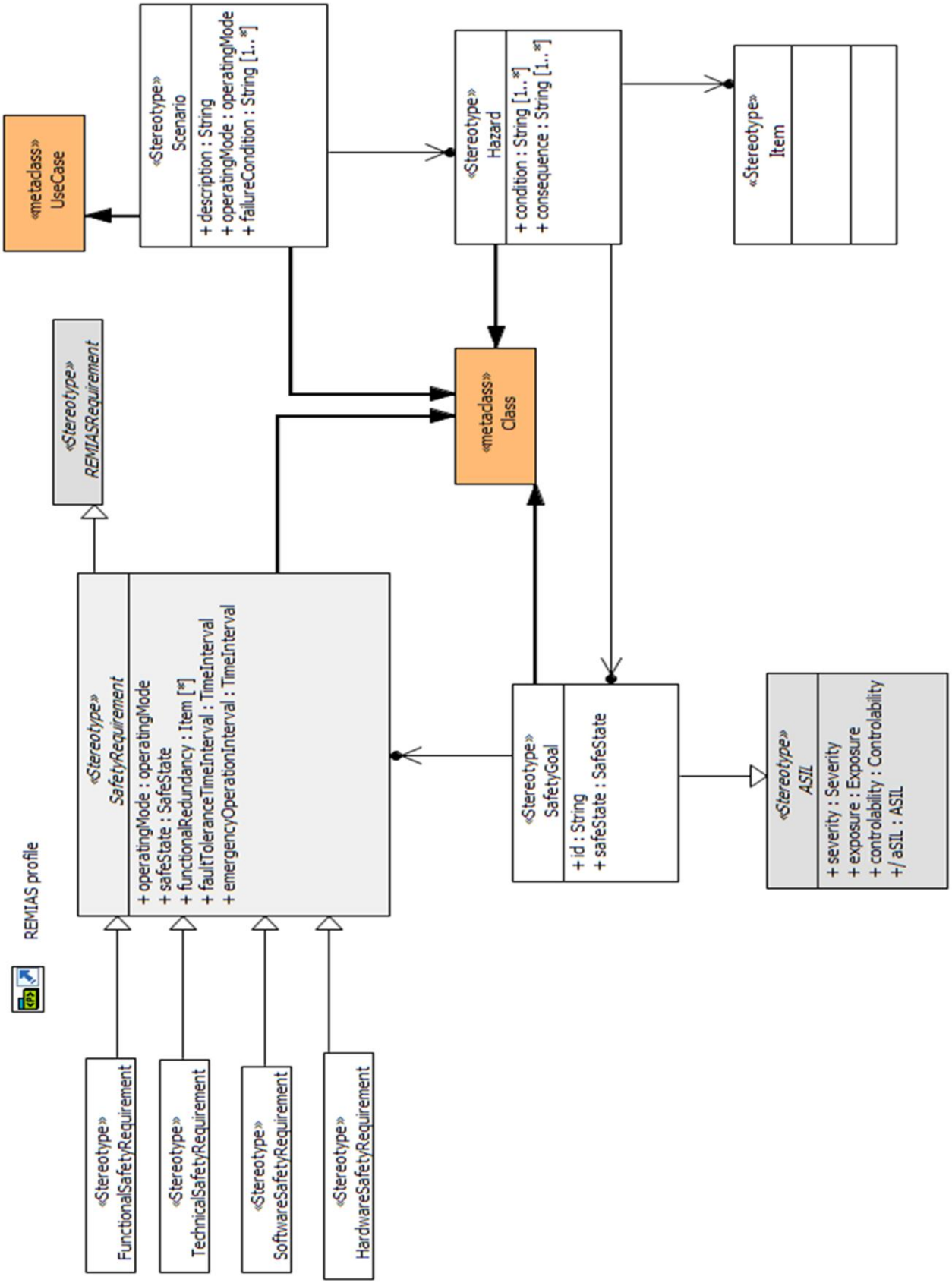


Figure 4-41. Extract of ReMIAS safety requirement package profile

4.3.1.2 Requirement traceability

The set of requirements shall have a hierarchical and organizational structure. Firstly, *Safety goals* are top-level safety requirements for the item. They lead to the functional safety requirements, which lead to the technical safety requirements. ReMIAS is defined as a static profile, i.e. rules have been implemented by code to allow automatic application: the hierarchy and organizational rules too.

The safety standard also states that safety requirements must be *"traceable to with a reference being made to a) each source of a safety requirement at the upper hierarchical level; b) each derived safety requirement at a lower hierarchical level, or to its realization in the design, and c) the specification of verification"*. That means that different traceability links exists:

- Derive link for traceability between requirements
- Satisfy link for traceability with architectural element
- Verify link for traceability with a verification or validation element

These traceability links have been implemented, extended from the DARWIN profile one (see Figure 4-32).

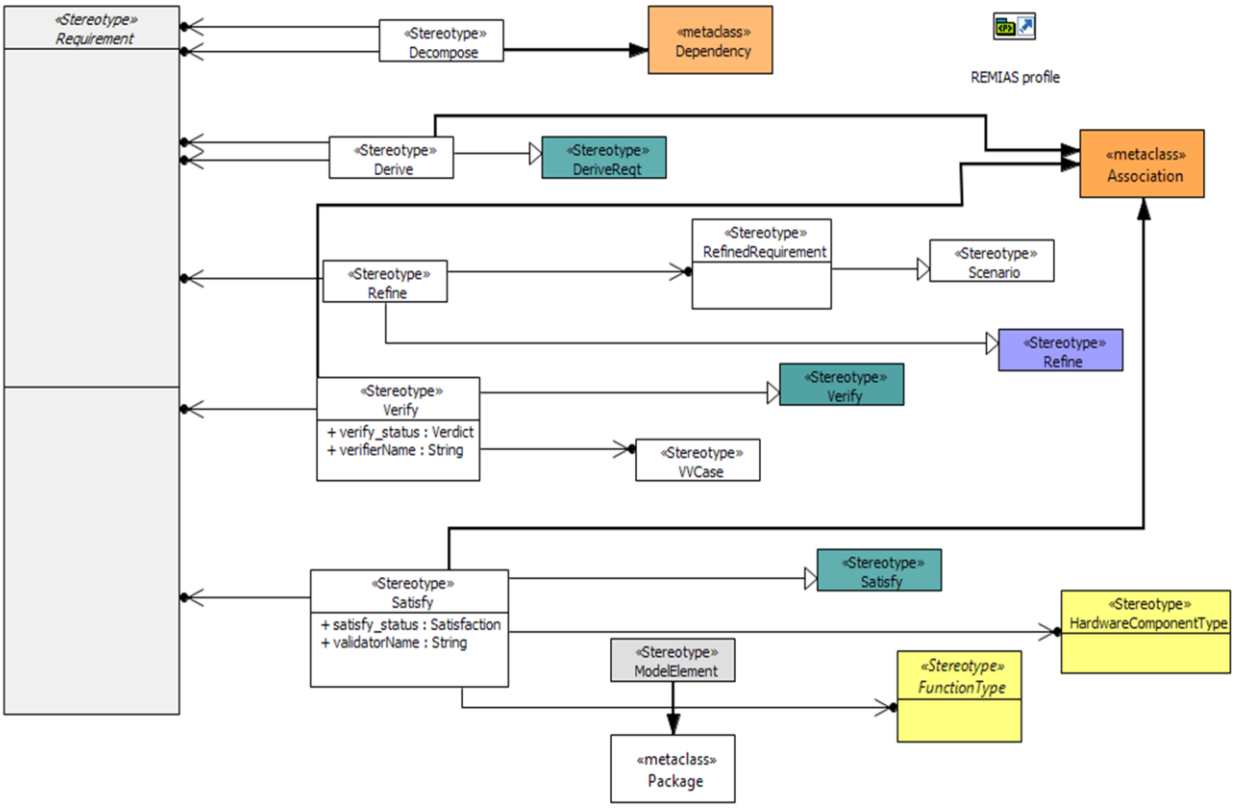


Figure 4-42. Traceability profile in ReMIAS

The last two traceability links refer to architecture and verification elements.

4.3.2 Architecture definition

Another characteristic of safety requirements is that they must be allocated to an item or an element. A critical point mentioned was the gap in the requirements allocation in a system architectural design (see Figure 4-33).

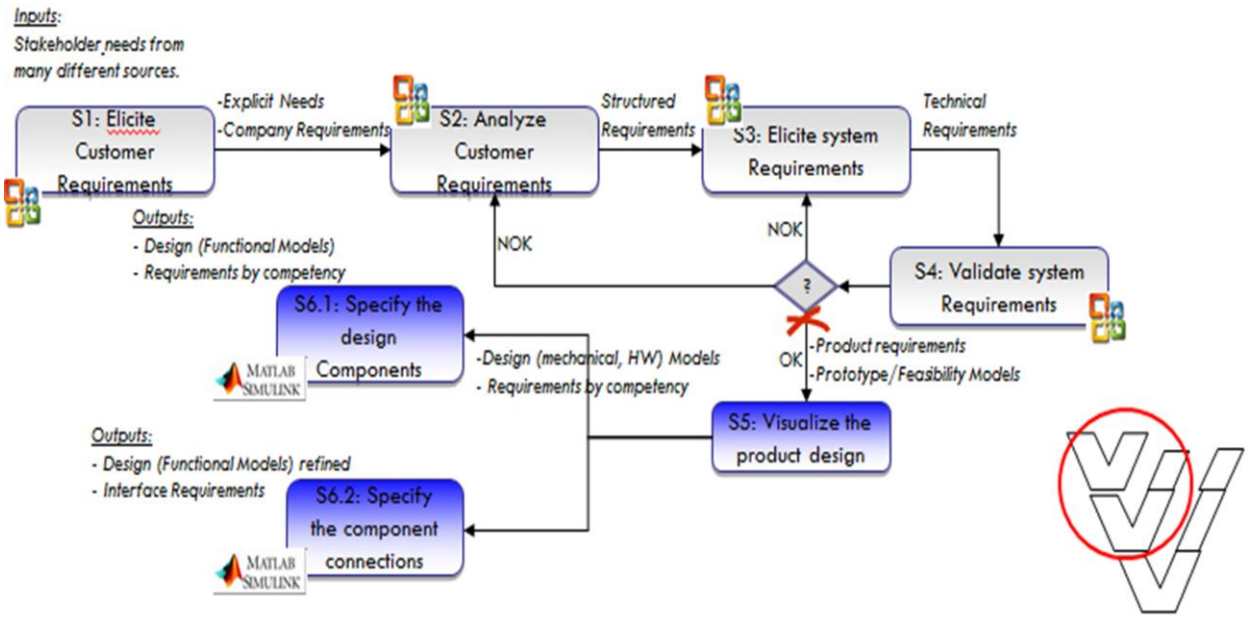


Figure 4-43. Generic System engineering process description

In ReMIAS, we have extended the DARWIN profile for requirement specification but it does not define architecture. There are many definitions of architecture that all agree on the fact that architecture is the overall structure of a system consisting of components and relations between them. Shaw and Garlan [SHA 96] define an architecture as *"a level of design that involves the description of elements (starting systems which are built), the interactions between these elements, models that guide their composition and the description of constraints on these models.*" Bass Clements and Kazman [BAS 03] also have a similar definition: *"a system structure that includes the software components, the externally visible properties of those components and the relationships between them"*. These definitions confirms also the SPICE one which defines the architectural design as a *"process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of the system"*. EAST-ADL2 proposes a good way to specify automotive architecture systems and respects these definitions. We then follow the actual EAST-ADL2 structure and add the traceability information (from the DARWIN extended traceability profile) in their traced design elements named *Functiontype* for functional architecture at analysis level and *HardwareComponentType* for technical architecture at design level. To respect the organizational structure imposed: safety requirements within each level are grouped together, corresponding to their architecture level.

One thing is to define an architecture but it must also verify that it complies with the requirements. The verify traceability link helps to ensure this objective.

4.3.3 Verification and Validation

ISO26262 specifies the properties and characteristics of requirements quality assurance (Verification and Validation) and the recommended measures and methods following the ASIL (Automotive Safety Integrity Level) of system to-be. The automotive SPICE also defines processes for the test and the integration of systems. We have made efforts in our work to support the verification and validation part (see Figure 4-34). It is inspired by the EAST-ADL2 profile.

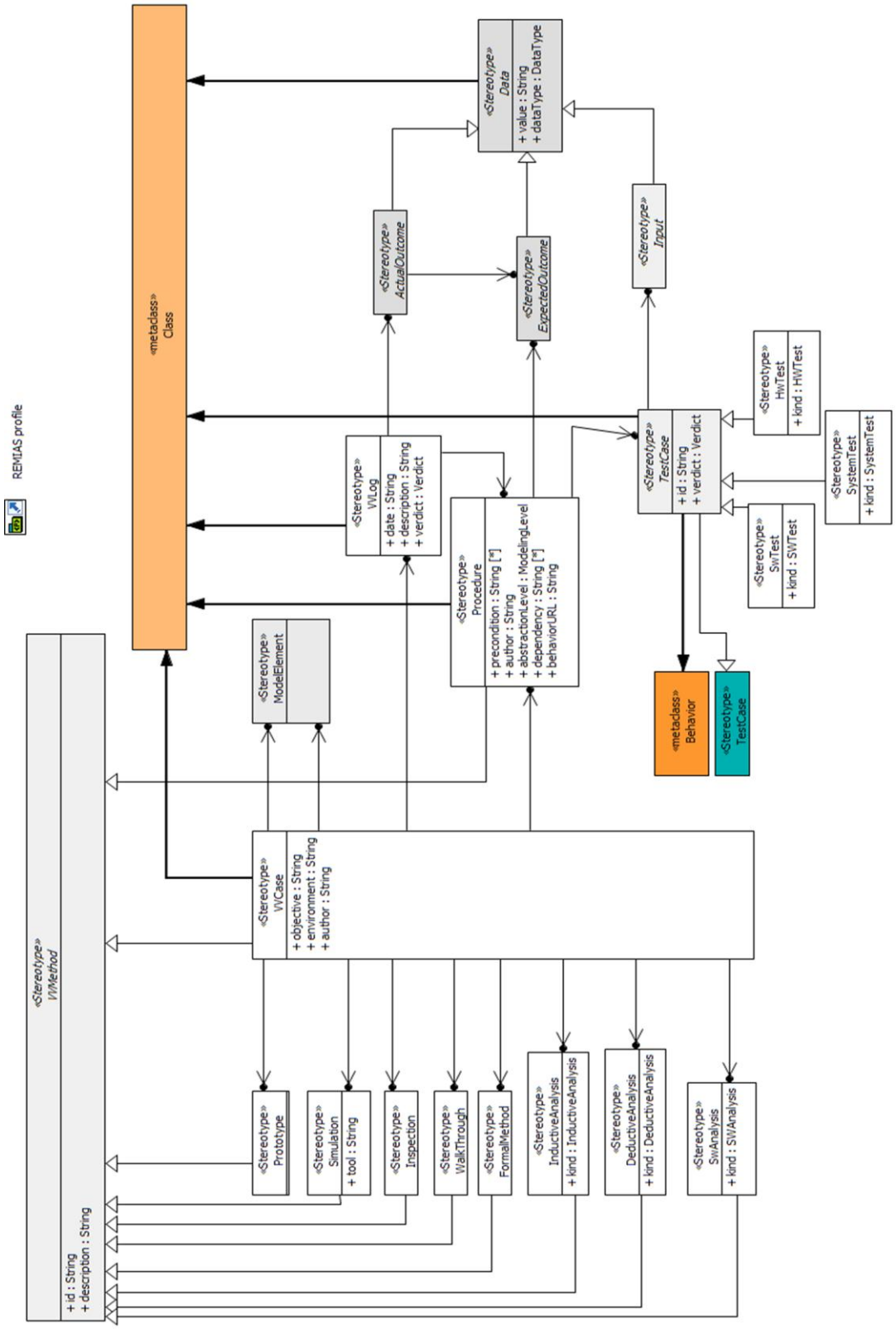


Figure 4-44. Extract of ReMIAS V&V profile package

Firstly, *pass* and *fail* criteria must be specified for each workproduct: it is represented by the *verification_status* property in the verify link. It should also be spelled the environmental conditions, the preconditions as the methods used for the verification. ISO26262 recommends many methods from simple review, to simulation, prototype or again test case, in adequacy with the complexity and the degree of criticality of the element to be verified. For testing, it is requested to specify in addition, the input data, the expected behavior (*ExpectedOutcome*) and their values. After execution, an unambiguous statement (*Verdict*) of the level of compliance of the verification results (*ActualOutcome*) with the expected results must be explained.

4.3.4 ReMIAS as static profile

ReMIAS is defined through the Papyrus MDT modeling platform. The profile affords compliance between automotive standards and system engineering. We use dedicated languages based on UML profiles allowing them to be combined into a more complete and still consistent language:

- SysML and DARWIN for requirements specification supporting the automotive concerns because it follows the classification induced by the ISO26262 standard as regards the properties and quality characteristics that it provides for their definition,
- EAST-ADL2 for specification design and verification & validation part.
- Based on traceability links inherited from DARWIN, we improve the requirement traceability by establishing links between requirements but also between requirements and design elements at system level in view of following the recommendations of two automotive standards: automotive SPICE and ISO26262.

In particular, the ISO26262 requires the application of the functional safety approach (see Figure 4-35). The main tasks related to our focus are:

- To identify and to define the item under safety analysis.
- To perform a hazard analysis and a risk assessment which leads to the definition of the safety goals
- To define the functional safety requirements and concepts.
- To define technical safety requirements and concepts derived from the precedents.

It also recommends a system design definition that complies with and implements the elicited requirements. This last recommendation converges partially with SPICE's ENG.3 step.

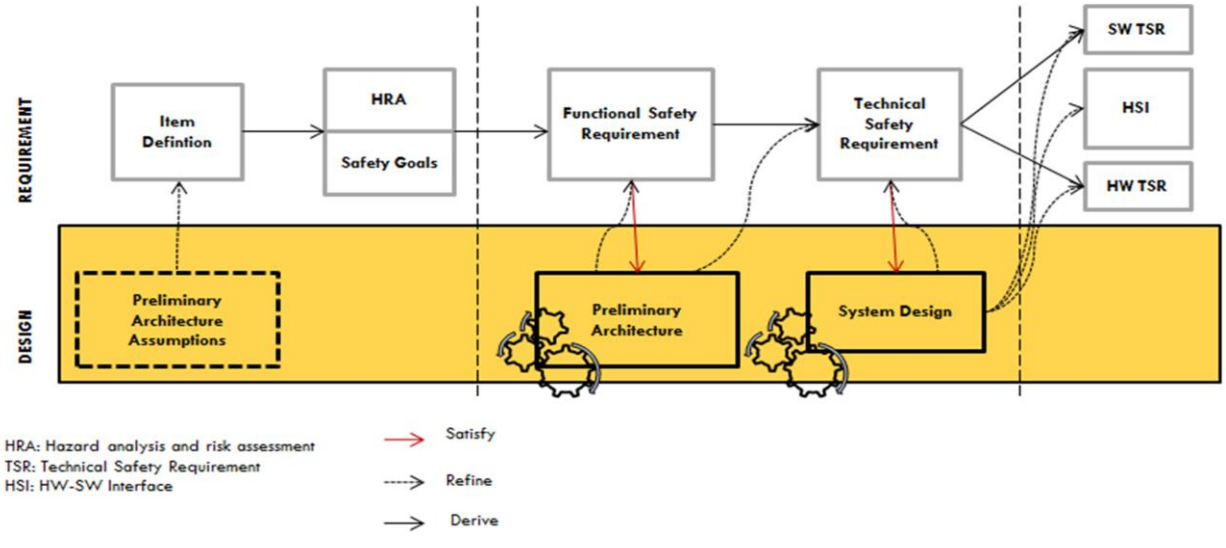


Figure 4-45. ISO26262 steps for safety requirements specification

The two first points are out of our scope as the determination of the safety goals together with their ASIL is an activity well-established in the automotive domain as part at top of the requirements management. This information is provided for us as input that we can describe with the ReMIAS profile. So, we are mainly interested by the next parts which implement the last two safety tasks listed above.

On this last point, defining the profile as static helps. A static profile allows rules to be added by coding to the profile to ensure some automation. In our particular case, we use this mechanism to implement some ISO26262 recommendations. Indeed, requirements management begins with the identification of *FunctionalSafetyRequirement* which inherits the ASIL's *safety goal* to which they are linked. With the ReMIAS profile, when a traceability link is defined between a requirement and a safety goal, the ASIL is automatically inherited. Note that a traceability link from a safety goal is only allowed with a requirement stereotyped *FunctionalSafetyRequirement* if the ASIL has the values *A, B, C, D*. Otherwise, if the ASIL value is *QM* (for Quality Management), the traceability is not possible with safety requirements. Similarly, the traceability link *derive* is allowed from *FunctionalSafetyRequirement* to *TechnicalSafetyRequirement*; it is the same between *TechnicalSafetyRequirement* and *Software* and *Hardware Safety Requirement*. Between safety requirements inside the same level (having the same stereotypes), only traceability links *refine* and *decompose* are permitted.

Another automatic set up is the decomposition of the ASIL, according to the recommendations defined in Chapter 9 of the ISO26262.

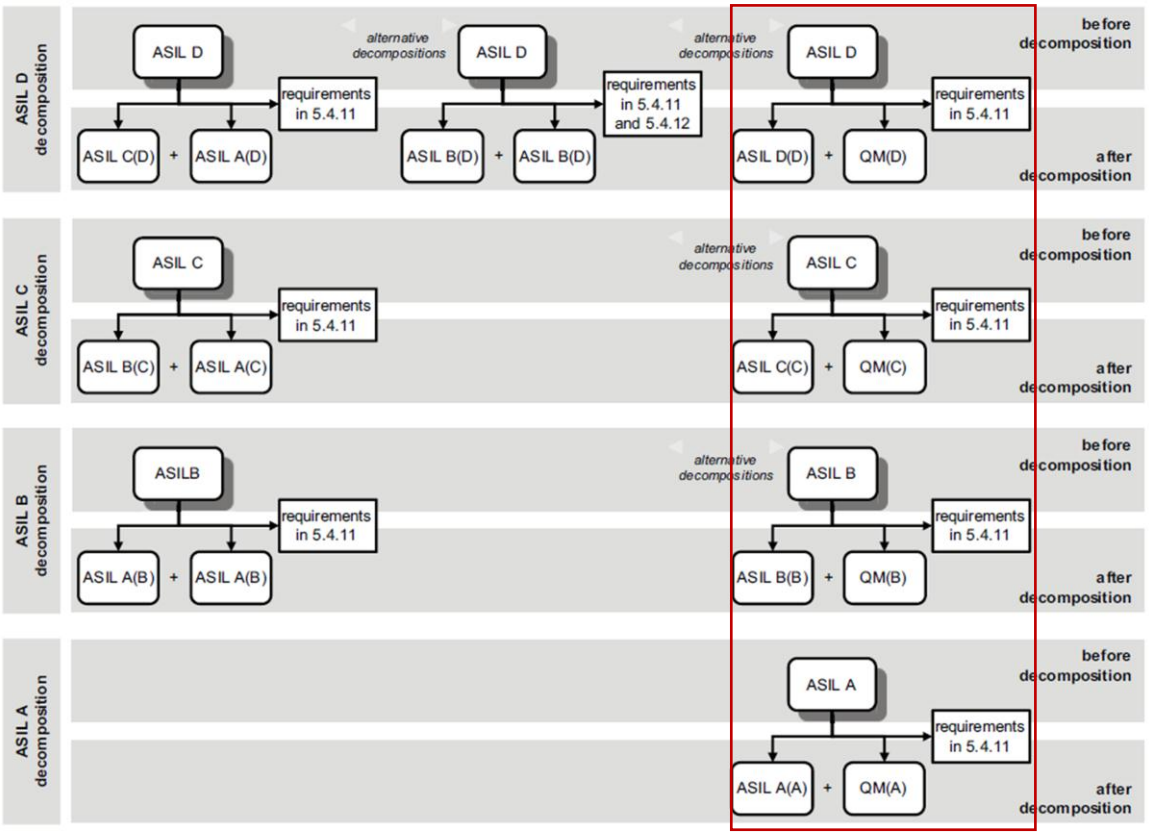


Figure 4-46. ASIL decomposition rules

In cases where many decomposition rules are possible, by default, this is for the one allowing the ASIL level *QM* (see red box in Figure 4-36 above) that is implemented. The user can change that, but for that the model remains valid, the choice must conform to the rules of the decomposition table.

Each time, the *isSafety* property of the requirement is adjusted according to the requirement of the ASIL. In fact, since a requirement inherits the ASIL, the value of the *isSafety* property goes to *True*, except when the ASIL is *QM*, as is the default case for all requirements non stereotyped safety.
Some other properties are inherited automatically with the profile. A requirement is *atomic*, when it is a leaf element, deducted from the hierarchy. By default, the property *source* of a requirement corresponds to the name of the original document where it was defined (customer requirements document). A requirement is *verifiable* when it is linked to a verification and validation element. Also, it is *feasible* when it is implemented by an architectural element. In this case, each requirement by level must be linked to a specific item type. Thus, *FunctionalSafetyRequirements* are linked with *FunctionType* kind elements and *TechnicalSafetyRequirement* with elements from design level. The traceability information appears in the requirement properties and in the component properties, as already defined in the EAST-ADL2 profile. In addition, other automatic rules present in EAST-ADL2 regarding the architectures definition (level of abstraction, inheritance shares, ports, etc....) are included. In the same way as for requirements, ASIL inheritance is done at architecture level: a component has the same ASIL as the requirements it implements (the higher one if it implements several requirements that have different ASIL levels). When a *satisfy* or a *verify* traceability link is established between two elements, the information is updated in the properties in the profile (*Satisfy, satisfyBy, derive, derivedby*...). The *verification status* and the *verifier name* (respectively *satisfaction status* and *validator name*) are also set up in the requirement.
All these rules implemented in the static profile let best comply with the requirements for traceability recommended by the ISO26262. Nevertheless, the traceability management has a weakness. The safety standard asks for traceability with each source of a safety requirement at the upper hierarchical level. Unfortunately, in an illustrative case taken from the automotive industry, the customer needs, the high upper hierarchical requirements level in general, are provided in text documents formats such as MS Excel™ or MS Word™. We need therefore efficient tools to document them in the modeling environment. We focus on solutions which allow automatic importing and exporting of the requirements from native specification documents without writing them manually as a model first. Our proposal also allows an export in the new OMG Requirements Interchange Format (ReqIF).

4.3.5 Requirements exchange from specification documents to different formats

In our approach, the requirements exchange is performed between simple MS documents and a requirements model into the Papyrus MDT[7] tool. In the approach, we also generate documents conforming to ReqIF format. We may consider a roundtrip scenario, as illustrated in Figure 4-37 below:

- The first one is the importation of requirements previously defined.
- The second scenario is the exportation of requirements defined or modified by any tool.

7 Papyrus MDT, wiki.eclipse.org/MDT/Papyrus

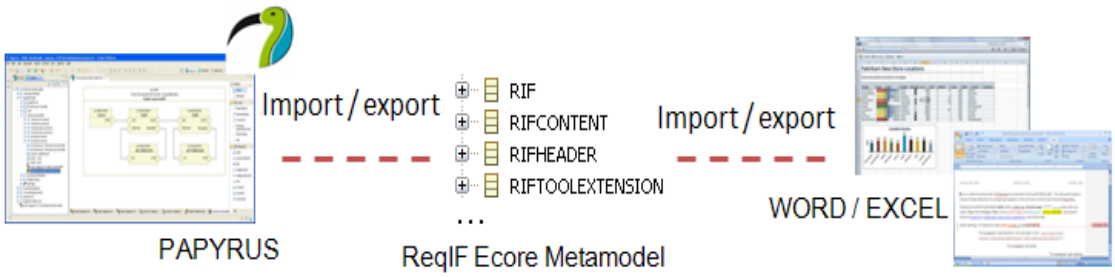


Figure 4-47. Requirements exchange using ReqIF from MS documents to MDT Papyrus tool

We start from use cases defined and inspired from the ReqIF specification for implementing our solution (see Figure 4-38).

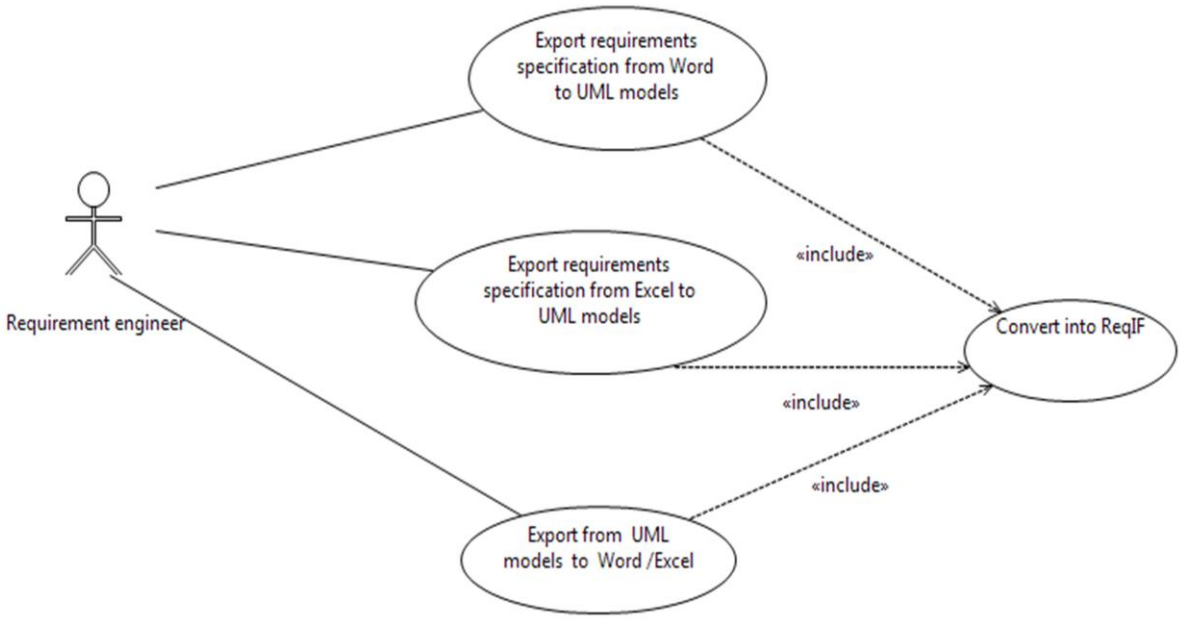


Figure 4-48. Uses cases implemented by Office2Papyrus plugin

For implementing the main use cases, we use mapping between the profile and MS document concepts.

4.3.5.1 Roundtrip from MS documents to models

For the main use cases, we consider the ReMIAS profile; and also SysML profile: as it is a profile widely used beyond the automotive domain, it can concern a larger public.
Think of an excel document, it allows the creation of an arbitrary number of sheets. Inside a sheet, the requirements are generally defined line by line. The line has columns which represent attributes. Considering the profiles, the requirement metamodel is composed of a set of requirements which are also attributes. These requirements can be grouped by package. With these concepts in mind, it is therefore quite simple to imagine mapping between the two representations (see Table 4-5).

Table 4-5. UML and Microsoft elements mapping

UML element	Microsoft document
An instance of requirement	Row
Requirement Type	N/A
Package	Sheet, Chapter
Attribute name	Column header
Attribute Value	Cell value

For traceability between requirements, we assume that for a given line (a requirement according to our mapping), a specific column of a sheet contains the list of *ID* of requirements which it is derived from. At each requirement *ID* in this list, a *derive* traceability link will be created, linking it to the *ID* of the requirement considered.
Word format is more difficult to map to requirement as its content can follow many different templates. We therefore assume that our requirements in Word will follow a tabular template such as Excel (Table 4-6).

Table 4-6. Example of requirement template in Word document

Requirement ID	Attribute 1	Attribute 2	Attribute x	Reference IDs
Requirement ID Tag	Text	Text	Text	Reference ID tag1 Reference ID tag2 …

We can then easily use the same mapping. A *package* is represented as a chapter in this case. We also manage the different elements that can be embedded in Word such as the pictures, the pdf files, any others Microsoft documents (Excel, Word, Visio, PowerPoint, etc …). These elements are represented as *comment* UML elements with the path where we can find them.
To be accessible to commercial tools, we also define a mapping with ReqIF.

4.3.5.2 Through the generation of a ReqIF model

The Requirements Interchange Format has been set up with the goal of exchanging specifications between *"modern requirement authoring tools"* [HIS 08]. Some already existing tools allow the creation of requirements models compliant with the ReqIF format such as ProR [JAS 10], but it is not our goal. Commercial tools like DOORS and Reqtify have already incorporated an importer/exporter for ReqIF. Several other companies, like MKS, or Atego (ARTiSAN Studio Tool vendor) are also working on an implementation of the ReqIF specification. EAST-ADL2 also proposes a RIF importer/exporter extension, as illustrated in Figure 4-31. There are some differences between the RIF and the ReqIF standard. RIF, defined by ProSTEP iViP Association [PRO 12], is the previous version of ReqIF, which was initially built for automotive domain applications and which was not normative. As ReqIF is built in the same way as the RIF format, it should be easy to replace the usage of RIF in EAST-ADL2 in order to be compliant with the new ReqIF formalism. This would mean replacing the RIF metamodel and model references (see Figure 4-39 below) by those of ReqIF.
Nevertheless, from the evaluation of use cases performed in the context of the European CESAR project [CES 09], some major issues are found: the importation does not work if the ReqIF file is generated from another tool. For example, Reqtify tool can only import and assess a ReqIF file generated with Reqtify and it is the same case for DOORS tool and EAST-ADL2 model.

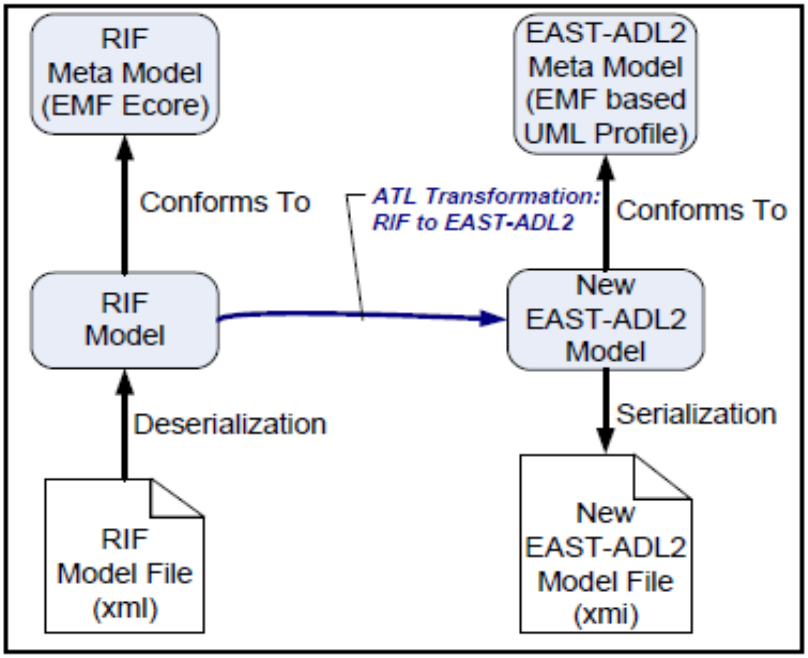


Figure 4-49. Method for deriving a new EAST-ADL2 Model from a RIF Model

Since the ReqIF file formats exported are different (different implementations of the same standard), the tools cannot access each other's exported data. The use of the ReqIF format is strongly dependent on its implementation in the RM-tools. In this context, the interoperability is always established on a tool-by-tool basis. Our approach avoids this problem.

4.3.5.3 Mapping Proposal for ReqIF concepts

Both ReMIAS (as well as SysML) and ReqIF have the concepts of a *requirement*, of a hierarchical structure (*package*… in UML model; *SpecGroup*… in ReqIF) and the possibility of defining relationships between requirements. Some correspondences between the modeling concepts and ReqIF concepts can be implicitly inferred based on the mapping table proposed by ProSTEP iViP Association and presented in Table 4-7. Let us take for example the *"Requirement"* type with attributes such as *ID* and *text*. You could create a *SpecType* named *Requirement* with matching *AttributeDefinitions* for *ID* and *text*. The *AttributeDefinitions* could also have *DatatypeDefinitions*. Based on the defined *SpecType*, you create *SpecObject* having *AttributeValue* for all the *AttributeDefinitions*.

Table 4-7. Mapping between ReqIF and UML concepts

Model element in ReqIF	UML concept
SpecObject	An instance of requirement
SpecType	Requirement Type (Requirement in this case)
SpecRelation	Derive ➔Trace
SpecGroup	Package
AttributeDefinitionSimple	Attribute name
AttributeValueSimple	Attribute Value
DatatypeDefinitionString	Requirement Type

SpecRelation in ReqIF qualify a traceability link and is defined by a source and a target element (as *SpecObject)*. We can notice in Figure 4-40 below that traceability information appears as attributes of the requirement in UML.

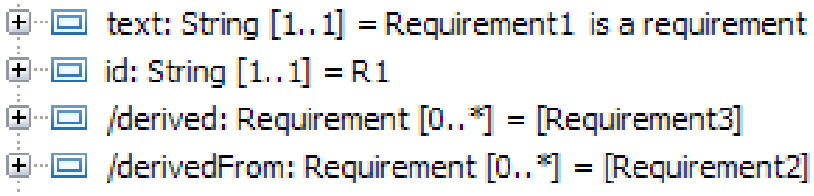


Figure 4-50. Requirement attributes in a SysML Model

The mapping here will be with *SpecRelation* concept from ReqIF: each reference element will be a *SpecRelation* whose source is the current requirement and target the relevant reference. The ReqIF detailed Header class (*Author, Comment, Creation Time, Identifier, Source, ToolsID* and *Title*) is also considered. The other ReqIF elements are defined but are outside the scope of our example.
For the mapping between The ReqIF concepts and the Microsoft document, a *SpecGroup* is represented by all the requirements of a sheet or of a chapter in Word. A *SpecObject* represents the actual requirement, and compared with Microsoft document, a row in a sheet or a table represents a requirement. A requirement typically has a number of Attributes; in MS Excel™, each row has the same columns which are the attributes, and each cell contains the value of an attribute. It is the same for a table in word document. Therefore, attributes of a requirement in our model will correspond to the columns (see Table 4-8). We have also defined above that a specific column contains a list of the ID references of requirements from which the actual one used is derived. A *SpecRelation* is created for each of them.

Table 4-8. Mapping between ReqIF concepts and Microsoft concepts

Model element in ReqIF	Microsoft concept
SpecObject	Row
SpecType	N/A
SpecRelation	Each ID contained in the references ids column
SpecGroup	Sheet, chapter
AttributeDefinitionSimple	Column header
AttributeValueSimple	Cell value

According to the OMG organization, *"a compliant implementation of the Requirements Interchange Format (ReqIF) must implement all elements (…). Further, a compliant implementation must also recognize and support the high-level exchange protocol and associated exchange document (…)"*. To achieve this statement, we use the ReqIF concept *Embedded file* for embedded elements in a Word document.

4.3.5.4 Implementation

We implement the approach as an Eclipse plugin named Office2Papyrus that can automatically generate an UML file in Papyrus MTD from Microsoft Word / Excel. Generation operates in both directions. For this purpose, we use Ecore SysML and ReMIAS metamodels. The transformation also generates an intermediate format through an XML file that conforms to XML Schema ReqIF as illustrated in Figure 4-41.

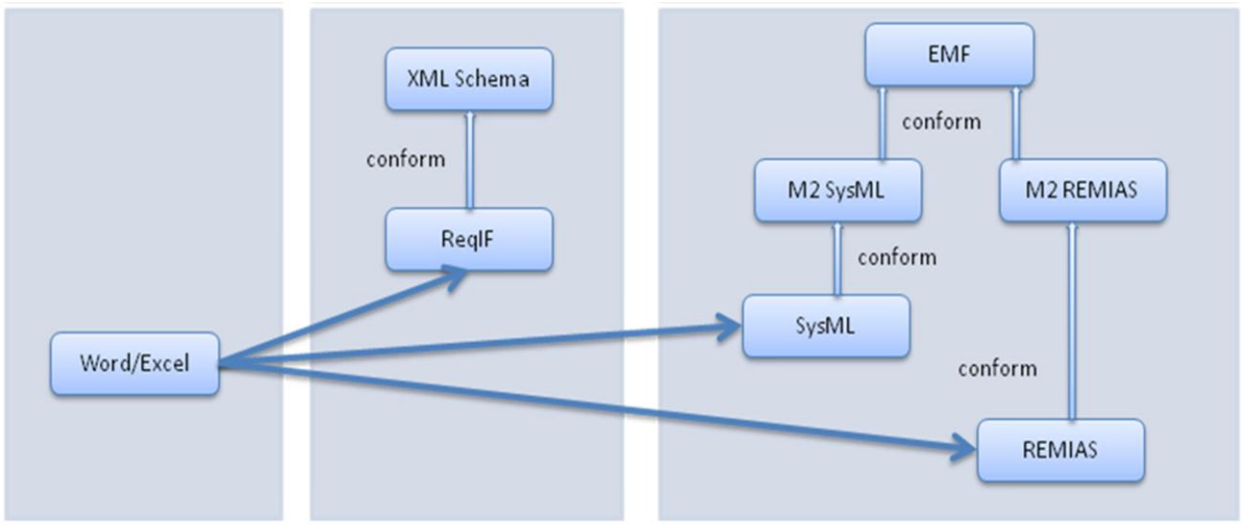


Figure 4-51. Office2Papyrus plugin functionalities

Our preceding use cases are detailed following different scenarios.

Table 4-9. Export requirements specifications from Excel to UML models

Use case name	Export requirements specifications from Excel to UML models
Actor	Engineer
Objective	The engineer imports an Excel document in Eclipse and converts it into an UML file
Preconditions	The document respects the structure defined valid to allow importation
Scenario	The engineer opens the plugin in the tool: • He imports a Word Document • He selects « Convert Excel to UML » • The document is converted into an XML ReqIF format • The document is transformed into an UML file
Post conditions	Two files are generated simultaneously: The XML ReqIF file and the UML file conform to SysML or ReMIAS
Exceptions	If the Excel document does not respect the preconditions, then the generation is not performed.

In our case, a document is valid when it respects the mapping rules presented above. The Excel sheet has as many columns as attributes. Office2Papyrus works with any template, owned or standard, and a requirement is represented on a row in a sheet with its attributes in columns. In case of a transformation to SysML, only *ID* and *text* attributes are defined, also with information about the derived requirements. A specific column contains the *reference ids* of the upper level requirements. Then three columns are considered. Otherwise, for ReMIAS for example, all attributes can be taken into account. An XML configuration file allows the attribute corresponding to each column header to be specified.

Table 4-10. Export requirements specifications from Word to UML models

Use case name	Export requirements specifications from Word to UML models
Actor	Engineer
Objective	The engineer imports a Word document in Eclipse and converts it into an UML file
Preconditions	The document respects the structure defined valid to allow an importation, which means requirements are defined in a table format
Scenario	The engineer opens the plugin in the tool: • He imports a Word Document • He selects « Convert Word to UML » • The document is converted into XML ReqIF format • The document is transformed into an UML file
Post conditions	Two files are generated simultaneously: The XML ReqIF file and the UML file conform to SysML or ReMIAS
Exceptions	If the Word document does not comply with the preconditions, then the generation is not performed.

In order to help identification and extraction of requirements tags and texts, a valid Word document contains a table defined in the same template as an Excel document (see Table 4-11).

Table 4-11. Requirement template in Word document

Requirement ID	**Attribute 1**	**Attribute 2**	**Attribute x**	**Reference IDs**
Requirement ID Tag	Text	Text	Text	Reference ID tag1 Reference ID tag2 …

The requirement extraction function is able to identify useful information in this graphical representation. A specific column, not necessarily the last one, contains ID tags of referenced requirements. All embedded elements (inside the columns table) are saved individually in a directory and an html page is created with a link to each of them. This html page is translated as XHTML tag in ReqIF and put as a comment in UML (the html path).

Table 4-12. Export requirements from UML models to Word/Excel

Use case name	Export requirements from UML models to Word/Excel
Actor	Engineer
Objective	The engineer imports an UML file in Eclipse and converts it into Word/Excel and ReqIF documents, in parallel
Preconditions	The UML file is valid, i.e. the model respects the requirements diagram definition constraints
Scenario	The engineer opens the plugin in the tool: • He imports a Word Document • He selects « Convert UML to Word/Excel » • The UML file is transformed into an XML ReqIF document • The UML file is converted into Microsoft Word and Excel documents

Post conditions	Three files are generated simultaneously: The XML ReqIF file, the Word document and the Excel document
Exceptions	If the UML diagram does not respects the SysML or ReMIAS constraints, then the generation fails.

We use the plugin Office2Papyrus in Papyrus MDT, where from an UML file, we can have the corresponding graphical representation that shows that the UML file produced is valid: SysML, ReMIAS or any other UML profile, as long as in the configuration file, you specified the correct parameters for the mapping rules between attributes and MS documents templates. We have verified with an XML tool that the ReqIF XML file generated is compliant with the OMG ReqIF metamodel. The plugin is robust as it can manage files with thousands of requirements.
This approach is thus not specific for requirements exchanged between MS Excel™ or MS Word™ and MDT Papyrus tool and thus it can be adapted to other tools supporting XML ReqIF schema and UML metamodel.

4.3.6 Summary

One objective of the thesis was to identify in the different standards, the needs to be covered as part of requirements management and to introduce them in a modeling framework. The ReMIAS profile achieves this goal. We searched through the profile various means and methods of producing these automotive standard requirements which afford compliance between automotive standards and system requirement engineering. We use dedicated languages based on UML profiles with a combination of them in a more complete and still consistent language: SysML and DARWIN have given inspiration for requirements specification. The purpose of the requirements engineering process is to trace analyzed requirements to design elements for validation. Parts of EAST-ADL2 are extended for specification design and verification & validation part. The traceability is ensured, based on traceability links inherited from DARWIN. We have also investigated automatic importation from requirements specification documents into the modeling environment through an Eclipse plugin: Office2Papyrus which has proven its important worth by its publication in [6]. Our approach automatically documents a requirement repository into UML-based modeling tool from Microsoft documents using some specific templates, avoiding redundant work for requirement engineers, without changing their practices and without losing any existing information when they validate requirements on a design model. The result is that we can use modeling and text based approaches to specify requirements and integrate them in a standardized way. Other benefit of Office2Papyrus is that it can be used in any tool. We use the Requirement Interchange Format (ReqIF) standard to confirm this statement. We think that the tool vendors will progressively align with the correct ReqIF specification, which will improve interoperability between commercial tools (including requirement management, design modeling tools) and our plugin Office2Papyrus.
With this contribution, we are able to define the product in a detailed model that integrates different views: the requirements and design phases can be traced throughout the complete development process; we talk about a product model which is built regarding the standards demands through the engineer's activities that lead to this concrete product. The set of the engineer's activities represents the process model which describes the workflow to follow to fulfill the standards recommendations. The process model is the subject of the next chapter.

4.4 Metamodel for process development

ReMIAS supplies the language to describe a product according to a domain-specific level of development. It defines entities describing the aspects of the artifact under development and the necessary parts of its environment, as well as the relations between these entities. We finally obtain a product model that represents the concrete product to be delivered. The entities in the product model are outputs of activities performed during the development process and advised by the standards. Then an activity can be understood like a process pattern in the small as it is an atom of a process. Indeed, we have seen in section 3 that the justification of the product model is its application in the definition of a process model [SCH 02]. Below, Figure 4-42 presents a general description of a process model as described in automotive standards ISO26262 and SPICE and its mapping by activity or sub process on the different parts of the engineering development process together with their relationships on a V-cycle.

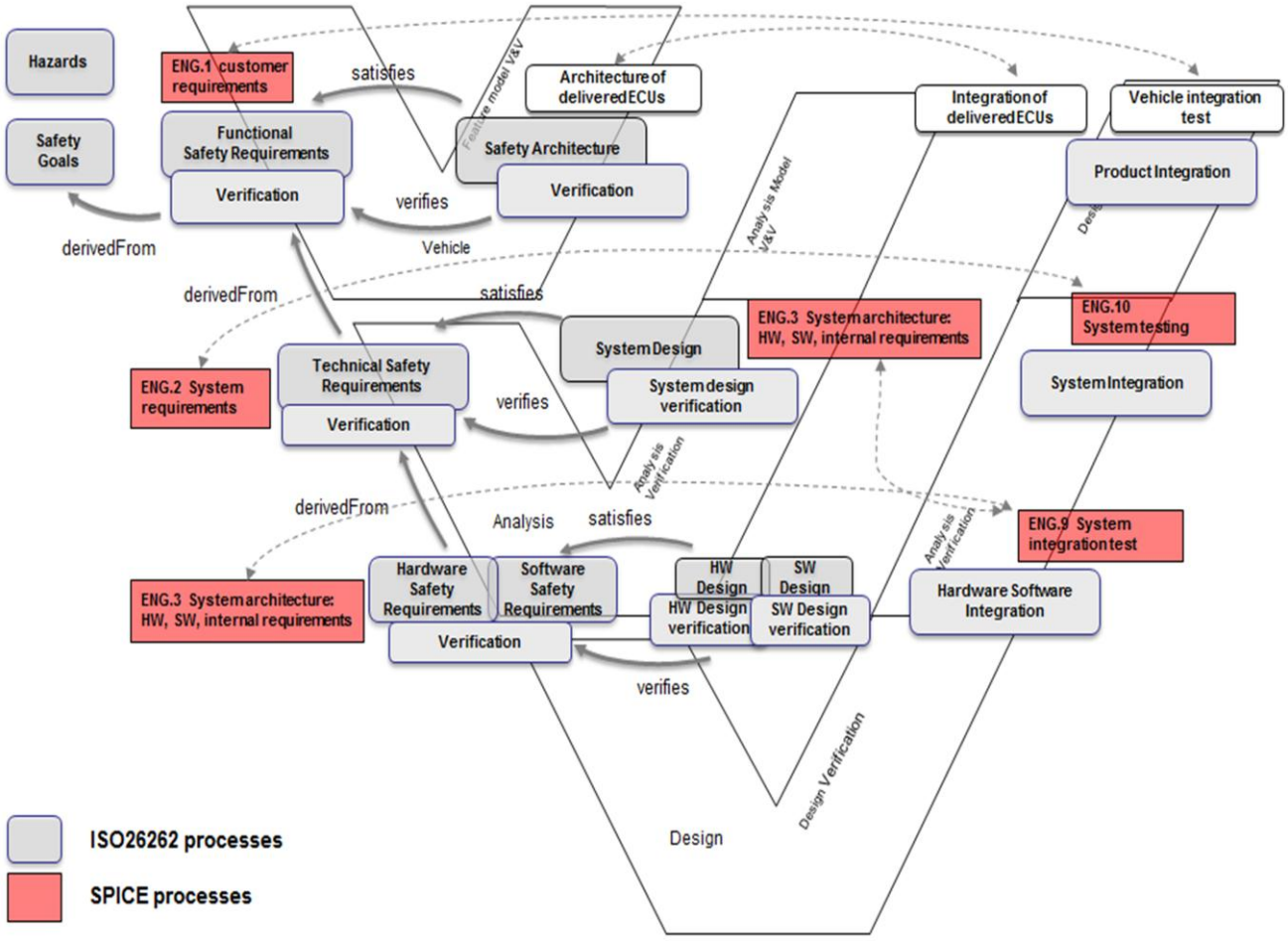


Figure 4-52. ISO26262 and SPICE processes on V-cycle

Such a model provides understanding of the traceability between system requirements from a development process and requirements from standards; i.e. the traceability between activity/process and system entities for project monitoring purposes. We have then identified tasks, techniques (how to perform a particular technical activity and, how to use a particular notation as part of that activity); methods (guidance for how to use the notations and techniques together to

perform a related collection of technical activities or achieve a particular goal); roles and workproducts from the standards (see Figure 4-43).

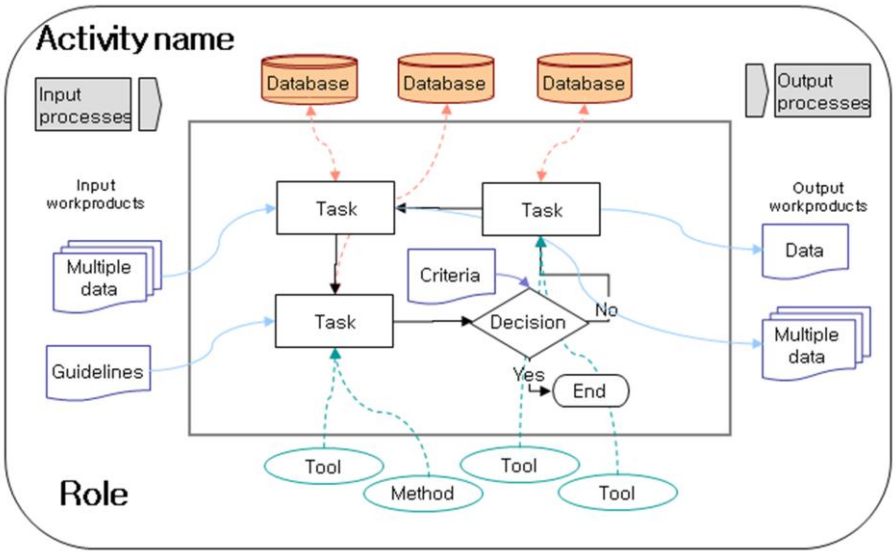


Figure 4-53. Example of an activity metamodel

In real terms, ReMIAS' entities are some workproducts of activities. But, as the UML metamodel, on top of which the profile is defined, does not define the concepts to handle the notion of process, task, role etc... the structural relations underlying the product and the process model are missing. Thus, it is necessary to formalize the latter in an appropriate manner to keep their dependencies expressed in terms of the product rather as in unspecific ways such as ad-hoc descriptions. We chose for this reason to use the SPEM language because it is derived from UML, aspects of which have been specialized to meet the goal of process modeling.

4.4.1 Structural process metamodel

Our process model must describe the standards recommendations that support the activities of development. In sum, we need to support the elements of our common metamodel defined above, obtained after merging of the standards concepts. Our analysis shows that many of these concepts are similar to the core concepts of the SPEM metamodel.

4.4.1.1 Core concepts

The core concepts of the common metamodel have semantically corresponding elements in SPEM (see Figure 4-44).

So, all specialized elements of *MethodContentElement* are semantically similar in both metamodels. An *Activity* defines basic units of work within a process as well as a process itself. In other words, each *Activity* represents a *Process*. In this sense, it can be mapped to *Clause* Concept in the standards metamodel. *ToolDefintion* can be mapped to *Tool*; as well as *RoleDefintion* to *HumanRessource* Concept. A *TaskDefinition* is a *MethodContentElement* that defines work being performed by *RoleDefinition* instances. Like *Activity*, a *Task* is associated with input and output *WorkProducts*.

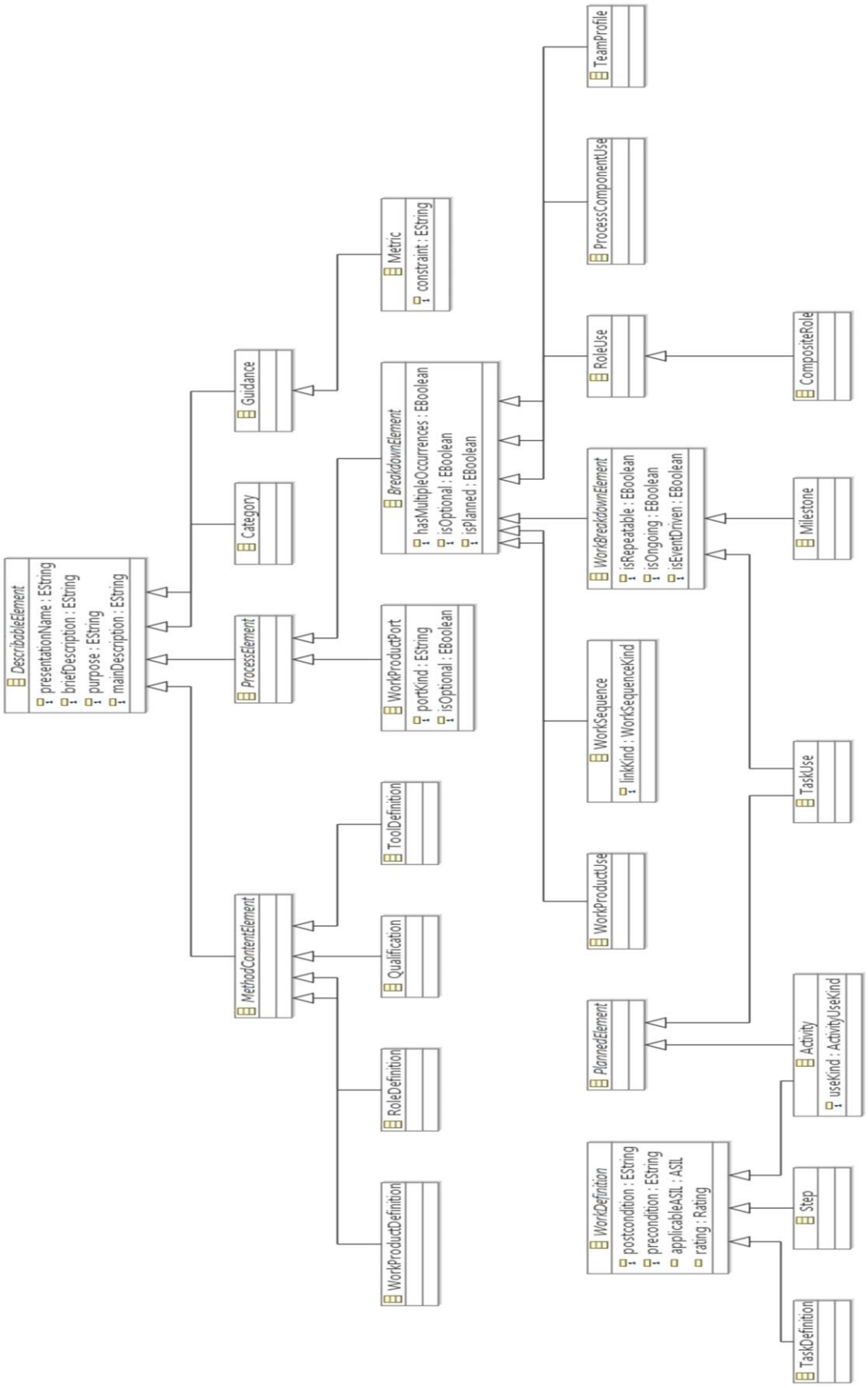


Figure 4-54. SPEM Method Content metamodel

Inputs are differentiated in mandatory versus optional inputs. There are different kinds of *Workproduct* that can have interrelations between them (see Figure 4-45).

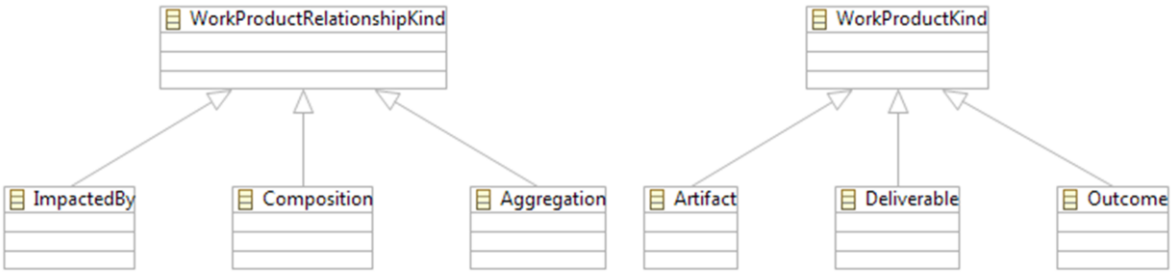


Figure 4-55. Workproduct metamodel

A *Step* is used to organize a *TaskDefinition*'s Content description into parts or subunits of work. Then, *TaskDescription* can be mapped to *Requirement* concept and *Step* too as *Requirement* can be composed of other Requirements. Following the semantic, *WorkproductDefintion* can be mapped to *Workproduct* concept. Nevertheless, it will extend this concept by adding back the attribute *isExternal*. In the same way, *Category* matches the *Category* concept as is a *DescribableElement* used to categorize, or group any number of *DescribableElements*. Because Categories are *DescribableElements* themselves, they can be used to recursively categorize other *Category* instances as well. Then *Category* also matches *Group* concept. *Guidance* is a generic element that provides additional information related to *DescribableElements*. Particular *Guidance* is classified with kinds that indicate a specific type of guidance (see Figure 4-46) and that map many of the standards concepts.

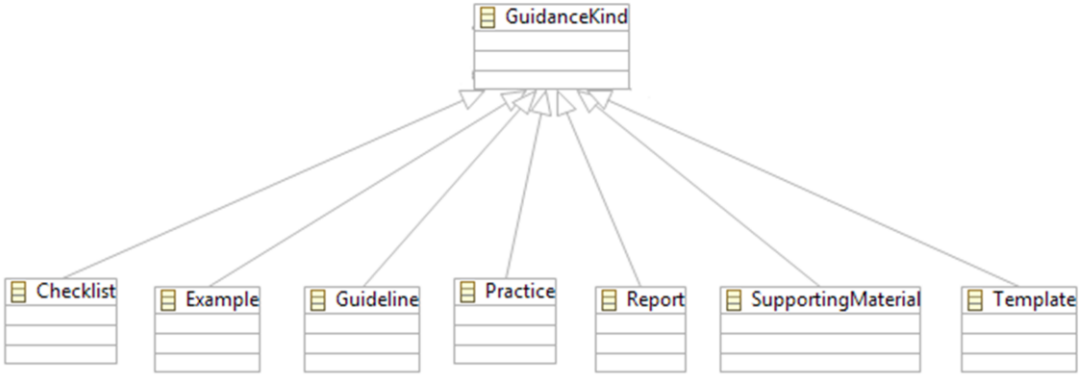


Figure 4-56. Extract of some Guidance kind types

Then, *Example* matches the similar concept, *Note* can be seen as a *Guideline*, an *Annex* can be a *Practice*, a *Report* or a *Template* according to the usage done in ISO26262, *Infrastructure* Concept can match *SupportingMaterial* as is a catch-all for other types of guidance not specifically defined elsewhere (see Table 4-13).

Table 4-13. SPEM elements and standards concepts mapping

Model element in SPEM	Standards concept
Activity	Clause
Task	Requirement
Step	Requirement
Tool	Tool
Role	HumanResource
WorkProduct	WorkProduct
Category	Category
Guideline	Note
Practice, Report, Template	Annex
Example	Example
SupportingMaterial	Infrastructure

However, other concepts are not supported by the SPEM metamodel, in particular the concepts such as *ASIL, BasePractice, etc*… To cover these concepts, it is necessary to develop some extensions to the process modeling language.

4.4.1.2 Extensions points

The first concept which needs extension points in SPEM is the *ASIL* level. Indeed, we know that each *Clause* or *Requirement* is applicable according to some ASIL levels. The second concept is the rating scale from SPICE applicable at *Requirement* and *Clause* levels. In SPEM, *WorkDefinition* is the abstract classifier that generalizes the two concepts. It must then have the propriety to specify the applicable ASIL for it. In ISO26262, there are also *Methods* tables or *Properties* tables that can be attached to *Clause* or *Requirement*. These tables are kinds of guidance elements that contain a set of properties (respectively methods) that can have three different values (*Highly Recommended, Recommended, No recommended*) depending on the ASIL to be reached. These properties (respectively methods) must be used consecutively or alternatively. The other extension point necessary is to manage the assessment features from SPICE like *Base Practice, Process Attribute* and *Generic Practice*. Indeed, each *Process* in SPICE has its parameters, which allows their capability level to be defined. The process equivalent concept in SPEM is *Activity.* This must also be extended by new elements in order to handle the preceding assessment concepts (see Figure 4-47).

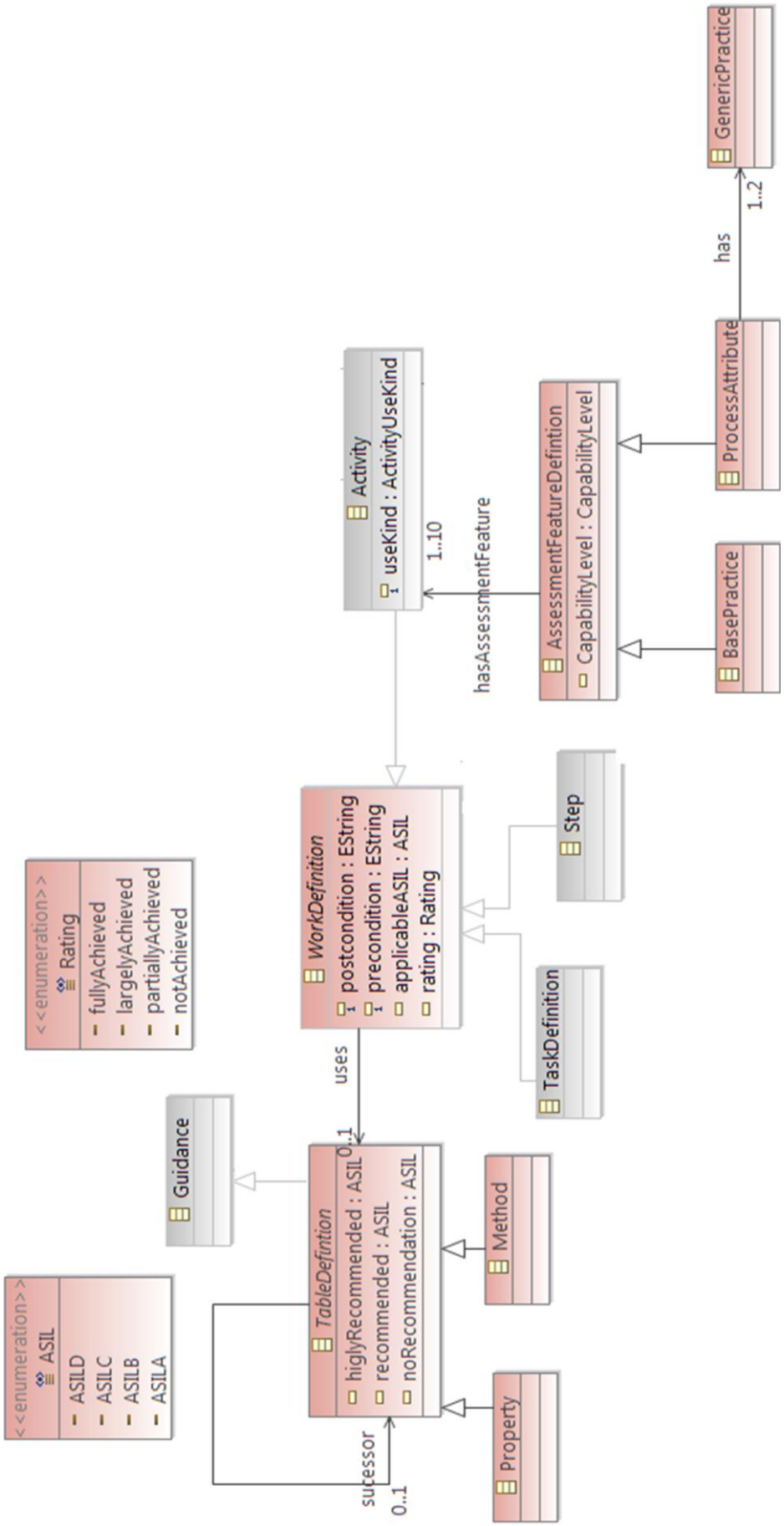


Figure 4-57. SPEM metamodel extension

We have seen that the process model is defined at the top of a product model and that the entities defined with ReMIAS profiles can be considered as output workproducts of some process activities. In SPEM, *Artifact* is the kind of workproduct that provides a description for tangible work product types. To integrate the relation between product and process models, we have seen in section 3 that *WorkproductKind* concept was specialized in the CESAR project [CES 10a] with an extension mechanism to introduce artifact types of domain models (see Figure 4-48). As the *Workproduct* can have different kinds of relationships between them, this characteristic meets the compositional aspect of artifacts. So, for example, a requirement model is an artifact that is composed of requirements (which are also artifacts) and that could be an output of the *Functional Safety requirements* activity.

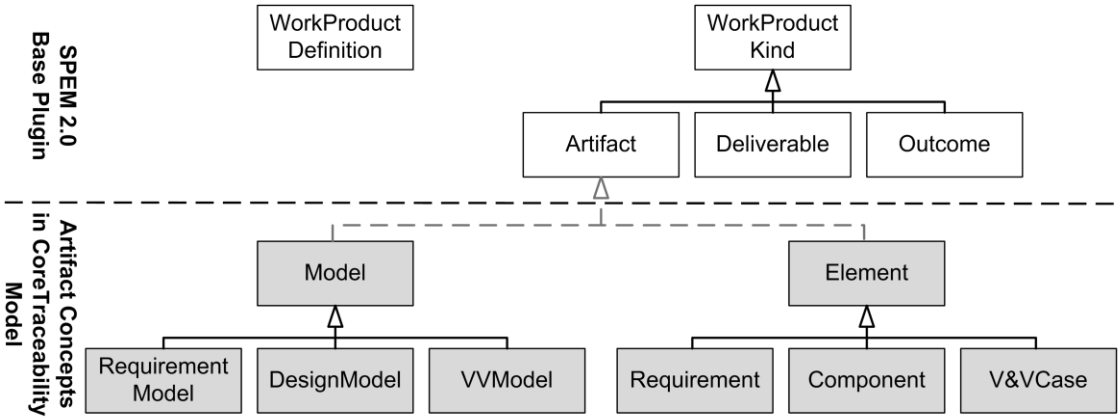


Figure 4-58. Integration of Process and Product Model Work Product Concepts

The global conceptual metamodel obtained with these extensions made to SPEM modeling language represents the metamodel that an organization desiring to be compliant with both standards must implement to obtain a generic process model. It has been implemented as Ecore metamodels in EMF.

4.4.2 Configuration of process depending on specific goals for specific projects

With the metamodel, we can define a process model in order to achieve certification for SPICE and ISO26262. Typically, the process model includes all activities, tasks, roles and workproducts as in our Excel framework. The process model is modeled as a Method Plugin in SPEM. Although documenting this is a significant workload (more than 4000 requirements, thousands of workproducts only for the ISO26262 consideration), a general process is not appropriate for all projects. Process models are usually developed to capture successful processes so that they can be reused (and improved) from one project to another. We have already used the definition of some quality factors to delimit the context of our assessment when we used the Excel framework. Here too, we must customize the process model to fit the project specific context and characteristics. Example of such characteristics is the ASIL level reached, the size and the complexity of the project, the technology involved, the engineers involved etc… Applying these characteristics to the generic process model enable project specific processes to be generated, based on the particularities of each project. The resulting processes in SPEM are named delivery processes. Nevertheless, defining a specific delivery process for each project in an ad-hoc manner is time consuming and error prone regarding the huge amount of data to manipulate. Furthermore, as the same organization can be involved in different types of project and the context and project characteristics are continuously

evolving (changing management aspects), it becomes quickly difficult and expensive to adapt the process to different project situations. As done with the algorithms implemented in the Excel assessment framework in first chapter of this section, it is therefore necessary to have a way of generating the delivery processes automatically.

4.4.2.1 Specialization of process model based on context characterizations

Each development project needs its own specific process in order to create an effective and efficient product. This specific process contains a selecting set of process actions that contribute to product quality. Then, each project imposes the definition of the process that best fits it: it is the delivery process principle. A Delivery Process is a special process describing a complete and integrated approach for performing a specific project type. In order to let the process be most effective and efficient, this set of process actions should be selected based on practices, characteristics, techniques and constraints requirements aligned with the business context. In order to facilitate the selection of process actions, an overview should be also done on project characterizations. These characteristics indicate that certain artifacts should or should not be included as part of the adapted process, according to certain context values like for example, among others:

- The nature of the item under development. Indeed, given that if it is a new development, the hazard analysis and risk assessment must be performed and if not (Reuse without modification, Reuse with modifications) it is an impact analysis which must be performed and some change management requirements must be applied.
- The ASIL of the item because the number of tasks to be performed increases depending on the severity level.
- The recommendation level of methods and properties desired.

These characteristics should be used some attribute values in the source process model element to determine the artifacts to be considered.
The SPEM variability mechanisms must be also be used as the process model is defined using SPEM with the sharing of common features and some variability parts. Each *MethodContentElement* and the *Activity* metaclasses are variability elements in SPEM. A variability element is an element that can be modified or extended by other variability elements according to a variability type *(extends, replaces, contributes, extendsreplace*). For example, a method element can be linked to one of many variants. Additionally, an element can be considered as optional or not according to the *isOptional* attribute or *isExternal* attribute if it cannot consider them (see Figure 4-49).

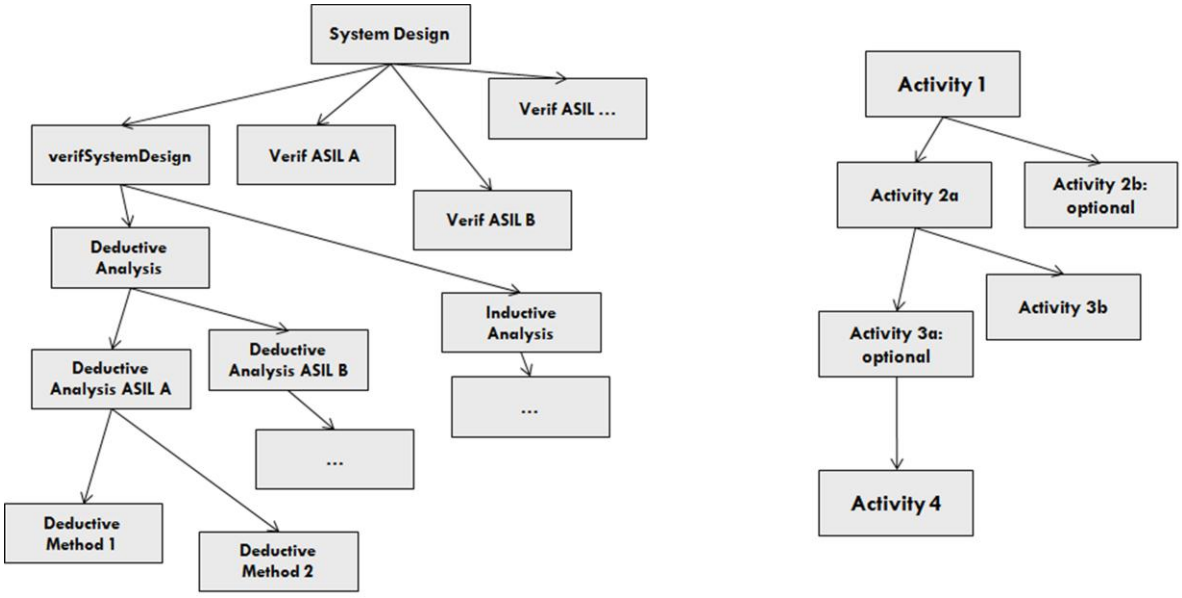


Figure 4-59. Example of generic process views

The different context characteristics can be composed incrementally: an element can be optional and not fit the desired ASIL for example.
Once a set of relevant project constraints has been selected, the delivery process specific to a project can be constructed. Such a model represents the activities, workproducts, techniques, roles and guidance of a specific development project, configuring a process model with variabilities resolved and that guarantees the required product and project qualities.

4.4.2.2 Implementation

The Process model is implemented as an instance of the SPEM metamodel into EPF. We are currently implementing a plugin to incorporate the extension points to be compliant with the standards common metamodel together with the process configuration approach. To use the configuration functionality, it will be necessary to enter manually through an interface the particular attributes of the project context as a configuration of the delivery process that we want to generate automatically. The set of all context attributes that could be configurable are not yet fixed definitively. The plugin will consider the variation points according to these attributes in the process model (with its variabilities parts) to automatically produce an adapted process as the delivery process. All and only the required process elements which fit the configuration options will be present, thus the delivery process obtained would be de facto the most efficient and effective for the development project considered.
The sequence of steps in a delivery process model should be included in the project planning for project running purposes.

4.4.3 Planning and monitoring

Process delivery described with SPEM can be enacted in different ways. The most common way is to map the processes into Project Plans, enacting these with project planning and enactment systems and then running these representations with the help of an execution language. Project running has two purposes for product development:

- Monitor current status of the product, and as such the conformance of the product to the requirements
- Observe the relationship between a process action and its impact on product quality

We have seen that measurement is a prerequisite to respond to these two purposes. Measurement program engineering is described in the literature as *"the design and implementation of a set of process, product and resource metrics, to achieve predefined objectives within an organization"* [TRI 01]. It is the preliminary step to process improvement. First of all, measurement provides information on the conformity of the product with the product quality specification. Secondly, measurement enables the evaluation of the extent to which process actions result in the intended effects. A measurement program carried out in practice contains three main phases:

- Definition of the measurement metrics
- Data collection from the measurement data according to the plan
- Interpretation of the measurement data

SPEM does not integrate the endeavor layer where this measurement and check is possible when a process is running, nor the measure definition phase. In [CES 10a], these phases are covered in a requirement based project monitoring goal. The environment is implemented through the Permeter tool.

4.4.3.1 Definition of measures

The first phase is handling as an extension of SPEM with generic measurement concepts compared to the existing metric concept in SPEM. Permeter tool supports the definition of progress metrics for different tasks in a development process as well as the definition of milestone goals for general milestone plans. The definition of the metrics is compliant with measurement standards [ISO 07d, ISO 07b, INC 98, ISO 10, MED 09], as such these inspired SPICE and ISO26262 and used the metamodel based on the ontology [GAR 06a, CES 10a] of key concepts of a process measurement (see Figure 4-50). Then, it is easy to rely on our assessment framework at the time they are defined.

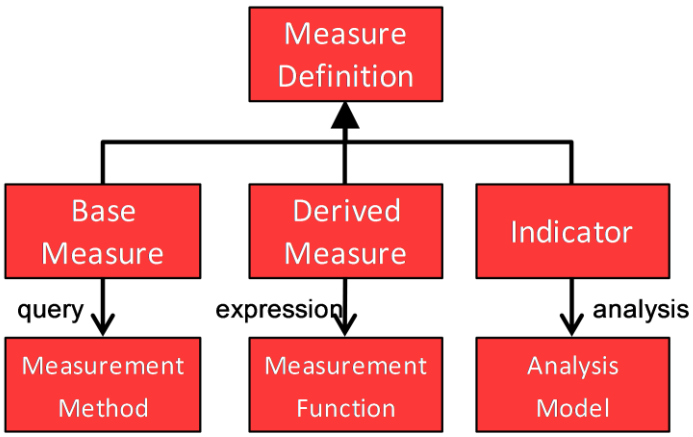


Figure 4-60. Measure Defintion metamodel

Measures are part of a content package along with the definition of workproducts, tasks, roles and guidance. A measure is linked to one or more workproducts and can be linked to a task. An *Indicator* provides the possibility of using more complex assessment models to assess the progress of certain

task. *Measure* and *Indicator* are defined as OCL queries that are interpreted as a runtime. Furthermore, the tool offers the capability of defining milestones and also the definition of specific assessable goals that need to be fulfilled at the milestone (see Figure 4-51).

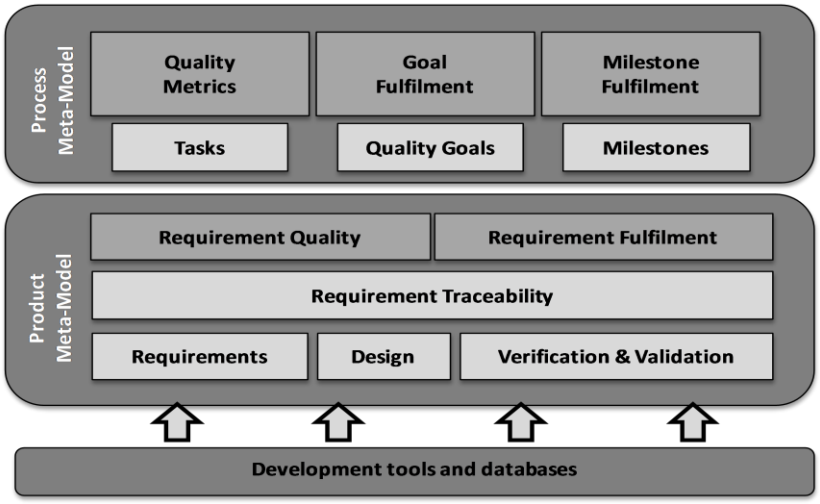


Figure 4-61. High-level overview on assessable elements and their dependencies

Once the *Method Content* including its measurement procedures is defined, it is possible to compose whole development processes as a delivery process. At this stage, the configuration approach that was conceptually explained above can be used.

4.4.3.2 Collection and interpretation of the measurement data according to the plan

For the second phase, the tool offers the functionality to create project plans based on one or more defined processes in order to apply a process-based measurement definition in a project. The main integral part between the project plan and the actual development is the concrete instance of the system or component the plan is defined for. For example, it is possible to create a standards compliant project plan for a selected system or sub-system according a given configured delivery process. Permeter itself offers no specific functionality for project planning (e.g. persons in charge, deadlines, etc.). For this purpose, exporting of the created project plan template to MS Project is supported. The project plan, linked to the process and its measurement definition can be exported to MS Project so that it is possible to assign durations, milestone deadlines etc… The content of the different work products is calculated dynamically based on the concrete component and the semantics of the workproduct types. Next, the tool also offers the possibility of re-importing the refined project plans with all the information necessary to perform progress or trend analyses of the different activities in the plan.

The third phase is the quality and progress assessment. Based on the defined project plan and its measurement definition, Permeter allows an assessment against current development data. To summarize, the following assessment functions are available over time:

- Project overview through a Gantt chart that shows all activities, tasks and milestones of the project
- Progress of tasks and milestone achievement
- Progress over time for metrics
- Quality status of different workproducts from the system model (artifact outputs from ReMIAS model)

- Requirement quality status for requirements and requirement sets
- Traceability from process work products to concrete development artifacts

4.4.4 Summary

The last expansion necessary to reach the goal of the thesis was the modeling and measuring of product and process depending on the product quality specification. By considering the extensions in SPEM packages, the method for process configuration depending on specific goals for specific projects and Permeter extensions, the objective is achieved. Modeling of the process is ensured through the extended metamodel defined. The metamodel includes all concepts of the standards common metamodel discussed in the merging of SPICE and ISO26262 work. The integration of the process and product properties is managed thanks to the workproduct extension elements from [CES 10a]. The instantiation of the metamodel provides a generic organizational process compliant with automotive standards in the center of our preoccupations. From this generic process model, we define an approach to tailor it into specific processes considering the context and characterization of specific projects in order to create effective and efficient products. Since this tailoring process is intended to be automatic, it is expected to reduce the tuning time and cost, and also to allow fewer adaptation errors as only process elements that are required for the particular project context are considered. In addition, high quality can be expected, because the process is adjusted to the goal in each particular project context. The assessment goal is tackled with the use of the EPF tool and the Permeter tool provided by [CES 10a]. A drawback is that OCL query writing for the measures definition is too complex for a typical user. To be able to define measures requires in-depth knowledge of the language. We are currently implementing a graphical interface to make this task user friendly and accessible to everyone.

We note also that all these innovations cover the three abstraction layers in the process modeling domain, namely the metamodel level, the model level and the endeavor level [IS0 07a].

4.5 Conclusion

The PhD thesis work pursues the objective of using the requirements in a modeling environment to define systems safe and compliant with automotive standards, following a hierarchical modeling process specifically covering the system engineering activities. From the state of the art and practices, we have found three major areas which needed expansions in the current state of the art and practice. In this section, we have presented our contributions for each of them.

A study of ISO26262 and SPICE standards was performed to extract safety concepts and process rules relevant to the automotive field; both are aimed at standardizing the development of safety-critical automotive E/E systems to manage their increasing complexity. Among the most important, the "*Specification and management of safety requirements*" chapter 6 of ISO26262-8 and the first HIS Automotive SPICE Engineering Process Groups (called ENG): *requirements elicitation* (ENG1), *system requirements analysis* (ENG2), *system architecture design process* (ENG3) were analyzed. A modeling framework seemed to be best suited to formalize and exploit these elements. From this work, we then proposed an extended metamodel to describe the two standards in a common framework without altering their respective contents.

Because product quality is also affected during the certification, we have defined the REMIAS profile. It is a UML-based approach that provides advanced techniques and notations supporting the full requirements life cycle. The link with architectural modeling is also necessary to complete the system

engineering processes. These parts were integrated in the profile together with their traceability aspects. To go further into the traceability management, the proposal is completed by our plugin Office2Papyrus. This can export and import a requirement repository into UML-based modeling tool from MS requirements specification documents, avoiding redundant working for requirement engineers.

Considering the purpose of the assessment, our contribution proposes a generic methodology in an acceptable certification perspective where an HIS assessment and a functional safety audit is simultaneously performed. The main commitment is the definition of a framework where we apply the SPICE assessment method to the common metamodel defined above. Thus, system engineers can evaluate the adequacy of their systems and of their processes of development to standards and get an idea of the maturity level they have achieved. To always follow the model-driven approach, the solution has been implemented in a modeling process-based language.

This last improvement from the state of the art brings innovations in modeling, customization depending on a specific context, planning and control of a development project, focusing both on the end-product quality approach and the process development approach.

The validation of the research works is tested in an automotive application in the next section.

5

Industrial application of the approach

5 Evaluation of the approach

5.1 Introduction

This section presents the evaluation of the model-based methodology for automotive products development described in section 4. Central to this evaluation is the thesis hypothesis, previously presented in section 1, which states to **"Define a synthesis between product and process development following a model-based paradigm and to provide integration of standards reference, formulated in terms of requirements."**
The terms *model-based paradigm*, *standards reference*, *requirements, quality* provide the key concepts against which the research presented in this thesis is evaluated. The evaluation examines if the contributions create a flexible environment supporting successive phases of the development life-cycle, from requirements management up to system design, particularly in an industrial context. It also examines if the approach correctly manages the requirements formalism (and their traceability) and the architecture aspects as they are fundamental to prove reliability against expectations of automotive standards. Further, the evaluation checks if the approach successfully addresses the process aspects. Finally, the section examines if the relations between products and processes artifacts are identified and modeled in such a way that interoperability and interrelationship are effective. As safety aspects are significant interest for automotive industry, the ISO26262 standard should be used as guidance. The efficiency and the quality of the approach will be measured regarding some goals such as time consumed, integration of safety artifacts, tool support, interoperability and limitation of manual activities. In this section, the evaluation is introduced following by a detailed examination of the research outcomes and contributions. Part of this evaluation was published in [10].

5.2 Forms of evaluation of applied research

Evaluation is an important foundation for the methodological justification of research contribution. The research evaluation appeals to different forms of evaluation, ranging from peer review, tool support to a pilot industrial application.
Peer review was effective with respect to examining the technical consistency of each contribution.
The research outcomes of this thesis were presented to, and reviewed by, various researchers in academia and by systems, quality and software engineers in industry. The feedback was useful to revise and improve the contribution. Finally, based on this research, a number of papers were published and presented at peer-reviewed international conferences and workshops.
Tool support offered a means for checking the consistency and correct formulation of the profiles, metamodels and models defined in this thesis, in addition to demonstrating certain elements of usability and manageability. Most of them were implemented in Eclipse using the Papyrus MDT editor. The tool proposes a validation functionality which allows the validity of the elements to be checked with respect to the UML2 metamodel. The EPF and Permeter tools were used to implement process and measurement parts of the contribution, for checking process compliance, for example. Microsoft Excel was also used to validate usability of the certification merging approach. Furthermore, the tools also offered a means for focused and effective peer-review.

The pilot industrial application, in addition to showing technical consistency and efficacity, demonstrated the feasibility of using the research outcomes of this thesis in an industrial context. It was carried out jointly with systems and quality engineers. Strong evidence of feasibility and practicality can be proven due to the degree of independence between the author and engineers involved in the project. It is introduced in the next chapter.

5.3 Industrial pilot application

In this section, a pilot industrial application is presented from the automotive sector. Indeed, to validate the outcomes of the research works, we have established an empirical system that it was built incrementally, i.e. we enriched it as we progressed with the work. This industrial pilot application is a real embedded automotive system project currently in development named BSG_E (In French, Boîtier de Servitude Générique – Electronique). Due to the commercial sensitivity of different aspects of the system, some system details have been removed while others have been abstracted in the evaluation.

5.3.1 Overview

The Pilot Application is an embedded electronic body controller derived from a dedicated product line. This product line is composed of several body controllers, each of them managing certain functionalities in a car or a trailer. They can perform either gateway tasks or functional actions (lights, horn, windshield wiper, starter, heating or power management, for example) and communicate either with other controllers or directly with actuators. The selected body controller is the BSG_E. It is mainly connected to an Intelligent body controller (BSI), which send the requests through a CAN Low Speed protocol (CAN LS), and to the battery, which supplies power. As the main function, the BSG_E product covers the piloting of front fog lights of the vehicle (these lights could also be used as cornering lights) through power components. This module also manages the following functions:

- The electrical protection of downstream wires (not loads);
- Ensuring the internal and output diagnostic;
- Ensuring the dialogue with the main car ECU (BSI) by a CAN low speed communication network;
- Managing and storing local defects.

The Figure 5-1 gives an overview of the system boundaries.
As there is only a small number of system functionalities, the evaluation can successfully focus on the results of the research work. It includes areas such as requirements engineering, architecture modeling, multi-formalism modeling, interoperability, process and tool support. The current state of practices regarding these areas will be presented first.

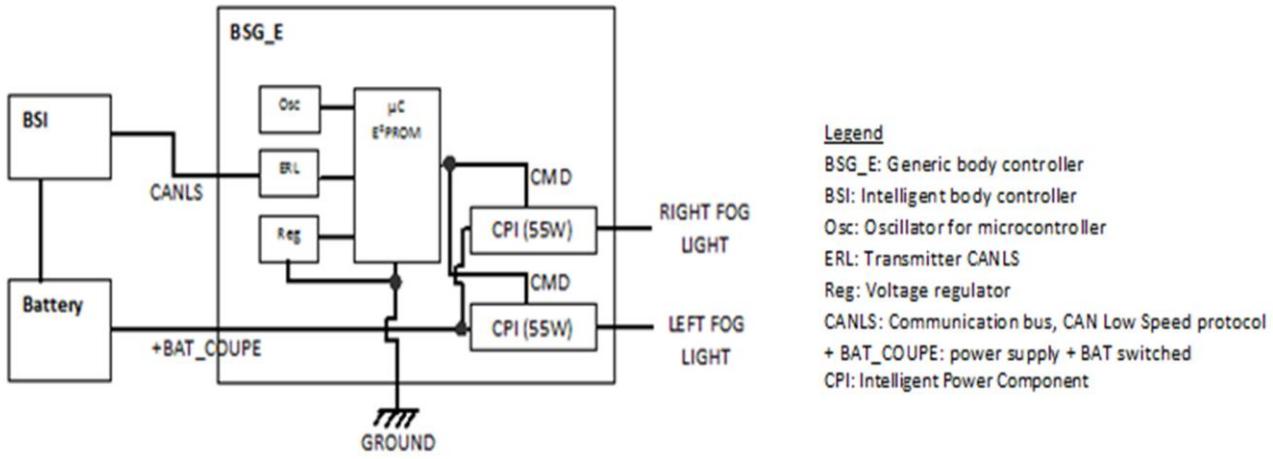


Figure 5-62. Overview of BSG-E system boundaries

5.3.2 Problem statement in current practices

Modern systems, specifically embedded systems in the automotive domain, are gaining in overall interactive complexity and constraints coupled with the pressures of tight schedules. These embedded systems are defined according to complex processes combining different formalisms following the main phases of a typical V-Model cycle, from the initial stages of specification to code product. As our goal is to provide a modeling environment for the system engineering, we focus only on the first phases of the cycle, namely the requirements management and the architecture definition. The following Figure 5-2 shows the workflow with the associated tools, currently used for the product development.

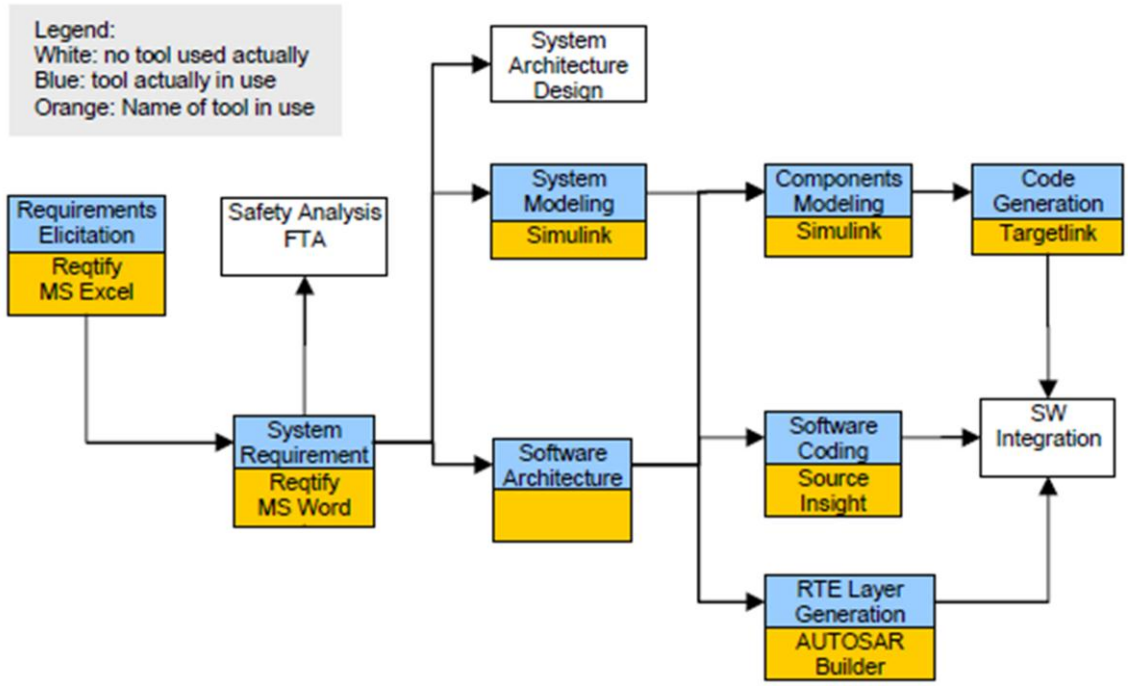


Figure 5-63. Workflow with the associated tools, currently used for the product development

Many gaps are detected in this process. The requirements specifications are done in majority through general office automation tools (such as MS Excel™ or MS Word™). The traceability of the requirements is assumed by the Reqtify tool which gives the coverage of the implementation requirements. Although the tool helps to ensure the management of the traceability between requirements from textual documents, gaps remain because this traceability information is not ensured until the design elements. Other gaps concern the architectures that are not managed by tools. Identically the Safety and Fault Tree Analysis are done manually. Concerning functional safety, a specific approach is used but it is inadequately integrated in the process development and compliant with the expectations of the ISO26262.
The expectations are intended to deal with these problems, in particular through defining a methodology which can be used by an integrated tool chain where automation and interoperability are as efficient as possible. This challenge is to attempt taking into account safety and process considerations from the Automotive SPICE referential and the ISO26262 standard.
The following topics require improvement with respect to processes:

- Tool support for the requirements analysis and system architecture definition
- Static analysis for models (requirements, architecture)
- Automation of links between different phases for limitation of “from scratch” activities
- Unique platform for requirements and system architecture manipulation
- Automation of the process for limitation of manual activities
- Integration of ISO26262 methods and tasks in the process
- Consideration of SPICE methodology

The following topics require improvement with respect to tools:

- Communication between tools thus limiting manual work
- Means to trace requirements into all files
- Support for architecture design
- Support for converting textual requirements into design models
- Enhance models exploration
- An overview of the complete project can be given at any time
- Gives usable criteria for quality evaluation
- Model the process used
- Complete static analysis for models
- Means to validate architecture

These expectations are reflected in the PhD thesis objectives. The main objective of this pilot industrial application is to examine whether our model-based approach can be used to make the improvements.

5.3.3 Methodology, notations and tools support

To fulfill the previously mentioned gaps identified in the current practices, a representative scenario was chosen, reflecting an important subset of the whole activity in order to assess the applicability of solutions in terms of methods and tools regarding our objectives. The following Figure 5-3 describes the workflow of the development which is adopted.

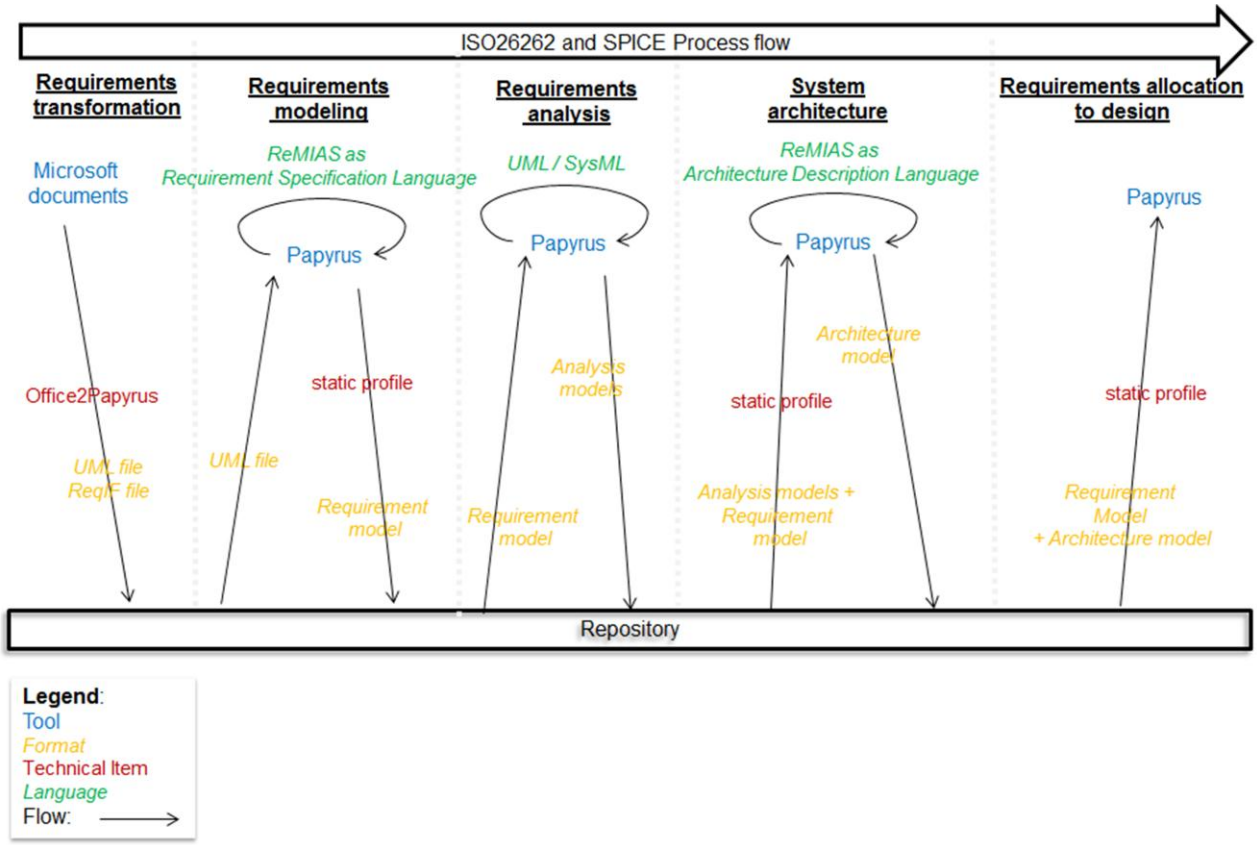


Figure 5-64. Workflow of development adopted

The methodology relies on some model-driven engineering methods and dedicated tools like Papyrus MDT. We mainly focused on the following items:

- For the requirements part, the tracking of requirements through the conversion of textual requirements into design models and their traceability based on graphical formalism.
- For the architecture part, the system architecture description based on the graphical representation of the requirements.
- From the interoperability point of view, the automation of links between different phases of the development process and especially, the bridges between tools.
- The connection of the third point with the automotive referential SPICE and ISO26262.

The tracking of requirements will be implemented throughout the development process. Requirements management is a very important challenge in the automotive industry. To assist the designer in managing requirements during the system development, a good formalization helps to avoid misinterpretation. Checking of traceability consistency checking results to answer difficult questions early, which reduces rework through identifying issues and detail's needs earlier. We start from a representative set of existing requirements in natural language, which is managed firstly via a requirements specification document, before being converted to a requirements model with Papyrus MDT.

Next steps define the system architecture of the product. Component based design is also a key issue to manage complexity and costs. The methodology defines different models used at system

level. The goal is to have a complete product description in a system model with incremental architecture definition. System architecture is also tackled using Papyrus MDT tool.
Automation of links between the requirements management and the architecture definition phases is mandatory to gain major benefits of the process and limit “from scratch” and manual activities. Here, an innovating technique is proposed to better focus on the engineering process in a more integrated way. Our Office2Papyrus plugin, a lightweight data integration layer, is used to ensure traceability between textual documents and model entities stored in a model repository. With this tool, the requirements are directly included in a modeling process, so that they can be connected to the developed design for an easier bi-directional traceability.
To complete the approach, an accurate automotive standards assessment is defined at process level. The aim is to carry out each phase following the recommendations of the SPICE and ISO26262 standards. It includes safety consideration and software assessment at requirement and system design levels.
All these aspects will be developed and evaluated in our body controller application.

5.4 Experiences with applying our approach

5.4.1 Experiences with applying our approach at product level

5.4.1.1 Importing requirements in modeling environment phase

A set of requirements in natural language from customer specification documents has been selected for evaluation. Importing requirements from Word or Excel into the model environment is performed with the Office2Papyrus plugin. The generation creates two files: an XML file that we have tested as compliant with the ReqIF XML schema provided by the OMG; and a UML file. The UML file is compliant with the structure of the document (Word and Excel, respectively) following the template rules below:

- In Word, a chapter regroups a set of requirements. A chapter can include other chapters, iteratively. It is represented in UML by a package and a package hierarchy. In Excel, this hierarchy structure is specified by sheet.
- In both format documents, a requirement is in a row (in a row table in Word) with its attributes in columns. An XML configuration file allows each column header to specify the corresponding model element attribute. In Word, the attributes can also include pictures, tables, embedded file etc., which are stored in the project workspace in repositories in their original format (.JPEG, .DOC, .XLS, .VSD, .PDF, etc…) except for the embedded tables in requirements which are stored like an html page.

In our files, we have as many columns as requirement attributes corresponding like it is defined in the ReMIAS profile. In the model explorer view of Papyrus, we can navigate through the model elements (see Figure 5-4).

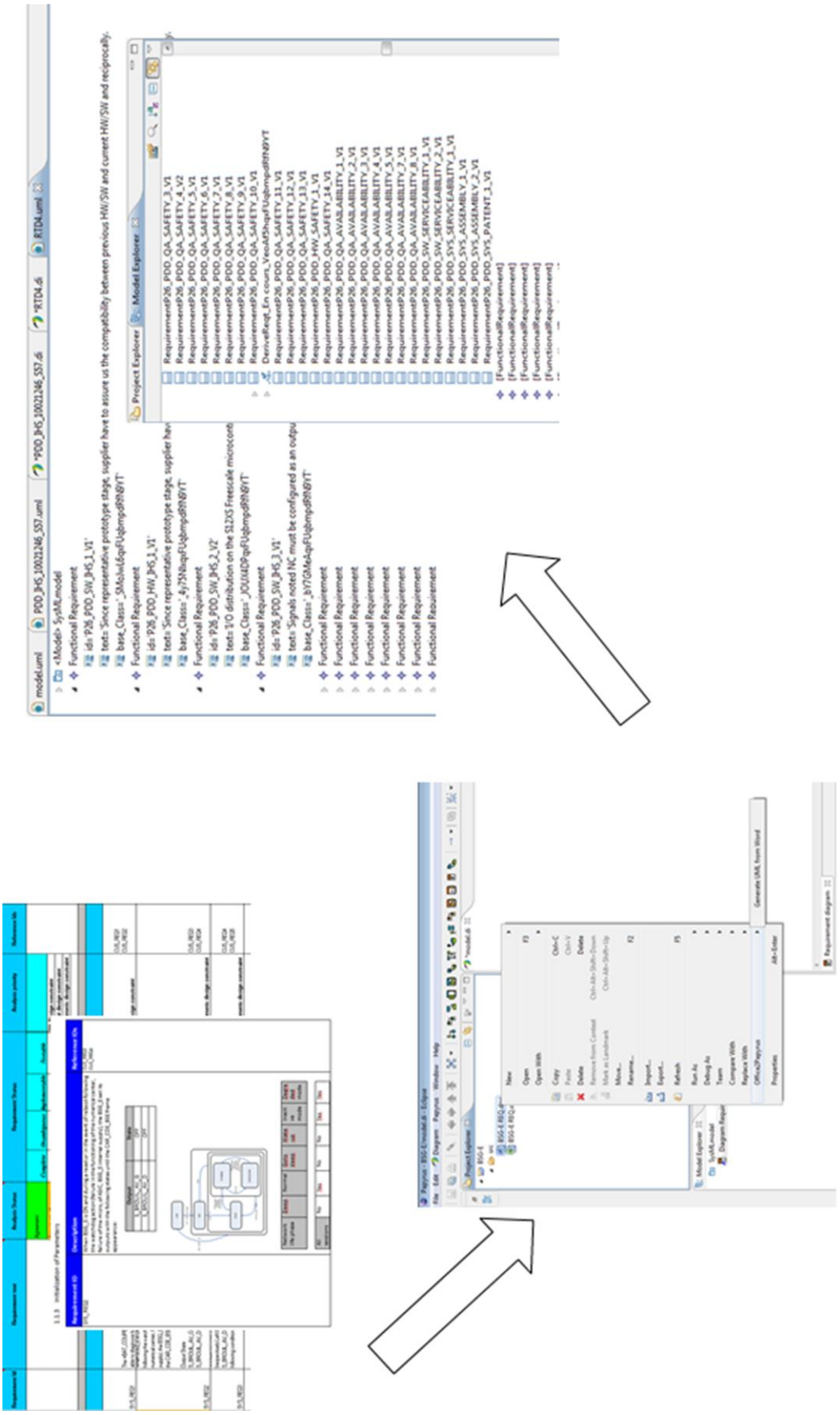


Figure 5-65. Requirements import in modeling environment

The integration between Office2Papyrus and Papyrus MDT is a huge advantage as Microsoft documents are widely used in the industry. The next step is to convert the requirements into a graphical form.

5.4.1.2 Conversion of requirements into graphical ReMIAS representation

The initial action for this step was to generate the UML file corresponding to a customer specification document in textual format. From the UML file, when it is valid, the Papyrus MDT allows the corresponding graphical view to be obtained: the diagram file is generated automatically when a Papyrus model is defined from this UML file. Then, with a drag and drop, the elements can be positioned in the graphical editor. This step can be performed in one action by selecting all the elements. By default, all requirements are stereotyped *FunctionalRequirement*.

5.4.1.3 Requirements specification and analysis phase

Further analysis allows the requirements specification to be refined with their real type. For instance, with respect to identifying which requirements related to safety: *FunctionalSafetyRequirement, TechnicalSafetyRequirement, etc.…* determining the ASIL level of the first requirements level is inherited from the safety goal and is out of our scope. The implemented rules in the profile help to define the other traceability information between requirements in order to be compliant with ISO26262. Thus, between requirements of the same type only *refine* and *decompose* links are possible. A derive link is possible between requirements of different type at different abstraction levels when they are related to safety as stated in ISO26262 (see Figure 5-5). Papyrus MDT allows the model or a subtree of the model to be validated to check if the current development is well-defined.

The effort made here improves the verification of some quality criteria of requirements, such as completeness, inconsistency, ambiguity, etc..... The requirement specification and analysis step is carried further with other SysML diagrams. In our evaluation, the main purpose and the boundaries of the system have been defined by collecting a set of use cases. Based on these, a set of activity and sequence diagrams has been created to describe detailed operational scenarios for each use case and theirs interactions with actors and the environment. The activity diagram was suitable for general functional requirements and the sequence diagram was suitable for handling timing properties like duration, time response functions, etc… (See Figure 5-6).
The main advantage of this analysis is the contribution to the high-level system architecture definition.

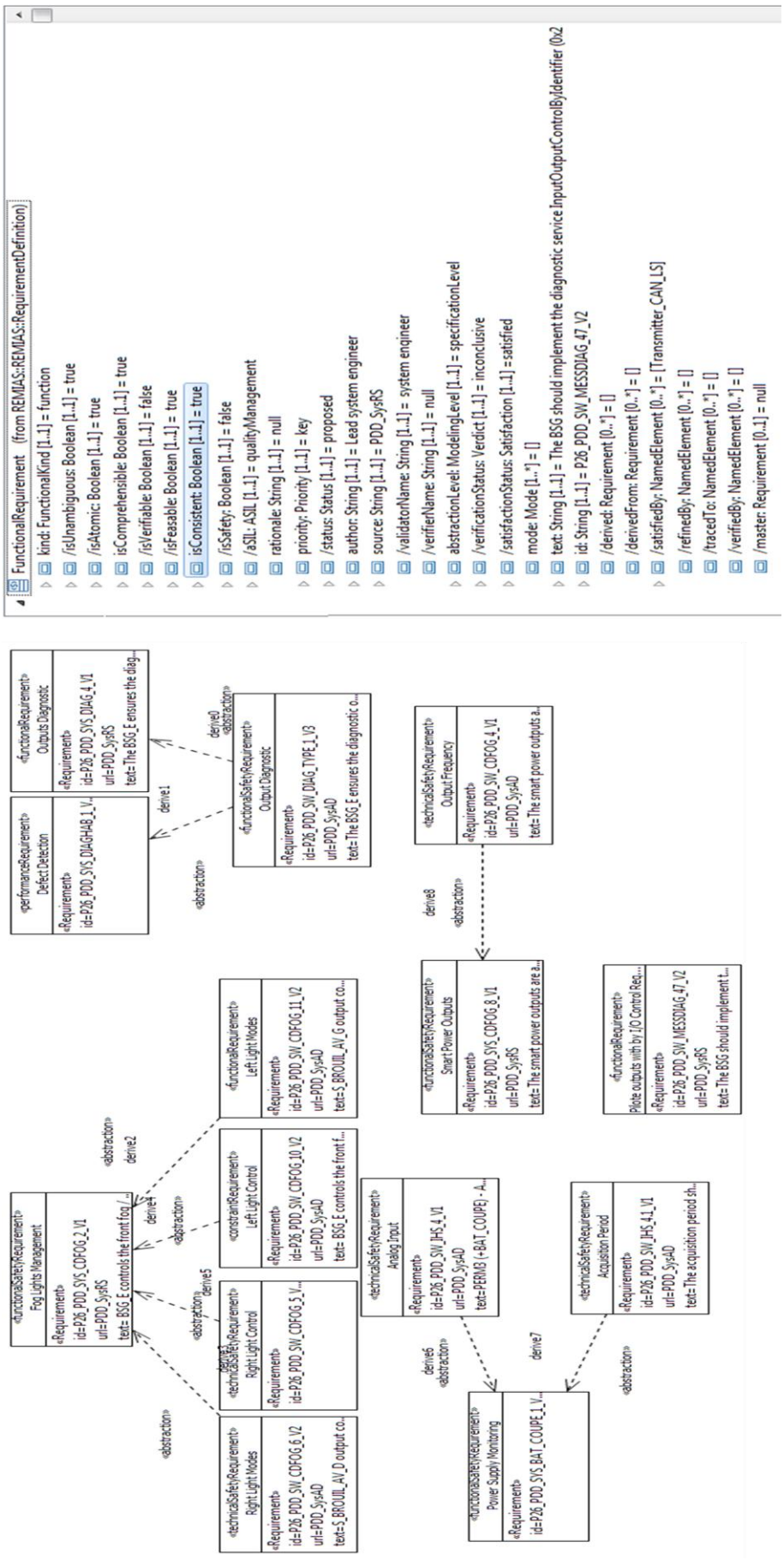


Figure 5-66. Requirement specification with ReMIAS profile applied

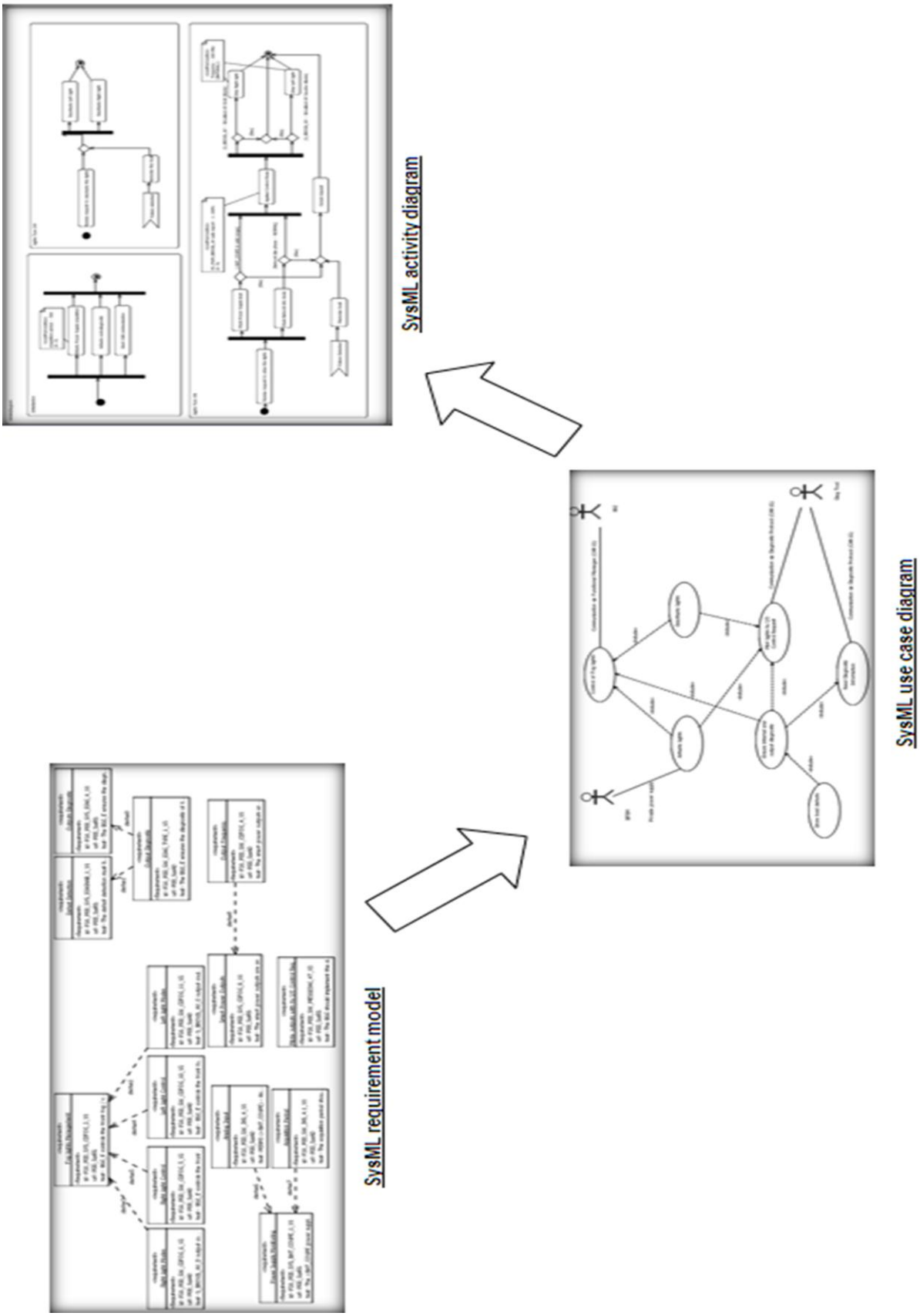


Figure 5-67. Diagrams for requirement analysis

5.4.1.4 System architecture phase

For the system architecture definition with ReMIAS, we refer to East-ADL2 architecture modeling. The primary feature used is the capability to structure a model into different abstraction levels. All these levels describe the same system, but from different viewpoints. We define the architecture in two steps which are an analysis level where we define a high level architecture and a design level where we detail the first architecture. After the requirement analysis phase, we have already identified the main functions or blocks of the system. We can then associate them with the right connectors to give them unity. This corresponds to the high level architecture at analysis level. At the design Level, two different views are proposed to separate the competency concerns: the Functional Design Architecture and the Hardware Design Architecture. In the Functional Design Architecture, the software (denoted SW) part is detailed from the Analysis Level. The goal is to detail the architecture into the smallest components, in such a way that we can attach one behavior to one component. The connectors between components and flows between functions are also refined and perfected. The Hardware Design Level represents the physical architecture of the system. A global Design Level view allows allocation to each Hardware (denoted HW) component, the SW components (one or many) that it realizes.

We have also defined an environment model to show the flows/information exchanged between our modeled system and other vehicle systems with which it interacts (battery, external environment for outputs) (see Figure 5-7). An intermediate view was defined to specify the functional interfaces between the different parts (hardware, software, environment).

5.4.1.5 Requirements allocation to design elements phase

At each abstraction level, and for each component, we associate the requirements (one or many) which are satisfied using the *satisfy* traceability link (see Figure 5-8). This information is automatically added in the requirement properties and component properties when the traceability link is graphically created. This allows a better visibility of the traceability. This information is supplemented with the ability to specify the different operating modes and system states in which the components meet the requirements. With the profile, the traceability can be performed by abstraction level. Then, the *FunctionalSafetyRequirement* can be *satisfied* only by elements at analysis level and the *TechnicalSafetyRequirement* by elements at design level. Other non safety requirements can be satisfied by any architecture elements.

After the modeling phase, textual documents (Word and Excel) from the model can be regenerated. Only the requirements (with updated data) that correspond to attributes in the template document text are exported (not the design elements).

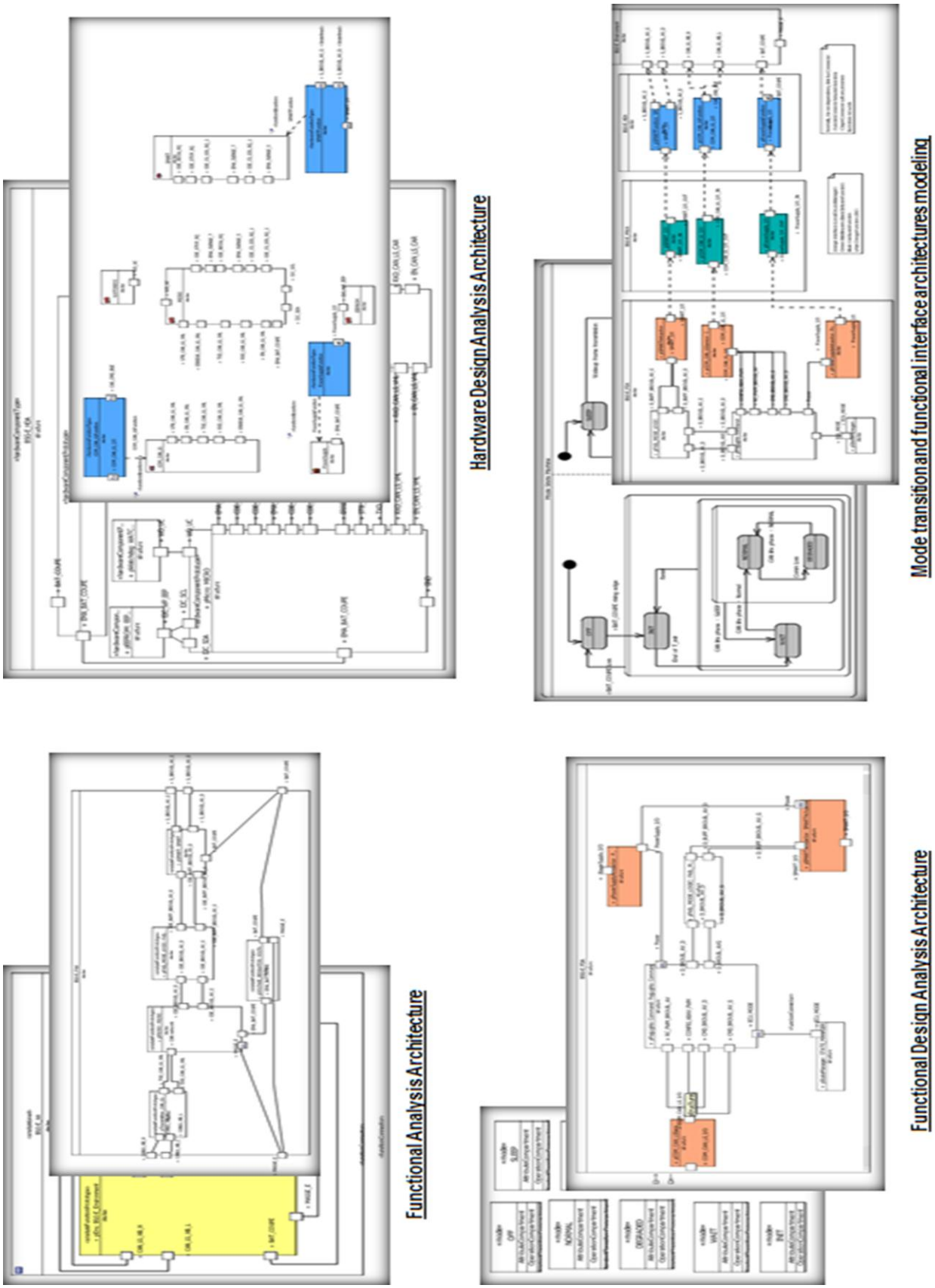


Figure 5-68. Different architecture views with ReMIAS

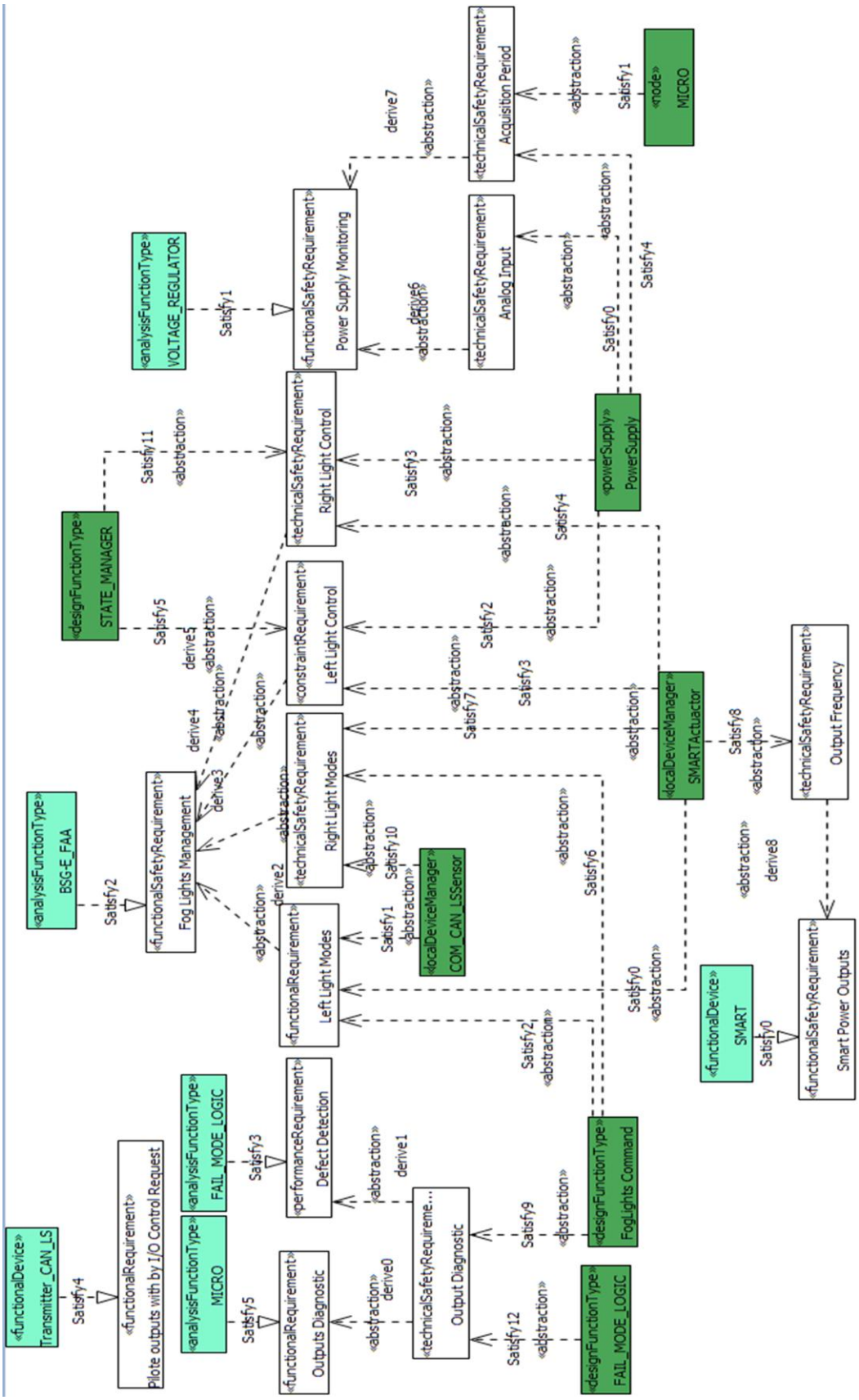


Figure 5-69. Requirements allocation to design elements

5.4.2 Experiences with applying our approach at process oriented level

5.4.2.1 Product development phase

We supplemented the proposed workflow development with a specific automotive point of view that integrates the generic methodology in an acceptable certification perspective with the SPICE and ISO26262 recommendations. We remind the reader that in this thesis, we rely on HIS Automotive SPICE scope when we refer to SPICE. Furthermore, we are only interested in the system engineering activities, namely requirement management and system architecture definition. These activities are covered specifically by ENG2: *System requirements analysis*, and ENG3: *System architectural design* processes in SPICE. In ISO26262, we are interested in the last clause of part 3: concept phase, and the first clauses in Part 4: product development at the system level. This involves the *functional safety concept*, the *specification of technical safety requirements* and *system design*, given that the *initiation of product development at the system level* (first clause of part 4) requirements are of management concern. The hazard identification and risk assessment are outside the scope of this evaluation as this is a current task performed at customer level.

We work directly from the customer requirements documents. Although the ReMIAS profile implements the verification and validation part it, we do not treat it in the evaluation.

Concerning SPICE then, the purpose of the system requirements analysis process is *"to transform the customer requirements into a set of desired system technical requirements that will guide the design of the system*" [AUT 10b]. The standard defines the architectural design as a *"process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of the system*" [AUT 10b]. It must also identify which system requirements are to be allocated to which elements of the system. Both points are relevant to the topic of our contribution (without capability level consideration). The requirements formalization using ReMIAS with SysML diagrams (requirement, use case, activity, sequence diagrams) allows the standard expectations to be met with regards to ENG.2. The ENG.3 part is also completely covered if we consider the different views defined through our modeling with the architecture part.

All these benefits are also compatible with the ISO26262 standard. Indeed, concerning this latter, ReMIAS handles requirements and architecture development. Parts of the chapter 8 of ISO26262 *specification and management of safety requirements*, and requirements decomposition with respect to ASIL tailoring, are considered. The implicit process inferred by the standard is also complied with through the traceability management, among others.

5.4.2.2 Assessment phase

The SPICE assessment is a well-established practice in industry, particularly in the automotive industry. The challenge was to assess the functional safety from the ISO26262 deployment and integrate it with this process assessment standard at the lowest cost and with minimal effort. In [ISO 11], a statement suggested that *"if a SPICE assessment is performed, then this SPICE assessment and a functional safety audit can be simultaneously performed, as there is sufficient commonality in the content that can help to avoid duplication of work or process between both"*. To obtain these coordinated processes, we have developed an assessment framework that meets this purpose as it addresses the two standards at the same time. It is based on the assessment method usually used for SPICE with rating and capability level evaluation. Throughout the pilot application development,

the framework has been a great help as a planning synchronization tool. By setting the parameters and constraints of the pilot application, we have been able to perform the following tasks without having to read the standard each time:

- Identify the different tasks to accomplish and then discard those that do not fit into our context;
- Identify the workproducts to produce and their content, with the optional parts if required;
- Provide guidance Guide for the recommended tools and methods to be used;
- Identify who is responsible for each task and each process;
- Identify the current progress in the project development regarding the tasks already accomplished and tasks which have not been accomplished. This can help in corrective decision-making when a significant delay is observed;
- Identify the quality level reached by the project, by task, by process, by manager, with regards to the rating scale reached.
- Simultaneously make an audit for both standards as a rating applied to a requirement is automatically applied to similar requirements in the other standard, in accordance with a certain rule.

5.5 Pilot Industrial Application Outcomes

5.5.1 Findings

Regarding the topics listed above concerning areas that need improvement, the general impression is that the contributions mainly support the development process.

Table 5-1. Summary of findings with applied workflow on BSG-E

	What is fully supported	What is partially supported
Topics expected to be improved for processes	• Tool support for the requirements analysis and system architecture definition • Automation of links between different phases for limitation of “from scratch” activities • Unique platform for requirements and system architecture manipulation • Automation of the process for limitation of manual activities • Integration of ISO26262 methods and tasks in the process • Consideration of SPICE methodology	• Static analysis for models (requirements, architecture)
Topics expected to be improved for tools	• Communication between tool that limits manual work • Support for architecture design • Support for converting textual requirements into design models • Gives an overview at any time of the complete project	• Means to trace requirements into all files • Enhance models exploration • Gives usable criteria for quality evaluation • Complete static analysis for models • Model the used process • Means to validate architecture

Significant improvements are appreciable with regards to the understandability of the requirements and their traceability, as well as the interoperability with the modeling environment. The ability to manage requirements coverage is the functionality that this point lacks. The import of textual requirements into Papyrus MDT, an easy way to obtain a requirements diagram, is a huge innovation as far as real industrial projects are concerned. Indeed, the requirements in this format extend the understandability of the development team compared with the textual representations. A requirements diagram shows groups and relations between requirements which is usually not directly visible in a traditional approach. It also eases manual reviews for completeness and consistency. The use of ReqIF formalism is a further extra solution to extend the interoperability of the solution. The ReMIAS profile allows the architecture to be defined and facilitating its representation. The advantages of this modeling are multiple. This helps the designer and facilitates the identification of the architecture elements since it stops architect engineers from wasting time wondering how to represent their architecture. Creating Papyrus MDT models is initially time-consuming and mostly concerns the classes' definition. But since the defined classes can be reused, this work can be limited once the initial work has been done.

Nevertheless, some observations are noteworthy, mainly concerning the tool support. We have used the Papyrus MDT tool for the product development. It is a mature and ergonomic tool. It proposes several functionalities to customize properties on the diagrams and diagram elements, etc. which facilitate the manipulation of the models. But it lacks a way to validate the architecture in comparison with the requirements. Currently, there is no support which confirms that the step between requirements analysis and architecture design has been carried out correctly. For example, it would useful to validate that all requirements are satisfied; the design is correctly defined regarding the internal and external interfaces, etc... The Papyrus MDT tool also offers many features that today we are unaware of or that we cannot use because we do not have sufficient information about them (e.g. merging a project, or merging a model) although it would have been very beneficial for a collaborative work.

The other main drawback is that we have not fully implemented the process modeling part in a graphical modeling language. Although this is available in an Excel framework, the framework is not sufficiently user-friendly to be widely used in industry. Furthermore, it fails when the volume of data to be managed becomes too large. The Permeter tool defined in the CESAR project is a good starting point that must be adapted with the addition of specific extensions we have thinking to address this weakness. In particular, an easy way to define criteria for quality assessment and control monitoring, automatic generation of a specific process for a project from a general one in accordance with parameters, interoperability between Papyrus MDT and Permeter to be able to easily use the product models (as workproducts) in the process model.

5.5.2 Benefits of the approach application

The solution proposed is a promising asset for embedded systems development in the field of intelligent transportation systems, where there is a huge margin to improve development processes. It aims to provide a better environment for the development of software and systems, management of requirements and architectures embodied metamodels, methods, and tools for safety-critical hard-real-time system development while making them interoperable. At the end of the evaluation, we can conclude that the contribution seems to fit the majority of our pilot application needs. Indeed, tracking requirements and system architecture design are covered, although the methodology or tools to validate them are missing. The automation of links between

different phases of the process at development level is ensured. This is denoted by following particular aspects:

- The conversion from textual requirements into graphical models using the Office2Papyrus plugin.
- The allocation of resulting requirements from the management step to architecture through the ReMIAS profile.
- The monitoring of the project and its current progress using the standard framework.
- The evaluation of the quality reached in the project with the assessment method.

5.6 Evaluation of thesis contributions

This thesis focuses on three contributions, namely:

- Merging of the ISO26262 and SPICE standards following a unique approach;
- Specification of the main requirements engineering activities in a model-based environment which considers safety aspects;
- Modeling and measuring of the product and process depending on the product quality specification.

Each of these contributions was evaluated using at least two forms of evaluation as described in the beginning of the section. To assess the thesis contributions, the development activities outcomes of our application have been evaluated in accordance with the criteria tables defined in section 3.
The first table 5-2 presents the criteria induced from the standards for ensuring that the requirements are properly managed. Our analysis showed that these criteria were covered by different modeling languages thereof, but to different degrees and not all at the same time.
The first criterion concerns the requirements characteristics quality and attributes. The DARWIN profile already met this requirement and in ReMIAS this part of the profile is reused.
The requirement allocation to architectural elements required the ability to define and use design elements. We used the architecture part of EAST-ADL2 to embody this feature in our profile. The third criterion was not filled by the state of practice mainly because it is specificity proper to the new standard ISO26262. Through the rules implemented as a static profile, this criterion is also met. Consequently, because it reuses the profile Part which fulfilled some of the criteria and implements some standards requirements as rules in static profiles, the ReMIAS profile meets all the criteria relating to our goal.

Table 5-2. Results against Standards criteria at product level

Standards criteria	SysML	DARWIN	EAST-ADL2	ReMIAS
C1. Requirement characteristics quality and attributes	✗	based on ISO9126 [ISO 07b] ✓	N/A	used the DARWIN specification ✓
C2. Requirement allocation on architectural elements	with allocation and block diagram ✓	N/A	Different architecture views ✓	Reuse EAST-ADL2 architecture views and static profile rules ✓
C3. Requirements hierarchy and structure	✗	✗	✗	Implemented by static profile rules ✓

The second criteria table 5-3 concerned the standards appropriation and their manipulation from a process point of view,: including their modeling and their monitoring, mainly with the Permeter tool. These criteria are mainly covered by merging both standards proposed and the assessment framework regarding the scientific basis. These are covered further by the third contribution with the modeling language extensions and the process customization approach. Although they are not developed technically, their conceptual evaluation suggests that they are promising innovations.

Table 5-3. Results against Standards criteria at process level

Standards criteria	SPEM	
C1. Process modeling	Classification or hierarchy of methods, tools and properties according to severity levels does not support any concept ✗	Implemented in the Excel assessment framework and the SPEM extension proposal ✓
C2. Process measurement	Quality rules of ISO26262, the determination of the ASIL, for example, is not taken into account in SPEM ✗	Possible to choose the requirements following the ASIL level in the Excel framework and SPEM extension ✓
C3. Specific process configuration	From a generic process, it is not possible to automatically generate a specific process for a project ✗	Specialization from constraints of the standards requirements for a specific project in Excel framework and EPF tooling proposal ✓
C4. Process monitoring	Process control is not possible as long as SPEM does not provide a running layer ✗	Proposed by the Permeter tool functionality ✓

Concerning this last contribution, a further advantage is that the process part is linked with the product development as the metamodel extension proposed by the CESAR project allows the product models to be considered as workproducts in the process model.

5.7 Conclusion

This chapter presented an evaluation of the three main contributions of the research presented in this thesis. This evaluation was carried out by means of peer review, tool support and a pilot industrial application which clearly illustrates electronic embedded systems, such as multiform requirements, system architecture, real-time and safety-related properties, etc… These forms of evaluation generated results supporting the hypothesis of this thesis. The final conclusions of this research are presented in the next section.

6

General Conclusions

6 General Conclusions

6.1 Introduction

This chapter concludes the thesis. The research questions introduced in the first section are revisited and the research contributions summarized. The contributions are evaluated against the success criterion. Limitations are discussed and opportunities for further work highlighted.

6.2 Review of research questions

This research project deals with three major domains: requirements engineering, safety and quality management in a modeling environment framework.
We have focused on three main areas of contribution, namely:

- The merging of the ISO26262 and SPICE standards with the aim of proposing a unique framework for certification of automotive systems;
- The requirements metamodel incorporating safety recommendations concerning system engineering activities involving requirements and system architecture;
- A process metamodel including a measurement part to assess the product and process relationships.

The research contributions are reviewed to answer the research questions. As a reminder, the purpose of this dissertation is to: **define a synthesis between product and process development following a model-based paradigm and to provide integration of standards reference, formulated in term of requirements.**

In section 1, we have decomposed this statement into some research questions. Firstly, w*hat properties are necessary in a requirements engineering modeling framework?*
This question was addressed by developing the ReMIAS profile based on a reading of the ISO26262 standard for managing requirements attributes and the SPICE standard for being inspired by the process for requirements activities. To summarize, the key attributes of the contribution are:
Requirements specification. This was identified as an effective approach to bring quality into requirements management.

Management of safety aspects in requirements engineering. This aspect considers ASIL determination and classification regarding this ASIL for requirements. The first step is to proceed to safety requirements management and then to apply the ISO26262 recommendations.

Hierarchical structure and classification. Following the safety standard, the requirements structure and hierarchy is different from common research because “safety” is not a particular quality requirement but another new type of requirement which follows specific rules to ensure the safe aspect of automotive systems.

System architecture definition and requirements allocation to architectural elements. The metamodel provides traceability from requirements to architectural elements as recommended by the standards.

The second research question was w*hat are the requirements and attributes to formalize the development processes regarding their enactment, their measurement and monitoring?*
This question was addressed by developing a process metamodel together with a measurement metamodel. To summarize, the key innovations are:
Process metamodel integrating the ISO26262 and SPICE assets. This was necessary to handle the elements support to follow and assess compliance with ISO26262 like the ASIL, the tools and methods classification according to some recommended levels, the capability levels and process attributes, etc…

Specific process following a specific project's objectives. Creating a process which complies with the automotive standards and that is in the same time fitness following the specific constraints of a project within a context which allows project management to be improved.

Metrics and goals definition. The definition of metrics and goals is the way to measure the effectiveness and evolution of process activities and tasks. Measurement is used for two purposes: to evaluate product quality and to evaluate process effects. Throughout performance, these definitions help in follow-up the completeness of process activities, the readiness of workproducts and monitoring of the project.

The final research question was *what methodology is efficient enough to merge the certification approaches for both process development and end-product?*
This question is answered by the development of a framework where the ISO26262 and SPICE requirements are merged. Key innovations in this contribution are matching both requirements standards and developing a unified assessment tool for certification. Our contribution will allow system engineers to evaluate the adequacy and the maturity of theirs processes and systems for certification.

The following additional contributions were made by this research project:
Providing a common language for supporting communication between different stakeholders. A key challenge in industrial projects is negotiating a common perspective. Therefore, this is supported by an intuitive and rich representation based on a graphical notation. This also frees us from multi different formalisms

Model-based tool chains. To support traceability from the first specifications documents to the architecture definition, our approach propose Office2Papyrus plugin, based on a model-based environment to limit "from scratch" and manual activities, commonly known as source of errors, traceability package in ReMIAS profile for ensuring traceability between modeling elements, integration between product and process models for traceability with the development activities
Thus, the research project met all objectives stated in section 1.

6.3 Validity of research contributions

Section 1 stated that this research would benefit both academic research and industry practice. The success criterion therefore required that the research outputs must be validated in industry and to contribute to other academic research. This was fully achieved.

Application in industry. The certification framework was used in DELPHI to develop an audit methodology which implied application of both the SPICE and ISO26262 recommendations. The requirements and architecture metamodel have been used by system engineers to improve their compliance with the SPICE standard covering the System architecture engineering process (ENG 3), which was not at all addressed. The theory behind obtaining a specific process from a generic one, which was partially implemented through the certification framework has received much interest from quality engineers in the sense that they can focus directly on standards recommendations which impacts a present project and they can free from those which are not applicable within this context, thus avoiding overkill procedures. These users have invested significant time in applying the research results, thereby indicating its perceived relevance. We might expect also to provide this tool to various certification bodies as a basis for conformity assessment, adapting them so they can eventually conform to the format used currently.

Contribution to research. The ISO26262 study forms a substantial component of a collaborative research project SASHA. The traceability management from specifications documents to a model-based environment has received significant interest from CESAR, a European collaborative research project, where it is cited as a main result. The contribution to research is thereby validated.

Furthermore, these contributions have resulted in several publications and are source of collaboration contacts with many actors in industry. This further indicates the relevance of the contributions to academic research and industry practice.

6.4 Limitations

The primary general limitation of this thesis is the lack of tool support. Some theoretical research results have not been implemented and thus cannot be completely evaluated. The second limitation is that the evaluation focus mainly on a single company in the automotive industry whereas some parts of the contribution can be used in others areas of embedded systems. However, many discussions with industry and academic experts have supported the findings and indicated the potential for applicability outside the company. Exploring the generality of the findings through additional case studies is an opportunity for further research.

A number of specific limitations arise from the modeling framework on which it is based. Models which incorporate hundreds of thousands of activities, tasks in a process context, requirements, and architecture elements in a system point of view cannot be easily modeled using the approach. This shortcoming is common to all modeling frameworks based on a graphical notation. The tools used proposed a model explorer view to manipulate the elements and this tree view was widely used during the application studies due to its greater accessibility. Further research regarding model visualization in a graphical way is necessary to address this limitation.

Once a specific process is generated from the generic one, it is unsuited to represent new or modified activities based on the current state of project progression. Incorporating adaptive tasks or activities into the process model would provide an interesting opportunity for further work since adaptive behavior is an important characteristic of many processes.

The constraint definition using the OCL language is not easily accessible to engineers as they lack the required education on this. In addition, providing a graphical interface to manage these constraints is primordial to the usability of the process and measurement parts manipulation. Recent developments have been done to overcome this limitation.

6.5 Opportunities for further research

During the course of this research, some areas have been identified for further investigations.
A short-term opportunity for further work is to refine the ReMIAS profile and to integrate some others standard recommendations so it can be applied to multiple domains.
With Permeter tool and different assessment functions implemented, we collect some measurement data according to a project plan: quality status, progress metrics, tasks achievement, etc… An essential reflection would be to see how far can be considered classic certification document generation from these data elements.
We used different modeling tools which are based on the Eclipse framework and inspired of UML metamodel. With regards to further works, it would be better to access these multiple modeling environments within the same software tool resulting in a 'framework wizard'. The tool integration can lead to substantial efficiency improvements by reducing the time spent on data transfer and conversion tasks. This would require theoretical and implementation research since the framework structures pre-suppose individual metamodel conceptualizations and class structures.
Regarding the above research works proposed, it would also be very useful to have true interoperability between the system architecture and the detailed design realized by Simulink in some way, although it is commercial tool. Indeed, UML is a graphical language, which allows static representations to be made. It does not include mechanisms for simulation. However, in the automotive industry, generally, Simulink models describe the behavior of applications. EAST-ADL2 proposes referencing Simulink modules in the behavior of components but it only provides information about a path for the simulation file. Some academic research works have been conducted to transform a system architecture defined with EAST-ADL2 in Simulink blocks. We could go further. The goal would be to provide innovative solutions to animate UML models from data resulting from the simulation of Matlab Simulink / Stateflow models associated, in order to obtain an early assessment of the architectural choices made at system level. This requires exploring two main areas: firstly, the definition of a method for designing system architecture to allow interaction with Simulink and Stateflow for animation / simulation purposes; and secondly, the definition of mechanisms (such as the timing constraints description) to animate architecture models with the simulation data from Matlab Simulink / Stateflow models and give feedback to engineers.

Other research opportunities arise directly from the limitations of the contributions of this thesis regarding the adaptability of the process models. In this context, one research question can be investigated: *How can alternative corrective plans be automatically elaborated and proposed to a human user to complete a project whose execution deviates from the initial or more recent plan?*
Always in the adaptability context, two other research questions arise:

- How can existing process models be reused through adaptation, composition or weaving to build wider scope process models?
- How can process models be semi-automatically built from the mere specification, with some input artifacts already available?

6.6 Conclusion

This thesis has introduced a model-based requirements engineering (RE) process which takes into account needs defined in standards for certification support; specifically needs concerning the

automotive domain with the ISO26262 and SPICE standards. Three separate working areas are explicitly addressed:

- Requirements engineering, during which product quality targets are set.
- Process engineering with as main output a development process model which describes the steps taken in the project to develop the required product quality.
- Measurement engineering, during which product quality and the effects of process actions are measured. These measurements are analyzed and provide feedbacks both to requirements engineers on the attainment of product quality targets and also to process engineers on the effects of the development process.

The approach has been evaluated and applied in a major automotive supplier where it appears to be useful. Some parts of the contribution have also been used in research projects.

Appendix A

Overview and document flow of product development at system requirement and system architecture level in HIS Automotive SPICE

Process ID	ENG.2
Process Name	System requirements analysis
Process Purpose	The purpose of the System requirements analysis process is to transform the defined customer requirements into a set of desired system technical requirements that will guide the design of the system.
Process Outcomes	As a result of successful implementation of this process: 1) a defined set of system requirements is established; 2) system requirements are categorized and analyzed for correctness and testability; 3) the impact of the system requirements on the operating environment is evaluated; 4) prioritization for implementing the system requirements is defined**;** 5) the system requirements are approved and updated as needed; 6) consistency and bilateral traceability are established between customer requirements and system requirements; 7) changes to the customer's requirements baseline are evaluated for cost, schedule and technical impact; and 8) the system requirements are communicated to all affected parties and baselined. NOTE 1: System requirements may be categorized in terms of feasibility and risk. NOTE 2: System requirements may typically include functional, performance, interface, design requirements and verification criteria. Verification criteria define the qualitative and quantitative criteria for verification of a requirement. Verification criteria demonstrate that a requirement can be verified within agreed constraints. NOTE 3: Analysis of system requirements for testability includes development of verification criteria.
Base Practices	**ENG.2.BP1: Identify System Requirements.** Use the customer requirements as the basis for identifying the required functions and capabilities of the system and document the system requirements

	in a system requirements specification. [Outcome 1] NOTE 1: System requirements include: functions and capabilities of the system; business, organizational and user requirements; safety, security, human-factors, engineering (ergonomics), interface, operations, and maintenance requirements; design constraints and qualification requirements (ISO/IEC 12207) as well as application parameters influencing system functions and capabilities. **ENG.2.BP2: Analyze system requirements.** Analyze the identified system requirements in terms of technical feasibility, risks and testability. [Outcome 2] NOTE 2: Verification criteria for all system requirements should be defined for further development of system test cases. NOTE 3: The results of the analysis may be used for categorization of the requirements (see also ENG.2.BP.4). **ENG.2.BP3: Determine the impact on the operating environment.** Determine the interfaces between the system requirements and other components of the operating environment, and the impact that the requirements will have. [Outcome 3] **ENG.2.BP4: Prioritize and categorize system requirements.** Prioritize and categorize the identified and analyzed system requirements and map them to future releases of the system. [Outcomes 2, 4] NOTE 4: Refer to the process SPL.2 Product Release. **ENG.2.BP5: Evaluate and update system requirements.** Evaluate system requirements and changes to the customer's requirements baseline in terms of cost, schedule and technical impact. Approve the system requirements and all changes to them and update the system requirements specification. [Outcome 5**,** 7] **ENG.2.BP6: Ensure consistency and bilateral traceability of customer requirements to system requirements.** Ensure consistency of customer requirements to system requirements including verification criteria. Consistency is supported by establishing and maintaining bilateral traceability between the customer's requirements and system requirements including verification criteria. [Outcome 6] ***ENG.2.BP7: Communicate system requirements.*** Establish communication mechanisms for dissemination of system requirements, and updates to requirements to all relevant parties. [Outcome 8]

Output Work Products
01-51 Application parameter [Outcome 1]
08-16 Release Plan [Outcome 4, 5]
13-04 Communication record [Outcome 8]
13-21 Change control record [Outcome 7]
13-22 Traceability record [Outcome 6]
15-01 Analysis report [Outcome 2, 3, 4, 7]
17-08 Interface requirements specification [Outcome 3]
17-12 System requirements specification [Outcome 1,5]
17-50 Verification criteria [Outcome 2]

NOTE: For system requirements specifications, the IEEE-Standard 1233-1998, Guide for Developing System Requirements Specifications might be used.

Process ID	ENG.3
Process Name	System architectural design
Process Purpose	The purpose of the System architectural design process is to identify which system requirements are to be allocated to which elements of the system.
Process Outcomes	As a result of successful implementation of this process: 1) a system architectural design is defined that identifies the elements of the system and meets the defined systems requirements; 2) the system requirements are allocated to the elements of the system; 3) internal and external interfaces of each system element are defined; 4) verification between the system requirements and the system architectural design is performed; 5) consistency and bilateral traceability are established between system requirements and system architectural design; and 6) the system requirements, the system architectural design, and their relationships are baselined and communicated to all affected parties. NOTE: Definition of system architectural design includes development of verification criteria. Verification criteria define the qualitative and quantitative criteria for verification of a requirement. Verification criteria demonstrate that a requirement can be verified within agreed constraints.

Base Practices	**ENG.3.BP1: Define system architectural design.** Establish the system architectural design that identifies the elements of the system with respect to the functional and non-functional system requirements. [Outcome 1] NOTE 1: The system might be decomposed into several subsystems on different system levels, if necessary. **ENG.3.BP2: Allocate System Requirements.** Allocate all system requirements to the elements of the system architectural design. [Outcome 2] **ENG.3.BP3: Define Interfaces.** Identify, develop and document the internal and external interfaces of each system element. [Outcome 3] NOTE 2: Interfaces include specific interfaces required for application parameter usage. **ENG.3.BP4: Develop verification criteria.** Define the verification criteria for each element of the system concerning the functional and non-functional system requirements based on the system architectural design. [Outcome 1] **ENG.3.BP5: Verify System Architectural Design.** Ensure that the system architecture meets all system requirements. [Outcome 4] **ENG.3.BP6: Ensure consistency and bilateral traceability of system requirements to system architectural design.** Ensure consistency of system requirements including verification criteria to system architectural design including verification criteria. Consistency is supported by establishing and maintaining bilateral traceability between the system requirements including verification criteria and system architectural design including verification criteria. [Outcome 5] **ENG.3.BP7: Communicate system architectural design.** Establish communication mechanisms for dissemination of the system architectural design to all relevant parties. [Outcome 6]

Output Work Products
01-00 Configuration item [Outcome 6]
04-06 System architectural design [Outcome 1, 2, 3, 4]
13-04 Communication record [Outcome 6]
13-22 Traceability record [Outcome 1, 5]
13-25 Verification results [Outcome 4]
17-50 Verification criteria [Outcome 1]

Appendix B

Overview and document flow of product development at system requirement and system architecture level in ISO26262

Clause ID	(Part 3) 8	
Clause name	Functional safety concept	
Objectives	The objective of the functional safety concept is to derive the functional safety requirements, from the safety goals, and to allocate them to the preliminary architectural elements of the item, or to external measures.	
Inputs to this clause	Prerequisites	item definition in accordance with 5.5 hazard analysis and risk assessment in accordance with 7.5.1 safety goals in accordance with 7.5.2
	Further supporting information	preliminary architectural assumptions (from external source)
Workproducts	Functional safety concept resulting from the requirements of 8.4.1 to 8.4.4. Verification report of the functional safety concept resulting from the requirements of 8.4.5.	

Clause ID	(Part 4) 6	
Clause name	Specification of the technical Safety requirements	
Objectives	The objective of the initiation of the product development at the system level is to determine and plan the functional safety activities during the individual subphases of system development. This also includes the necessary supporting processes described in ISO26262-8. This planning of system-level safety activities will be included in the safety plan.	
Inputs to this clause	Prerequisites	Functional safety concept in accordance with ISO26262-3:2011, 8.5.1 Validation plan in accordance with 5.5.4
	Further supporting information	safety goals (see ISO26262-3:2011, 7.5.2); functional concept (from external source, see ISO26262-3:2011, 5.4.1); preliminary architectural assumptions (from external source, see ISO26262-3:2011, 8.3.2)
Workproducts	6.5.1 Technical safety requirements specification resulting from the requirements 6.4.1 to 6.4.5. 6.5.2 System verification Report resulting from the requirement 6.4.6. 6.5.3 Validation plan (refined) resulting from requirement 6.4.6.2.	

Clause ID	(Part 4) 7	
Clause name	System design	
Objectives	The first objective of this subphase is to develop the system design and the technical safety concept that comply with the functional requirements and the technical safety requirements specification of the item. The second objective of this subphase is to verify that the system design and the technical safety concept comply with the technical safety requirements specification.	
Inputs to this clause	Prerequisites	Item integration and testing plan in accordance with 5.5.3 Technical safety requirements specification in accordance with 6.5.1
	Further supporting information	preliminary architectural assumptions (from external source, see ISO26262-3:2011, 8.3.2); functional concept (from external source); functional safety concept (see ISO26262-3:2011, 8.5.1).
Workproducts	7.5.1 Technical safety concept resulting from the requirements 7.4.1 and 7.4.5. 7.5.2 System design specification resulting from the requirements 7.4.1 to 7.4.5. 7.5.3 Hardware-software interface specification (HSI) resulting from requirements 7.4.6. 7.5.4 Specification of requirements for production, operation, service and decommissioning resulting from requirements 7.4.7. 7.5.5 System verification report (refined) resulting from requirement 7.4.8. 7.5.6 Safety analysis reports resulting from requirement 7.4.3	

Appendix C

Technical specification of Office2Papyrus plugin

The Office2Papyrus plugin is defined by component. A view of the global architecture is given below.

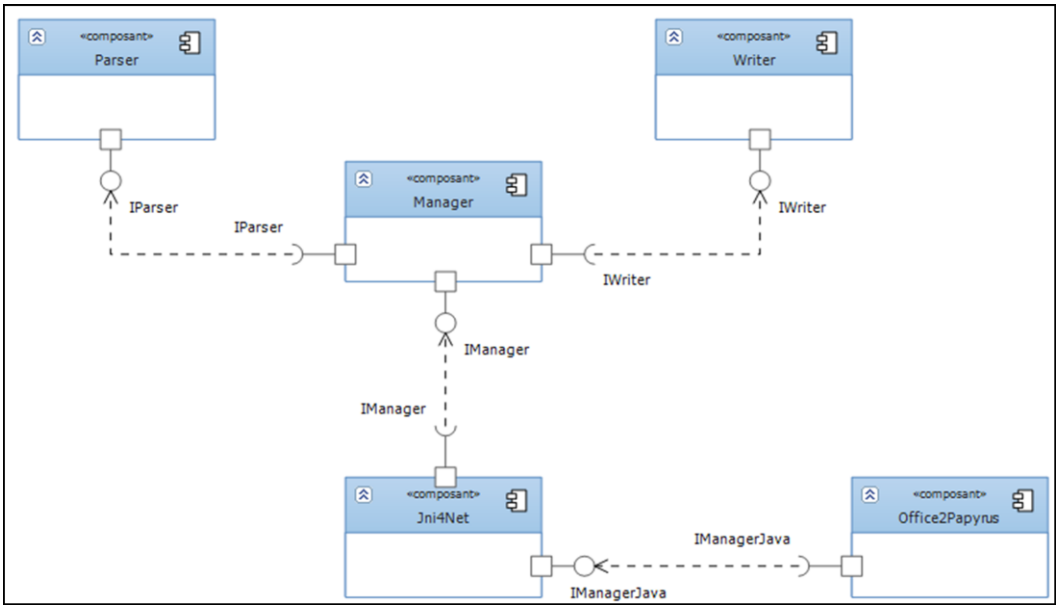


Figure C-70. Office2Papyrus component diagram

The **Parser** component offers the functionalities to read input files such as ExcelParser, UMLParser and WordPaser. It sets an event *RequirementRead*, triggered by the reading of each new requirement. Any component can subscribe to the event to be notified of a requirements reading.
The **Writer** component offers the functionalities to write output files such as ExcelWriter, ReqIFWriter, UMlWriter and WordWriter. Each *"writing"* consists of a queue in which the requirements are sequentially retrieved to write the output file.
The **JNI4Net** is an open source component which is responsible for managing the interoperability between the Manager component and the final plugin, Office2Papyrus.
The **Manager** is the component that allows all the transformations from Word, Excel and UML to be factorized. Called by the Office2Papyrus plugin through the component JNI4Net, it initializes the Parser component and appropriate Writers and declares a Thread for each of them. It then subscribes to the event *RequirementRead* and starts all Threads.
Office2Papyrus is the main component of the application. This is the Eclipse plugin that offers end users the ability to run all the features.
The following figure shows an example of the Office2Papyrus plugin when it is running. We focus on the representation of the operation of the different components and the communication between them. The example chosen is the generation of Excel to UML.

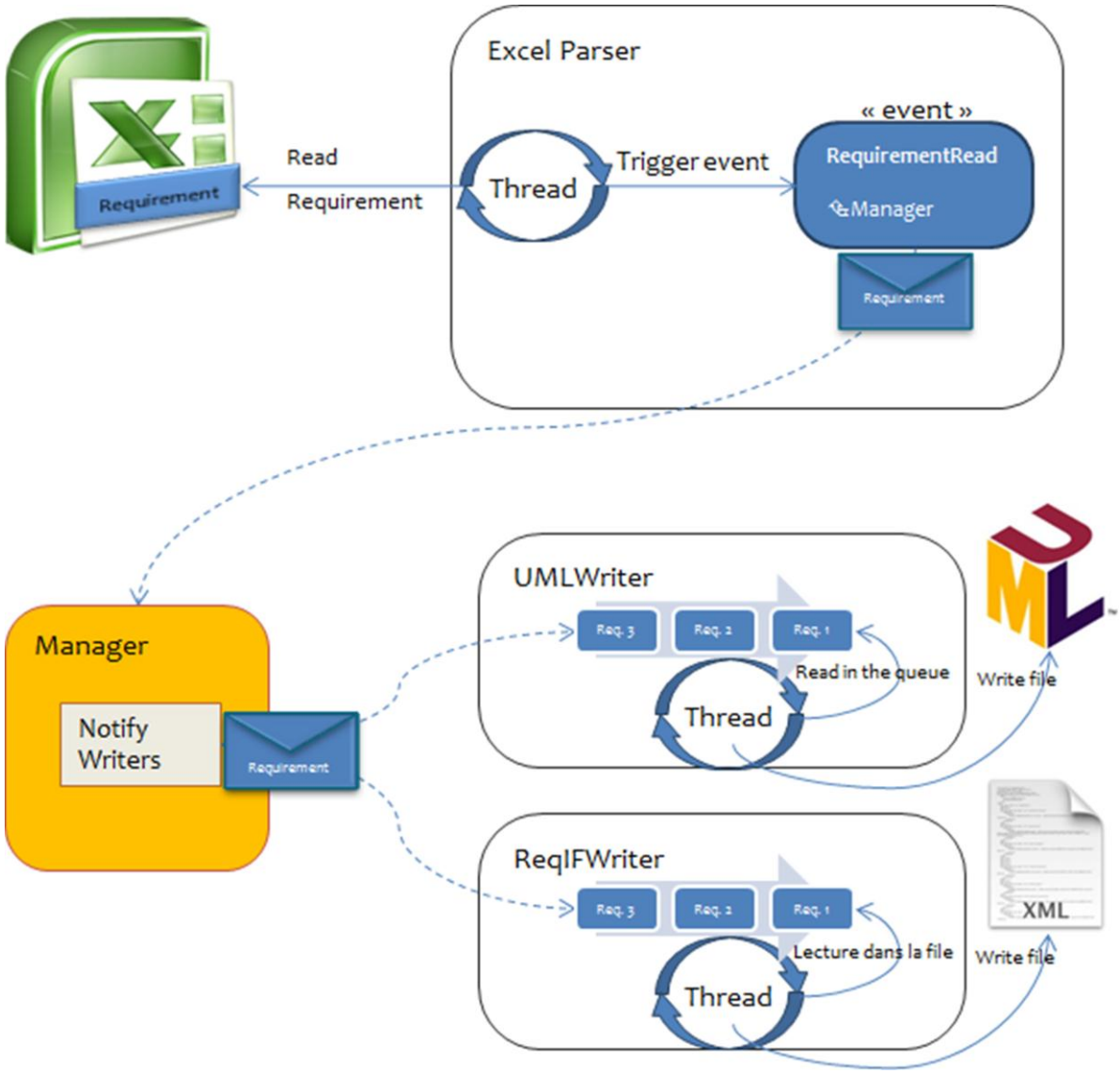


Figure C-71. Example of the plugin Office2Papyrus running

The sequence diagrams detail the generation progress of a UML file from Excel.

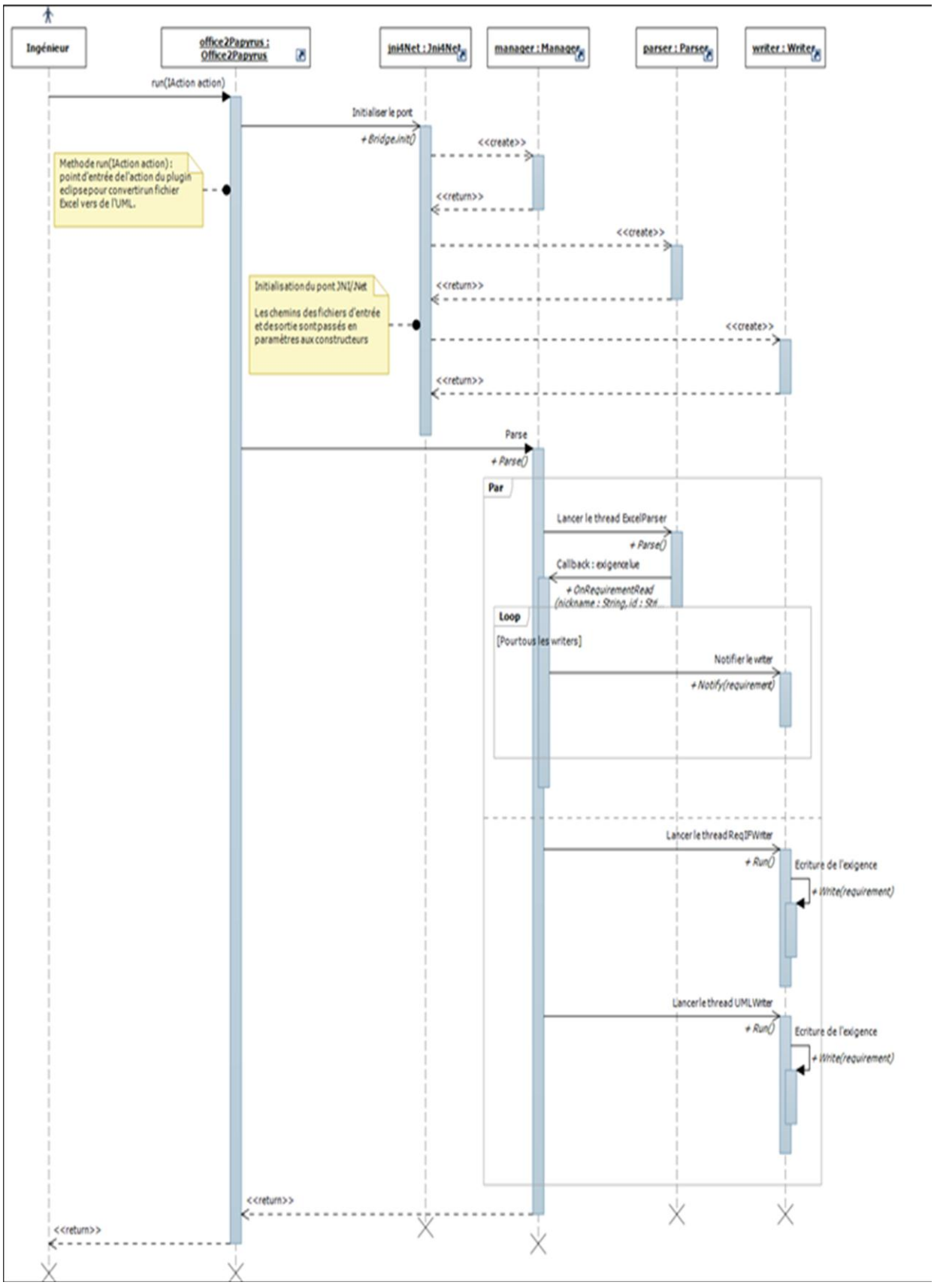


Figure C-72. Sequence diagrams detail the progress of the generation of a UML file from Excel

Appendix D

Running example of metamodel composition and extension

In chapter 4.2, we use an algorithm for the composition of SPICE and ISO26262 standards [FRA 07a, CHI 11, BAR 07]. The composition model proposed follows two steps applicable on single or group of elements (called pattern):

- The matching step that identifies the model elements (nodes or edges) that describe the same concepts in different models and that have to be composed;
- The merging step where the matched model elements are merged (to form a single class) to create new model elements that represent an integrated view of the concepts in the composed model.

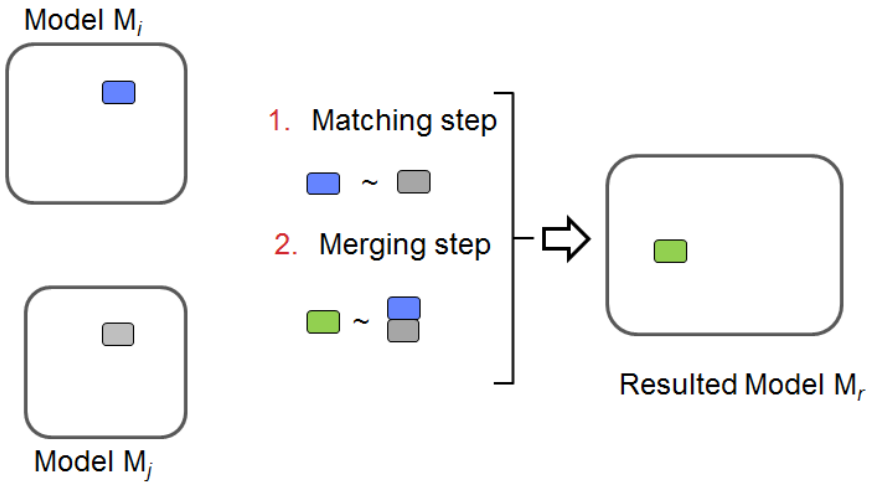


Figure D-73. Sequence diagrams detail the progress of the generation of a UML file from Excel

Let's us describe the algorithm application from an extract examples of the standards.

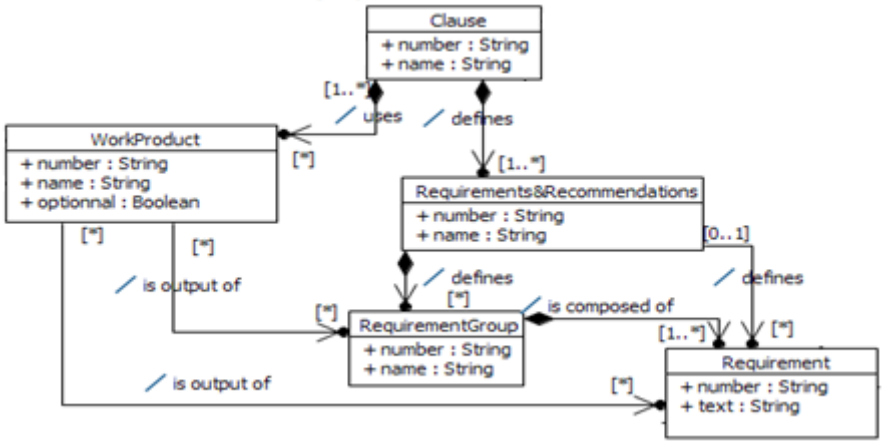


Figure D-74. Extract of ISO26262 model domain

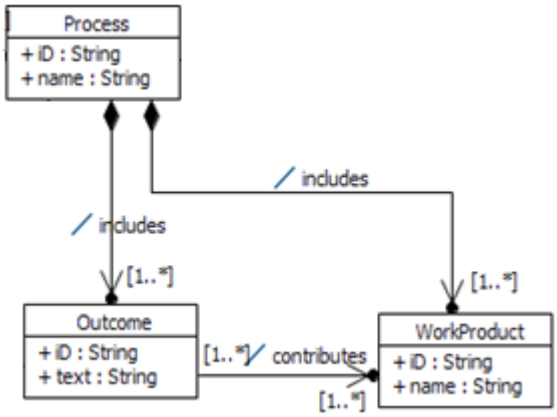


Figure D-75. Extract of SPICE model domain

Based on the following statement in ISO26262 "*The requirements or recommendations of each subclause shall be complied with for ASIL A, B, C and D, if not stated otherwise. If an ASIL is given in parentheses in ISO 26262, the corresponding subclause shall be considered as a recommendation rather than a requirement for this ASIL*", we found that Requirement and Recommendation have the same semantic except that a Recommendation is an optional requirement. In our model domain, Requirements&Recommendations and RequirementGroup are abstract elements. Then, this part of connected graph represents the Requirement concept.

In ISO26262, another statement says "*The results of safety activities are given as work products. In ISO26262, a work product is information or data which constitutes the result of one or more system safety process activities*". Workproduct is then result from requirements or set of requirements of a clause.

In SPICE, a process is defined under the following terms "*Each process is described in terms of a purpose statement. These statements contain the unique functional objectives of the process when performed in a particular environment. A list of specific outcomes is associated with each of the process purpose statements, as a list of expected positive results of the process performance*".

The definition of Workproduct is deduced from these others statements "*The presence of work products with their expected work product characteristics, provide objective evidence of the achievement of the purpose of the process*", "*The performance of a process produces work products that are identifiable and usable in achieving the purpose of the process*" or "*Work Product is cross-referenced to the Process Outcomes it addresses. All Work Products relate as Outputs to the Process as a whole*".

Workproduct in the two standards are the same meaning. In SPICE, the concept is result from Outcomes and in ISO26262 from Requirements. We apply the algorithm rules with matching and merging steps.

Furthermore, regarding their sense and their purpose, an Outcome has the same goal that Requirement. The two concepts are then similar. So, we apply the pattern matching principle between "Requirements&Recommendations, RequirementGroup, Requirement" and "Outcome".

Bibliography

Scientific References

[ABD 00] **Abdurazik, A:** Suitability of the UML as an Architecture Description Language with Applications to Testing. George Mason University, Technical Report ISETR-00-01, p. 24 (2002)

[ALB 08] **Albinet, A., Begoc, S., Boulanger, J.L., Casse, O., Dal, I., Dubois, H., Lakhal, F., Louar, D., Peraldi-Frati, M.A., Sorel Y., Van, Q.-D.:** The MeMVaTEx methodology: from requirements to models in automotive application design. In: the 4th European Congress on Embedded Real Time Software and Systems (ERTS²), Toulouse (2008)

[ALF 79] **Alford, M. W., and J. T. Lawson:** Software Requirements Engineering Methodology (Development). RADC-TR-79-168, U.S. Air Force Rome Air Development Center, Griffiss AFB, DDC-AD-A073132, New York (1979)

[APR 04] **April, A., Al-Shurougi, D.:** Software Product Measurement for Supplier Evaluation. In: Proceedings of the Software Measurement Conference (FESMA-AEMES), Madrid (2000)

[AWA 07] **Awan**, **R.:** Requirements Engineering Process Maturity Model for Market Driven Projects - The REPM-M Model. School of Engineering, Blekinge Institute of Technology, Phd thesis, Ronneby (2007)

[BAL 06] **Balmelli, L.:** An Overview of the Systems Modeling Language for Products and Systems Development. In: Journal of Object Technology (JOT) (2006)

[BAR 07] **Barbero, M., Jouault, F., Gray, J., Bézivin, J.:** A Practical Approach to Model Extension. In: Akehurst, D. H., Vogel, R., Paige, R. F. (Eds.): ECMDA-FA 2007, LNCS, Vol. 4530, pp. 32-42, Springer-Verlag Berlin, Heidelberg (2007)

[BAS 03] **Bass, L., Clements, P., Kazman, R.**: Software Architecture in Practice. Addison-Wesley Professional, 2nd edition (2003)

[BAS 90] **Basili, V., Caldiera, G., Rombach H.D.**: The Goal Question Metric Approach. Institute for Advanced Computer Studies, University of Maryland, Technical Report (1990)

[BEL 76] **Bell, T. E., Thayer, T. A.**: Software requirements: Are they really a problem?. In: Proceedings of the 2nd international conference on Software engineering (ICSE). IEEE Computer Society Press, pp. 61-68, Los Alamitos, CA (1976)

[BER 06] **Berander, P., Jonsson**, **P.:** A goal question metric based approach for efficient measurement framework definition. In: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering (ISESE), pp. 316-325, New York (2006)

[BEZ 02] **Bézivin, J.:** « Les nouveaux défis des systèmes complexes et la réponse MDA de l'OMG. Une approche de génie logiciel », Technical report, JFIADSMA'02, Lille (2002)

[BLA 03] **Blanquart, J.P., Muller, J.P.:** « Justification de sûreté de fonctionnement des logiciels spatiaux, évolution vers la certification ». In: « 5ème atelier of Justification de la sûreté de fonctionnement: approches industrielles, méthodes de construction et structure. Réseau d'Ingénierie de la Sûreté de fonctionnement », Toulouse (2003)

[BOE 78] **Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., Macleod, G. J., Merrit, M. J**.: Characteristics of Software Quality. North Holland Publishing, Amsterdam (1978)

[BOE 81] **Boehm, B.:** Software Engineering Economics. Prentice-Hall advances in computing science & technology series Englewood Cliffs (1981)

[BUR 06] **Burgaud, L.:** A Novel development framework combining requirement driven and model based engineering processes. In: 4ème Conférence Annuelle d'Ingénierie Système, Centre de Congrès Pierre Baudis, Toulouse (2006)

[BRI 96] **Brinkkemper, S.**: Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology, 38 (4). pp. 275-280, Elsevier Science (1996)

[CAB 07] **Caballero, I., Verbo, E., Calero, C., Piattini, M.**: A Data Quality Measurement Information Model Based On ISO/IEC 15939. In: Proceedings of the 12th International Conference on Information Quality, pp. 393-408, Cambridge (2007)

[CAB 08] **Cabot, J., Yu, E.:** Improving Requirements Specifications in Model-Driven Development Processes. In: Workshop Proceedings: International Workshop on Challenges in Model-Driven Software Engineering (ChaMDE), pp. 36–40, Toulouse (2008)

[CAN 06] **Canfora, G., Garcia, F., Piattini, M., Ruiz, F., Visaggio, C. A.:** Applying a framework for the improvement of software process maturity. In: Journal Software—Practice & Experience, Volume 36 Issue 3, John Wiley & Sons, pp. 283-304, New York (2006)

[CAR 06] **Card, D. N.:** The challenges of productivity measurement. In: Proceedings of the 24th Pacific Northwest Software Quality Conference, Portland (2006)

[CAR 98] **Card, D. N., El Emam, K.:** ISO/IEC 15939 – Software Measurement Process Framework. Software Productivity Consortium & Fraunhofer IESE (1998)

[CAS 10] **Casse, O., Reis Monteiro, M.:** A ReqIF/SysML profile example – Requirements exchange and roundtrip. In: the 5th Embedded Real Time Software and Systems (ERTS²), Toulouse (2010)

[CHI 11] **Chiprianov, V., Kermarrec, Y., Rouvrais, S.:** Practical meta-model extension for modeling language profiles: an enterprise architecture modeling language extension for telecommunications service creation. In: 7th Days of Model Driven Engineering, pp. 85-91, Lille (2011)

[CUE 08] **Cuenot, P., Frey, P., Johansson, R., Lönn, H., Reiser, M.-O., Servat, D., Tavakoli Koligari, R., Chen, D. J.:** Developing Automotive Products Using the EAST-ADL2, an AUTOSAR Compliant Architecture Description Language. In: the 4th European Congress on Embedded Real-Time Software (ERTS), Toulouse (2008)

[DAM 05] **Damas, C., Lambeau, B., Dupont, P., Van Lamsweerde, A.:** Generating Annotated Behavior Models from End-User Scenarios. IEEE Transactions on Software Engineering, 31, (12), pp. 1056-1073, (2005)

[DEC 09] **Dechev, D., Stroustrup, B.:** Model-Based Product-Oriented Certification. In: Proceedings of the 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), pp. 295-304, San Francisco, California (2009)

[DEL 04] **De Landtsheer, R., Letier, E., Van Lamsweerde, A.**: Deriving Tabular Event-Based Specifications from Goal-Oriented Requirements Models. Requirements Engineering Journal, 9, pp. 104-120, (2004)

[DEM 86] **Deming, W. E.:** Out of the Crisis. MIT Center for Advanced Engineering Study (1986)

[DEN 05] **Denney, E., Fischer, B.:** Software Certification and Software Certification Management Systems. In: Proceedings ASE Workshop on Software Certificate Management (SoftCement), Long Beach, California (2005)

[DER 11] **Deraman, A., Haji Yahaya, J.**: The Architecture of an Integrated Support Tool for Software Product Certification Process. Journal of Information & Systems Management, Volume: 1, Issue 1 (2011)

[DIC 03] **Dick, J.:** KeyChange for DOORS. Presentation, Integrate systems engineering (2003)

[DUB 10] **Dubois, H., Peraldi-Frati, M.-A., Lakhal, F.:** A model for requirements traceability in a heterogeneous model-based design process. In: the 15th IEEE International Conference on Engineering of Complex Computer Systems, pp. 233--244, UK (2010)

[DUS 02] **Balek, D.:** Connectors in Software Architectures. Doctoral thesis at Charles University, Czech Republic (2002)

[EAS 04] **Easterbrook, S. M**: What is Requirements Engineering?. In: Draft book, chapter 1 (2004)

[ELL 10] **Ellner, R., Al-H., S., Drexler, J., Jung, M., Kips, D., Philippsen, M.:** eSPEM – A SPEM Extension for Enactable Behavior Modeling. In: Modeling Foundations and Applications. Ed. Thomas Kühne and al. pp. 116-131, Springer Berlin, Heidelberg (2010)

[ESC 06] **Escalona, M-J, Koch, N.:** Metamodeling the Requirements of Web Systems. In: Web Information Systems and Technologies International Conferences, WEBIST 2005 and WEBIST 2006 Revised Selected Papers, Setúbal (2006)

[FAU 08] **Faulkner, S., Kolp, M., Wautelet, Y., Achbany, Y.:** A Formal Description Language for Multi-Agent Architectures. Agent Oriented Information Systems IV 4898, pp. 143-163 (2008)

[FEN 97] **Fenton, N. E., Pfleeger, S. L**.: Software Metrics – A Rigorous and Practical Approach. PWA Publishing Company, 2nd Edition, Boston, MA (1997)

[FOR 05] **Forsberg, K., Mooz, H., Cotterman, H.:** Visualizing Project Management. J. Wiley & Sons, Third Edition, New York (2005)

[FRA 07a] **France, R., Fleurey, F., Reddy, R., Baudry, B., Ghosh, S.:** Providing Support for Model Composition in Metamodels Providing Support for Model Composition in Metamodels. In: the 11th IEEE International Enterprise Distributed Object Computing Conference, IEEE Computer Society. pp. 253, Annapolis (2007)

[FRA 07b] **France, R., Rumpe, B.:** Model-driven Development of Complex Software: A Research Roadmap. Future of Software Engineering (FOSE), IEEE Computer Society, Washington, DC (2007)

[FUG 00] **Fuggetta**, **A.:** Software process: a roadmap. In: Proceedings of the 22nd International Conference on Software Engineering, pp. 25-34, Limerick (2000)

[GAR 06a] **Garcia, F., Bertoa, M.F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero**, **M.:** Towards a consistent terminology for software measurement. Information and Software Technology 48 (8), pp. 631–644 (2006)

[GAR 06b] **Garcia, F., Ruiz, F., Piattini, M., Canfora, G., Visaggio, C.A.:** FMESP: Framework for the modeling and evaluation of software processes. Journal of Systems Architecture, 52(11), pp. 627-639 (2006)

[GAR 09] **Garcia, F., Ruiz, F., Cruz, J. A., Piattini, M.**: Integrated measurement for the evaluation and improvement of software processes. In: Proceedings of the 9th European Workshop on Software Process Technology (EWSPT), LNCS, vol. 2786, Springer: Berlin, pp. 94–111. Helsinki (2003)

[GEE 07] **Geensys:** Reqtify – Bundles Definition/Interface List, (2007)

[GLI 00] **Glinz, M.:** Problems and Deficiencies of UML as a Requirements Specification Language. In: the 10th International Workshop on Software Specification and Design (IWSSD), San Diego (2000)

[GLI 07] **Glinz, M.:** On Non-Functional Requirements. In: the 15th IEEE International Requirements Engineering conference (RE), pp. 21--26 (2007)

[GOL 07] **Goldsby, H., Konrad, S., Cheng, B.:** Goal-Oriented Patterns for UML-Based Modeling of Embedded Systems Requirements. In: the 10th IEEE High Assurance Systems Engineering Symposium (HASE), USA (2007)

[GON 06a] **Gonzalez-Perez, C., Henderson-Sellers, B.:** An ontology for software development methodologies and endeavours. In: Ontologies in Software Engineering and Software Technology, Chapter 4; Springer, pp. 123-152 (2006)

[GON 06b] **Gonzalez-Perez, C., Henderson-Sellers, B.:** On the east of extending a powertype-based methodology metamodel. In: Proceedings of the Second Workshop on Metamodelling and Ontologies., (WoMM), LNI Vol 96, pp. 11-25 (2006)

[HAB 06] **Habli, I., Kelly, T.:** Process and product certification arguments: getting the balance right. SIGBED Rev. 3, 4 pp. 1-8 (2006)

[HAB 10] **Habli, I., Ibarra, I., Rivett, R., Kelly, T.**: Model-Based Assurance for Justifying Automotive Functional Safety. In: Proceedings of the SAE World Congress, Detroit, Michigan (2010)

[HEA 04] **Heaven, W., Finkelstein, A.:** A UML profile to support requirements engineering with KAOS. In: IEEE Proceedings - Software vol. 151, no1, pp. 10--27, Institution of Electrical Engineers, Stevenage, UK (2004)

[HEC 06] **Heck, P. M.:** A Software product certification model for dependable systems. CS-Report 06–20, Eindhoven: Technische Universiteit Eindhoven (2006)

[HEC 08] **Heck, P., Eekelen, M. v.:** LaQuSo Software Product Certification Model (LSPCM). URL: http://alexandria.tue.nl/repository/books/633706.pdf, (2008)

[HEC 10] **Heck, P., Klabbers, M., Marko, C. J. D., Eekelen, M. v.**: A software product certification model. Software Quality Journal 18(1), pp. 37-55 (2010)

[HEN 03] **Henderson-Sellers, B.**: Method engineering for OO systems development. Commun. ACM 46, 10, pp. 73-78 (2003)

[HEN 05] **Henderson-Sellers, B., Gonzalez-Perez, C.:** A comparison of four process metamodels and the creation of a new generic standard. Information and Software Technology, 47(1), pp. 49-65 (2005)

[HOE 08] **Hoermann, K., Mueller, M., Dittmann, L., Zimmer, J.:** Automotive SPICE in Practice: Surviving Interpretation and Assessment. Rocky Nook (2008)

[HUM 89] **Humphrey, W. S**.: Managing the software process. SEI series in software engineering, Addison-Wesley (1989)

[JAS 10] **Jastram, M.:** ProR, an Open Source Platform for Requirements Engineering based on RIF. In: SEISCONF (2010)

[JIA 04] **JIANG, L., Eberlein, A., Homayou, B.:** A methodology for requirements engineering process development. In: the 15th IEEE international conference and workshop on the engineering of computer-based systems No11, pp. 263-272 (2004)

[KAT 06] **Katasonov, A., Sakkinen, M.:** Requirements Quality Control: A Unifying Framework. Requirements Engineering 11(1): pp. 42-57, Springer-Verlag London, UK (2006)

[KEL 03] **Kelly, T. P.**: A Systematic Approach to Safety Case Management. In: SAE International, SAE World Congress, Detroit (2003)

[KEL 08] **Kelly, T. P.**: Can Process and Product-based Approaches to Software Safety be Reconciled?. In: Proceedings of 16th Safety Critical Systems Symposium (SSS), Springer (2008)

[KOL 09] **Kolovos, D.S., Di Ruscio, D., Pierantonio, A., Paige, R.F.:** Different models for model matching: An analysis of approaches to support model differencing. In: the International 2009 ICSE Workshop on Comparison and Versioning of Software Models, pp. 1-6. IEEE Computer Society (2009)

[KON 06] **Konrad, S., Goldsby, H., Lopez, K., Cheng, B.:** Visualizing Requirements in UML Models. In: the 1st international workshop on Requirements Engineering Visualization, IEEE Computer Society, USA (2006)

[KOW 12] **Kowalczyk, M., Steinbach, S.:** Managing Process Model Compliance in Multi-standard Scenarios Using a Tool-supported Approach. In: Dieste, O., Jedlitschka, A., Juristo, N. (eds.) PROFES 2012, LNCS, Vol. 7343, pp. 355-360. Springer-Verlag Berlin, Heidelberg (2012)

[KNE 06] **Kneuper, R.:** CMMI. Dpunkt, 2nd edition (2006)

[LAD 03] **Ladier, G.:** « Le DO 178-B/ ED-12B logique, historique, contenu, application ». Airbus France white paper (2003)

[LAP 96] **Laprie, J. C.:** « Guide de la sûreté de fonctionnement ». Edition Cépaduès, Edition 2 (1996)

[LAM 11a] **Lami, G., Fabbrini, F., Fusani, M.:** ISO/IEC 15504-10: Motivations for Another Safety Standard. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) COMPAS 2011, LNCS, Vol. 6894, pp. 284-295. Springer-Verlag Berlin, Heidelberg (2011)

[LAM 11b] **Lami, G.:** ISO/IEC 15504-10 Safety Extension, Yet Another Safety Standard?. Report, Automotive SPIN Italia Workshop (2011)

[LET 02] **Letelier, P.:** A Framework for Requirements Traceability in UML-based Projects. In: the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, in conjunction with the 17th IEEE International Conference on Automated Software Engineering, pp. 32--41, U.K (2002)

[LET 04] **Emmanuel, L., Van Lamsweerde, A.:** Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. In: FSE'04, the 12th ACM International Symposium on the Foundations for Software Engineering, pp. 53-62, Newport Beach (2004)

[LET 05] **Emmanuel, L., Jeff, K., Jeff, M., Sebastian, U.**: Monitoring and Control in Scenario-Based Requirements Analysis. In: Proceedings of the 27th international conference on Software engineering, pp. 382–391, St Louis, Missouri (2005)

[LIN 96] **Linger, R.C., Trammel, C. J.**: Cleanroom Software Engineering Reference Model v1.0. Technical Report CMU/SEI-96-TR-022 (1996)

[LOR 11] **Lorenzo, V., Schnekenburger, R., Tanguy, Y., Tessier, P., Gérard, S.**: « MDT::Papyrus, statut actuel et perspectives ». Journée Neptunes 2011, IRIT, Paris (2011)

[MAC 12] **Machrouh, J., Blanquart, J. P., Baufreton, P., Boulanger, J. L., Delseny, H. Gassino, J., Ladier, G., Ledinot, E., Leeman, M., Astruc, J. M., Quéré, P., Ricque, B.**: Cross domain comparison of System Assurance. In: ERTS² 2012 Congress, Toulouse (2012)

[MAD 09] **Behera, M.:** Tool-Supported Timing Analysis of Automotive Embedded Systems. Report - IT University of Göteborg, Chalmers University of Technology and the University of Göteborg (2009)

[MAI 04] **Maiden, J.:** RESCUE Process: Examples. Version 2.1; RESCUE process project document (2004)

[MAI 08] **Maibaum, T., Wassyng, A**.: A Product-Focused Approach to Software Certification. Computer, 41(2): pp. 91–93, (2008)

[MAI 96] **Maiden, R.:** ACRE: Selecting Methods For Requirements Acquisition. In: Software Engineering Journal, 11(3), pp. 183-192 (1996)

[MAT 12] **Simulink.** URL: http://www.mathworks.fr/products/simulink/

[MCD 84] **McDermid, J., Ripken, K.:** Life cycle support in the ADA environment. University Press (1984)

[MCG 02] **McGary, J., Card, D., Jones, C., Layman, B., Clark, W., Dean, J., Hall, F.:** Practical Software Measurement Objective Information for Decision Makers. Addison-Wesley (2002)

[MED 09] **Meding, W., Staron, M.:** ISO/IEC 15939 Creating an efficient measurement program. Ericsson white paper (2009)

[MES 09] **Messnarz, R., Ross, H.-L., Habel, S., König, F., Koundoussi, A., Unterrreitmayer, J., Ekert, D.:** Integrated Automotive SPICE and safety assessments. In: Software Process: Improvement and Practice, 14(5), pp. 279-288. John Wiley & Sons, Ltd. (2009)

[MOD 07] **UK Ministry of Defence (MoD):** Safety Management Requirements for Defence Systems. UK Ministry of Defence, LA (2007)

[MOO 01] **Mooz, H., Forsberg, K.**: A Visual Explanation of Development Methods and Strategies Including the Waterfall, Spiral, Vee, Vee+, Vee++ Models. In: Proceedings of the International Council for Systems Engineering (INCOSE) Conference, Melbourne (2001)

[MOR 03] **Morasca, S.**: Foundations of a Weak Measurement-Theoretic Approach to Software Measurement. In: Proceedings of Fundamental Approaches to Software Engineering (FASE), pp. 200-215 (2003)

[ODE 94] **Odell, J.**: Power Types. JOOP 7(2): pp. 8-12 (1994)

[OJA 04] **Ojala**, **P.:** Combining Capability Assessment and Value Engineering: A BOOTSTRAP Example. In: Frank Bomarius, Hajimu Iida(Eds.), Proceedings of the 5th International Conference on Product Focused Software Process Improvement (PROFES), LNCS 3009 Springer, pp. 471-484, Kansai Science City (2004)

[PAR 96] **Park, R.E., Goethert, W.B., Florac**, **W.A.:** Goal-driven Software Measurement — A Guidebook. Technical Report CMU/SEI-96-HB-002, Software Engineering Institute, Carnegie Mellon University, Pittsburgh (1996)

[PET 09] **Petry, E.:** Automotive SPICE® & ISO/CD 26262, Their Mutual Relationship. Report, Fifth Automotive SPIN Italia Workshop (2009)

[PET 10] **Petry, E.:** How to Upgrade SPICE-Compliant Processes for Functional Safety. Tutorial, SPICE Conference 2010, Pisa (2010)

[POH 05] **Pohl, K., Böckle, G., van der Linden, F.J**.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Heidelberg (2005)

[POH 11] **Pohl, K., Rupp, C.:** Requirements Engineering Fundamentals. Publisher: Rocky Nook 1st edition, Pages: 184 (2011)

[POH 96] **Pohl, K.:** Process-Centered Requirements Engineering. Wiley/Research Studies Press, New York (1996)

[PON 07] **Ponsard, C., Massonet, P., Molderez, J. F., Rifaut, A., Van Lamsweerde, A., Van, H. T.:** Early Verification and Validation of Mission-Critical Systems. In: Proceedings of Formal Methods in System Design (FMICS), pp. 233-247, Linz (2007)

[PRO 12] **ProSTEP iViP: RIF mapping table.** URL: http://www.prostep.org/fileadmin/freie_downloads /Empfehlungen-Standards/ProSTEP_iViP/PSI6_RIF_1.2_Mapping-Table.pdf.

[PRO 99] **Prowell, S.J, Trammell, C.J., Linger, R.C., Poore, J.H.**: Cleanroom Software Engineering: Technology and Process. Addison-Wesley (1999)

[RAL 01] **Ralyté, J., Rolland, C.:** An Assembly Process Model for Method Engineering. In: Proceedings of the 13th International Conference on Advanced Information Systems

Engineering (CAiSE), Klaus R. Dittrich, Andreas Geppert, and Moira C. Norrie (Eds.), Springer-Verlag, pp. 267-283, London (2001)

[RE 01] **Requirement Engineering**: Call for papers. Fifth IEEE International Symposium on Requirement Engineering, Toronto (2001)

[ROD 06] **Rodriguez, A., Fernandez-Medina, E., Piattini, M.:** Security Requirement with a UML 2.0 Profile. In: the First International Conference on Availability, Reliability and Security (ARES), Austria (2006)

[ROY 87] **Royce, W.:** Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, pp. 1–9, Los Angeles (1970), reprinted in Proceedings of the Ninth International Conference on Software Engineering (ICSE), pp. 328—338, (1987)

[SAN 08] **Sandmann, G., Thompson, R:** Development of AUTOSAR Software Components within Model-Based Design. In: SAE World Congress 2008, Detroit, Michigan (2008)

[SEL 07] **Selic, B.:** A Systematic Approach to Domain-Specific Language Design Using UML. In: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC). IEEE Computer Society, Washington DC (2007)

[SER 08] **Servat, D., Tessier, P., Gerard, S., Cuenot**, **P.:** An MDE approach for automotive with EAST-ADL2.In: Journées Neptune No5, pp. 59-63, Paris (2008)

[SHA 96] **Shaw, M., Garlan, D.:** Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall, 1st Edition (1996)

[SCH 96] **Schlenoff, C., Knutilla, A., Ray, S.:** Unified Process specification Language: requirements for modeling Process. NISTIR 5910, National Institute of Standards and Technology, Gaithersburg, MD (1996)

[SCH 02] **Schätz, B., Pretschner, A., Huber, F., Philipps, J.:** Model-Based Development of Embedded Systems. In: Congrès Advances in object-oriented information systems workshop No8 (OOIS), Montpellier (2002)

[SOM 10] **Sommerville**: Software Engineering. Ninth Edition, Addison-Wesley (2010)

[STA 95] **The Standish Group Report**: CHAOS, the Standish Group 1995. The Standish Group International, Inc The CHAOS Report (1996, 1998, 2000, 2002, 2004, and 2006), URL: www.standishgroup.com

[STA 01] **Stafford, J., Wallnau, K. C.:** Is Third Party Certification Necessary?. In: Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering (CBSE), Canada (2001)

[STA 09] **Staron, M., Meding, W.**: A Modeling Language for Specifying and Visualizing Measurement Systems for Software Metrics. Tampere University of Technology research report, Nordic workhop on MDE, pp. 300-315, Tampere (2009)

[STE 08] **Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.**: EMF: Eclipse Modeling Framework. 2nd edition, Addison-Wesley Professional (2008)

[STE 94] **Stevens, R., Paltu, M.:** Subject to Requirements. Computing (1994)

[TRA 04] **Tran Van, H., Van Lamsweerde, A., Massonet, P., Ponsard, C.**: Goal-Oriented Requirements Animation. In: Proceedings of RE 2004: 12th IEEE International Requirements Engineering Conference, pp. 218-228 Kyoto, (2004)

[TRI 01] **Trienekens, J., Kusters, R., Van solingen, R.:** Product Focused Software Process Improvement: Concepts and Experiences from Industry. In: Software Quality Journal, Kluwer Academic Publishers, pp. 269–281, Netherlands (2001)

[VAN 02] **Van Solingen, R., Basili, V., Caldiera, Gianluigi, and Rombach, D. H.**: Goal Question Metric (GQM) Approach. Encyclopedia of Software Engineering (Marciniak, J.J. ed.), online version @ Wiley Interscience, John Wiley & Sons (2002)

[VAN 08] **Van Lamsweerde, A.:** Requirements engineering: From craft to discipline. In: Proceedings of the 16th International Symposium on Foundations of Software Engineering (FSE), Lausanne (2008)

[VAN 09] **Van Lamsweerde, A.:** Requirements Engineering, From System Goals to UML Models to Software Specifications. Wiley (2009)

[VOA 07] **Voas, J.**: Software Product Certification. Wiley Encyclopedia of Computer Science and Engineering (2007)

[WA 99] **Florac, W.A., Carleton, A. D.**: Measuring the Software Process. Statistical Process Control for Software Process Improvement. Addison-Wesley: Reading, MA (1999)

[YAH 10] **Yahaya, J., Deraman, A., Hamdan, A.R.**: Continuously ensuring quality through software product certification: A case study. In: International Conference on Information Society (i-Society), pp. 183 - 188 (2010)

[YU 97] **Yu, E.:** Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: the 3rd IEEE Int. Symposium on Requirements Engineering (RE'97), pp. 226--235, USA (1997)

[ZHU 07] **Zhu, L., Gorton, I.:** UML Profiles for Design Decisions and Non-Functional Requirements. In: the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, pp. 8, USA (2007)

Standards References

[AUT 08] **Automotive SPICE**: Automotive SPICE, Process Assessment Model (PAM). Version 2.4, Status: Released 2008-08-01, Automotive SIG (2008)

[AUT 10a] **Automotive SIG:** Automotive SPICE, Process Assessment Model (PAM). Version 2.5, Status: Released 2010-05-10 (2010), URL: www.automotivespice.com

[AUT 10b] **Automotive SIG:** Automotive SPICE, Process Reference Model (PRM). Version 4.5, Status: Released 2010-05-10 (2010), URL: www.automotivespice.com

[AUT 12] **AUTOSAR**: AUTomotive Open System ARchitecture. URL: http://www.autosar.org/

[CEN 11] **European Committee for Electrotechnical Standardization:** BS EN 50159:2001 Railway Applications – Communication, Signalling and Processing Systems. CENELEC (2011)

[CMM 02] **Capability Maturity Model Integration for Software Engineering: (CMMI).** Version 1.1, CMU/SEI-2002-TR-028, ESC-TR-2002-028. Software Engineering Institute: Carnegie Mellon University, Pittsburgh, PA (2002)

[HIS 08] **HIS ReqIF:** Specification Requirements Interchange Format (RIF). Version 1.2 (2008)

[HIS 12] **HIS automotive SPICE**. URL: http://www.automotive-his.de/

[IEC 10] **International Electrotechnical Commission:** IEC 61508 (all parts), Functional safety of electrical/electronic/programmable electronic safety-related systems. Status: Release (2010)

[IEC 11] **International Organization for Standardization**: ISO/IEC International standard 15504 (all parts), Information technology – Process assessment. Status: Release 2004 -2011 (2011)

[INC 07] **International Council On Systems Engineering (INCOSE):** Systems Engineering Handbook. Version 3.1 (2007)

[INC 98] **INCOSE Measurement Working Group**: MWG, Systems Engineering Measurement Primer (1998)

[ISO 05] **International Organization for Standardization:** ISO/IEC 25000:2005 – « Ingénierie du logiciel -- Exigences de qualité du produit logiciel et évaluation (SQuaRE) ». Edition 1 (2005)

[IS0 07a] **International Organization for Standardization**: ISO/IEC 24744, Software Engineering — Metamodel for Development Methodologies, SEMDM (2007)

[ISO 07b] **International Organization for Standardization:** ISO /IEC 9126 - Software engineering -- Product quality. Edition 1, Status published (2007)

[ISO 07c] **International Organization for Standardization:** ISO/IEC 25030:2007 - Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Quality requirements. Edition 1, Status published (2007)

[ISO 07d] **International Organization for Standardization:** ISO/IEC 15939 - Systems and software engineering — Measurement process (2007)

[ISO 08] **International Organization for Standardization:** ISO/IEC 12207 Annex F and H of AMD1: 2008 and ISO/IEC 12207 AMD2: 2008 (2008)

[ISO 10] **International Organization for Standardization**: ISO/IEC CD 29148 - Software and Systems Engineering – Life Cycle Processes – Requirements Engineering (2010)

[ISO 11] **International Organization for Standardization:** ISO International standard IS026262 (all parts) Road vehicles –Functional safety. Status: First Edition 2011-11-15 (2011)

[OMG 08] **OMG:** OMG Software & Systems Process Engineering MetaModel (SPEM). Version 2.0, OMG document number: formal/ 2008-04-01 (2008)

[OMG 09a] **OMG:** OMG Business Process Model and notation (BPMN). Version 1.2, OMG document number: formal/ 2009-01-03 (2009)

[OMG 09b] **OMG**: Business Process Model and Notation (BPMN). FTF Beta 1 for Version 2.0. OMG Document Number: dtc/2009-08-14 (2009)

[OMG 10a] **OMG**: OMG System Modeling Language (OMG SysML). Version 1.2, OMG document number: formal/ 2010-06-01 (2010)

[OMG 11a] **OMG:** OMG Unified Modeling Language™ (OMG UML), Superstructure. Version 2.4.1, OMG document number: formal/2011-08-06 (2011)

[OMG 11b] **OMG:** Unified Modeling Language™ (OMG UML), Infrastructure. Version 2.4.1, OMG document number: formal/2011-08-05 (2011)

[OMG 11c] **OMG**: Requirements Interchange Format (ReqIF). Version 1.0.1, OMG document number: formal/2011-04-02 (2011)

[OMG 11d] **OMG**: UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE). Version 1.1, OMG Document Number: formal/2011-06-02 (2011)

Project References

Our solution draws inspiration from and inspired some research projects which address the issues in our scope in a complementary manner. These include the CESAR project (Cost-Efficient methods and processes for SAfety Relevant embedded systems) involving several stakeholders of different transportation fields (automotive, avionics, aerospace, rail, automatic), the SASHA project (Safety Check of Automotive SoftWare & Hardware) and the ATESST2 project which focuses more on the automotive domain.

CESAR Project

The CESAR project addresses industrial needs in many areas for embedded systems, especially for safety critical applications, developing ultra-safe embedded components, which can be used in an extremely competitive global environment, and which require significant cost reductions. These components are necessary for both safety and mobility, as well as producing green products targeted by the European Community.
If the certification rules and industry practices are specific to each sector, the scientific and technical basis of the component-based approach are common to all, and thus it is possible to develop innovative cross-cutting tools. The project will bring significant and conclusive innovations in the two most improvable systems engineering disciplines:

- Requirements engineering, in particular through formalization of multi-viewpoint, multi-criteria and multi-level requirements, in order to favor interchange in the development and supply chain and to ease the definition and identification of safety critical requirements in line with safety standards. Supported by traceability mechanisms, verification and validation of requirements shall be performed through improved techniques and methods.
- Component based engineering applied to design space exploration comprising multi-view, multi-criteria and multi-level architecture trade-offs.

In addition, CESAR intends to provide industrial companies with a breakthrough in system development by deploying a customizable systems engineering ”Reference Technology Platform" (RTP) making it possible to integrate or interoperate existing or emerging available technologies. This will be a significant step forward in terms of industrial performance improvement that will help to establish de-facto standards and contribute to the standardization effort from a European perspective. Benefiting from these multi-domain view points, CESAR addresses safety aspects of transportation and other societal mobility and environmental demands.

SASHA Project

The SASHA project addresses the safety of an automotive system, which is subject to the application of the ISO26262 standard. The objective is to control the quality and dependability of embedded systems. It addresses the architectures, the safety, the component standardization through the ISO26262 standard and the methods and tools. The project will therefore help:

- To establish a shared base of expertise and knowledge between upstream and downstream stakeholders of the V-cycle [MCD 84, INC 07]
- To share the results of safety analyzes
- To define and deliver the result safety case of the application
- To develop a methodological and tooled basis for reusing safety artifacts results integrated via the ISO26262.

The SASHA project therefore aims to structurally consolidate the two sides of the V-cycle given the methods and tools for design / development, taking into account the new requirements of ISO26262. SASHA relies on a continuous chain based on the contribution of the capital gain of each actor in the chain (in the bottom-up from the silicon supplier to the OEM as top-down from the requirement of the OEM to the components manufacturer). In this context, the project aims to support the industry with a tool and methodology that can significantly reduce expenses related to the introduction of ISO26262. This approach amounts to the development of ”templates"

negotiated between each actor that describe the safety attributes of a component until the design of embedded systems, and their final integration into the vehicle. The challenge for the automotive industry is to control these computer-electronic-mechanical embedded architectures.)
In the CESAR project, we are particularly interested in the proposal supporting the certification process through many different technical innovations of requirements specification and traceability by adapting a state of the art of model-based engineering domain and the process definition approach. Indeed, this proposal covers our needs concerning three main aspects:

- It allows integration of processes and requirements defined by a standard in the specific (product) requirements of a project
- It allows verification of compliance with the process indicating what is required in a standard
- It allows what has been done to be checked on the basis of evidence given by the product results against standards recommendations and user needs.

The immersive reading of the ISO2626 standard realized in the SASHA project which led to the detailed description of safety artifacts through a representation model on a conventional V-cycle process, following the scheduling of tasks as specified by the standard, give a basis for the thesis.

ATESST2 Project

The ATESST2 contributes to bridging the gap between cooperative systems and enabling design and verification technologies. The basis of the project is the architecture description language EAST-ADL, developed in past projects such as EAST-EEA (where the language was initiated), ATESST, TIMMO and EDONA projects. The language provides an information structure for the engineering information involved in automotive software development and ontology that makes the development of stand-alone automotive embedded systems more systematic and predictable.
In ATESST2, the EAST-ADL modeling approach is extended and new results are provided to support development and V&V of cooperative active safety systems. These results include:

- An architecture description language with improved means for capturing the requirements, characteristics and configurations of cooperative systems and the related analysis and V&V.
- Methodology and guidelines supporting language/tool adoption and cost-efficient development and V&V, and
- Harmonization of EAST-ADL with relevant standards including AUTOSAR and SysML.

The model-based development and V&V approach to be developed in ATESST2 contributes to improving communication among system stakeholders, documentation, and V&V capabilities. This is a shift from today's document-driven testing and simulation procedures, to a model-based way of working. This provides means for stakeholders to deal with the complexity and risk management of cooperative active safety systems.

Deliverables referenced

[ATT 10a] **EU Project ATESST**: EAST-ADL2 profile Specification, the ATESST Architecture Description Language. Version 2.1 (2010)

[ATT 10b] **EU Project ATESST**: Refined EAST-ADL2 Tool Support, deliverable 3.2, version number 1.0, (2010)

[CES 09] **EU Project CESAR**: Cost-efficient methods and processes for safety relevant embedded systems. ARTEMIS JOINT UNDERTAKING (JU), URL: http://www.cesarproject.eu/ (2009)

[CES 10a] **EU Project CESAR:** Requirement/Traceability Management. D_SP2_R3.2_M2 vol 3, Version 0.002 (2010)

[CES 10b] **EU Project CESAR:** Definition and exemplification of RSL and RMM. D_SP2_R2.2_M2, Version 1.0. URL: http://www.cesarproject.eu (2010)

[CES 10c] **EU Project CESAR:** Requirements Engineering State of the Practice Survey. D_SP2_R1.1_M1_SOP, (2010)

[SAS 09] **French Project SASHA:** URL: http://www.pole-moveo.org/documents/CPFUI8ANR2009.pdf