



HAL
open science

Multi-Faceted Modelling of Multi-Energy Systems : Stakeholders Coordination

Elmehdi Azzouzi

► **To cite this version:**

Elmehdi Azzouzi. Multi-Faceted Modelling of Multi-Energy Systems : Stakeholders Coordination. Chemical and Process Engineering. Université Paris-Saclay, 2021. English. NNT : 2021UPAST002 . tel-03166509

HAL Id: tel-03166509

<https://theses.hal.science/tel-03166509>

Submitted on 11 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multi-faceted modeling of multi-energy systems: Stakeholders coordination

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 573 , interfaces : approches
interdisciplinaires, fondements, applications et innovation
(Interfaces)

Spécialité de doctorat: Ingénierie des systèmes complexes
Unité de recherche: Groupe ISAE, SUPMECA, Laboratoire Quartz,
93400, Saint-Ouen, France
Réfèrent: CentraleSupélec

**Thèse présentée et soutenue à Saint-Ouen, le 21 janvier 2021,
par**

Elmehdi AZZOUZI

Composition du jury:

Jean-Michel Bruel Professeur à l'Université de Toulouse	Président & Rapporteur
Pierre De Saqui Sannes Professeur des Universités à l'école ISAE SUPAERO	Rapporteur & Examineur
Claude Baron Professeur des Universités à l'école INSA de Toulouse	Examinatrice
Olivia Penas Ingénieure de recherche à l'école Supméca Paris, groupe ISAE	Examinatrice
Frédéric Kratz Professeur à l'école INSA Centre Val de Loire	Examineur
Vincent Chapurlat Professeur des Universités à l'école des Mines d'Alès	Examineur
Jean-Yves Choley Professeur à l'école Supméca Paris, groupe ISAE	Directeur
Faïda Mhenni Maître de conférence à l'école Supméca Paris, groupe ISAE	Coencadrante & Examinatrice
Audrey Jardin Ingénieur chercheur à EDF R&D	Encadrante industrielle
Daniel Bouskela Ingénieur chercheur senior à EDF R&D	Encadrant industriel

Titre: Modélisation multifacettes des systèmes multi-énergies : coordination des parties prenantes

Mots clés: Méthodes formelles, Exigences formelles, Systèmes multi-énergies, Ingénierie système, Coordination de parties prenantes

Résumé: L'objectif de cette thèse est d'introduire une nouvelle méthodologie de co-conception basée sur l'ingénierie système qui place les parties prenantes au centre du processus de conception en supposant que les systèmes multi-énergies seront correctement conçus et exploités seulement si toutes les parties prenantes parviennent à des accords mutuels qui satisfont leurs objectifs intrinsèques respectifs. Ces accords émergent progressivement à travers un processus de négociation itératif entre les différentes parties et sont formalisés à l'aide de "contrats formels" qui peuvent être simulés pour faire de la vérification et de la validation tout au long du processus de conception et avant la signature de tout engagement. Les parties prenantes seront prêtes à signer des contrats dès lors qu'elles seront sûres que ces derniers remplissent leurs objectifs et qu'elles peuvent s'engager à respecter leurs obligations telles qu'elles sont énoncées dans les contrats. Des modèles sont utilisés pour structurer les relations entre les parties prenantes à tous les stades du cycle de vie des systèmes : les contrats formels sont dérivés d'objectifs de haut niveau et sont vérifiés à l'aide de "jumeaux numériques" qui sont composés de modèles d'exigences et de modèles comportementaux.

Title: Multi-faceted modeling of multi-energy systems : Stakeholders coordination

Keywords: Formal modeling, Formal requirements, Multi-Energy Systems, Systems Engineering, Stakeholders' coordination

Abstract: The purpose of this thesis is to introduce a new co-design Systems Engineering (SE) methodology that puts stakeholders at the center of the design process by assuming that the ME-CPS can be correctly designed and operated if all stakeholders reach mutual agreements that satisfy their intrinsic objectives. These agreements progressively emerge throughout an iterative negotiation process between stakeholders and are formalized using "formal contracts" which can be simulated for verification and validation all along the development process and before signature. Stakeholders will agree to sign contracts when they are sure that contracts fulfill their objectives and that they can commit to their obligations as specified in the contracts. Models are used to structure the relationships between stakeholders at all stages of the system's lifecycle: formal contracts are derived from high-level goals and are verified using "digital twins" that are composed of requirement models and behavioral models.

Acknowledgements

Throughout my thesis, I have received a great deal of support from people who are precious to me. I would like to first dedicate this thesis to my beloved parents for their lifelong support, wise counsel and unconditional love.

I would like to thank my industrial supervisors Audrey Jardin and Daniel Bouskela whose expertise was invaluable for the successful achievement of my thesis. Your relevant feedback pushed me to think out of the box and brought my ideas to a higher level. I really enjoyed working with you throughout these last three years

I would like to acknowledge my academic superior Faïda Mhenni and my thesis director Jean-Yves Choley for always being present to support me; this thesis could have never succeeded without your precious contribution. You were always there to guide me and encourage me throughout this difficult but amazing journey.

I would like to thank Jean-Michel Bruel and Pierre De-Saqui-Sanne for reviewing my thesis manuscript and for their interesting feedback and reports. I am also grateful to Olivia Penas, Frederic Kratz, Vincent Chapurlat, Claude Baron for agreeing to be examiners of my thesis. I have appreciated your relevant comments and observations. A special thanks to Christel Campagnon, Messaoud Bouned, Islem Djerad and Suzanne Thuron for the organization of the PhD defense. I am grateful to all the members of the SIMSE and MODULO projects Thuy Nguyen, Jean-Philippe Tavella, Dominique Croteau, Jean-Luis Bouvier, Clément Flinois and Luis Corona Mesa-Moles who have agreed to collaborate with me to put into practice the results of my research. Your support and constructive feedback have allowed enriching and refining my work to be suited to address the real industrial challenges facing the design of complex energy systems. I would like to thank all my sisters Zineb and Oumayma and all my family for always being there for me.

Finally, I could not have completed this adventure without the support of my friends Laure, Asmae, Othmane, Brahim, Oualid, Marouane that provided happy distractions to rest my mind outside of my research.

Synthèse en français

La prise de conscience de l’empreinte écologique entraîne une modification profonde de l’architecture du système énergétique en accordant la priorité aux énergies renouvelables qui occupent une partie croissante du marché, et en allant vers des systèmes locaux de gestion intelligente de l’énergie. En outre, la déréglementation du secteur de l’énergie en Europe a conduit à l’augmentation du nombre d’acteurs actifs sur le réseau énergétique, en concurrence avec les unités de production centralisées traditionnelles (nucléaire, combustible fossile, hydraulique...). Cependant, dans la pratique, les différents actifs qui composent ces systèmes sont exploités par des entités indépendantes ayant des objectifs et des perspectives différents.

L’objectif scientifique majeur de cette thèse est de proposer une approche formelle permettant de concevoir les grands systèmes énergétiques qui incluent un grand nombre de parties prenantes ayant des intérêts différents.

Pour ce faire, j’ai développé dans le cadre de cette thèse une méthodologie de co-conception basée sur l’ingénierie systèmes nommée CReMA en référence à “Common Requirement engineering Modeling Approach”. Cette méthodologie place les parties prenantes au centre du processus de conception en supposant que les systèmes multi-énergies seront correctement conçus et exploités seulement si toutes les parties prenantes parviennent à des accords mutuels qui satisfont leurs objectifs intrinsèques. Ces accords permettront de cadrer les interactions entre les parties prenantes à travers les obligations de chacun qui seront formalisés à l’aide de “contrats formels”.

La méthodologie CReMA a la particularité de proposer une vision **décentralisée** pour la coordination des parties prenantes vu qu’elle considère que chaque acteur a ses propres intérêts qui peuvent être de différentes natures telles que des intérêts financiers, écologiques, politiques ou bien un besoin en termes de ressources matérielles ou immatérielles. CReMA considère également que chaque partie prenante a des acquis qu’elle va mettre en valeur afin d’atteindre ses objectifs. Ces acquis peuvent également être de différentes natures (matérielles ou immatérielles). Ainsi, des interactions émergent entre les différentes parties prenantes tout au long du cycle de vie d’un système de systèmes, dans lequel des négociations ont lieu pour identifier des accords communs. Ces accords sont structurés et formalisés à travers les différents niveaux de la méthodologie CReMA. La méthodologie CReMA est

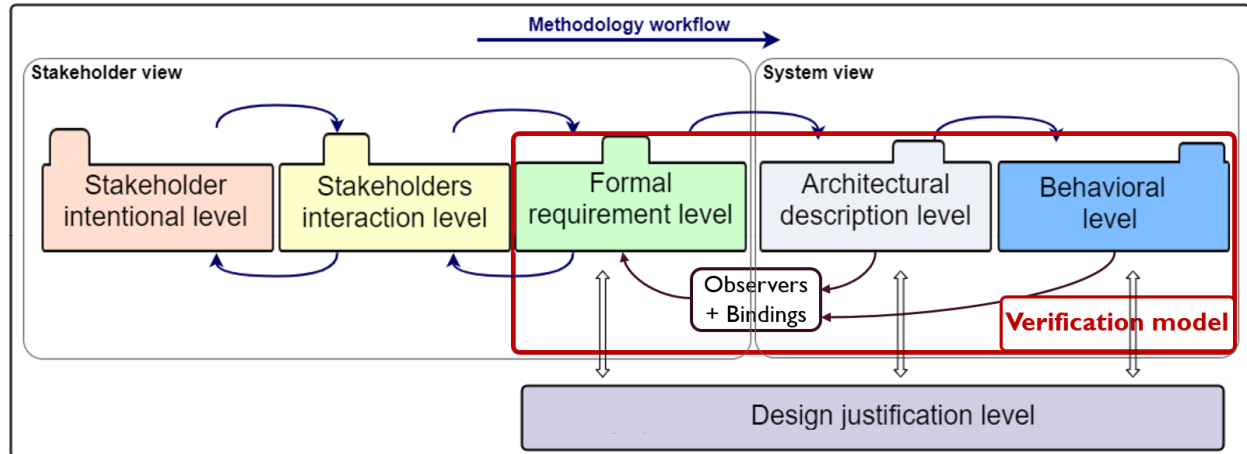


Figure 0-1: Niveaux de la méthodologie CReMA

construite sur la base de cinq niveaux de modélisation comme illustré dans la figure 0-1, et qui sont séparés en deux vues principales : la vue “parties prenantes” et la vue “système”. Un niveau supplémentaire, appelé niveau de “justification” de la conception s’étend sur les deux vues et sert de passerelle entre les deux vues pour assurer la traçabilité.

La vue “parties prenantes” vise à capturer les aspects sociaux à travers trois niveaux : (i) le niveau intentionnel qui s’intéresse aux objectifs de haut niveau des parties prenantes et aux dépendances entre elles (ii) le niveau d’interaction qui capture les interactions entre les parties prenantes telles que des échanges financiers, des échanges physiques ou bien des échanges de données, et (iii) le niveau de formalisation d’exigences qui vise à formaliser d’une part, les objectifs de haut niveau des parties prenantes, donnant lieu à des indicateurs de performances, et d’autre part, les exigences liées aux dépendances entre les parties, ce qui donne lieu à des contrats formels. Le principal atout de cette vue “parties prenantes” est de donner aux différentes parties les moyens d’exprimer et affiner leurs intérêts d’une manière rigoureuse, et d’identifier les relations avec les autres parties qui leur permettront de satisfaire leurs intérêts.

La vue “système” vise à modéliser la conception physiques et techniques des systèmes qui sont sous la responsabilité de chaque partie prenante tout au long de leur cycle de vie. Deux niveaux méthodologiques synthétisent ces aspects de conception : (i) le niveau de description de l’architecture qui permet de capturer une vision statique des solutions de conception, et (ii) le niveau comportemental qui permet de capturer le comportement dynamique de ces solutions. Ces deux niveaux sont tous les deux nécessaires pour faire de la vérification et validation formelle des systèmes. La méthodologie CReMA donne la liberté à ses utilisateurs d’adopter les outils de modélisation qui sont adaptés au domaine d’application concerné.

Afin de créer des modèles de vérification et validation formelles qui permettrons d’assister

les parties prenantes lors de leur prise de décision, la méthodologie CReMA utilise deux concepts nommés “Bindings” et “Observers” [63] (voir Figure 0-1) qui font office de capteurs virtuels qui observent les variables physiques des modèles de comportement et d’architecture, effectuent des transformations dessus et donnent en sortie des données fonctionnelles qui sont adaptées aux modèles d’exigences. Ainsi, chaque partie prenante peut effectuer des vérifications formelles concernant la satisfaction de ses objectifs et également de ses obligations vis-à-vis des autres parties.

La méthodologie CReMA a été appliquée dans le cadre d’un projet de recherche industrielle au sein d’EDF R&D afin de coordonner des parties prenantes pour la rénovation du système énergétique d’un quartier en région parisienne. L’objectif du projet est de rendre ce quartier "plus vert" et "plus intelligent", plus vert parce qu’on vise à intégrer davantage d’énergie renouvelable, et plus intelligent parce qu’on vise à avoir un système de gestion de l’énergie qui est plus efficace.

Cette application a permis de prouver la pertinence de la méthodologie CReMA qui a conduit les parties prenantes du projet à mener diverses réflexions et négociations. Ceci a permis de d’identifier et converger vers des accords communs. Des conclusions précieuses ont été tirées sur la satisfaction des objectifs et des obligations des parties prenantes à chacun des scénarios de rénovation qui a été simulé.

Contents

List of Figures	xi
List of Tables	xv
List of Abbreviations	xvii
Chapter 1 Introduction	1
1.1 Context	1
1.2 Complex systems – an introduction	2
1.2.1 Introduction to CPS, S-CPS, SoS	3
1.2.2 Focus on Multi-Energy Systems (MES)	5
1.3 Work objectives	9
1.4 Scientific challenges	10
1.5 Research background	11
1.6 Our contribution	12
1.7 Manuscript organization	14
Chapter 2 Complex systems design - a state of the art analysis	15
2.1 An overview of Systems Engineering	15
2.1.1 SE history	15
2.1.2 Introduction to SE	16
2.2 Stakeholders and stakeholders’ coordination	17
2.3 i* (star) framework	19
2.3.1 Strategic Dependency (SD) model	19
2.3.2 Strategic Rationale (SR) model	21
2.4 Requirement engineering (RE)	22
2.4.1 Characteristics of good requirements	23
2.4.2 Requirements classification	24
2.4.3 Requirements modeling approaches	24
2.5 Verification and Validation (V&V)	30
2.5.1 Definitions	30

2.5.2	Methods for V&V	32
2.5.3	V&V and requirement modeling languages and tools - Summary	34
2.6	Focus on justification frameworks and traceability	35
2.6.1	Traceability	36
2.6.2	Claim - Argument - Evidence (CAE) - justification framework	37
2.7	Systems engineering methodologies for complex systems	41
2.7.1	Overview	41
2.7.2	Comparison and evaluation of existing methodologies	42
2.7.3	Summary and discussions	52
Chapter 3 Complex systems design – a new methodology		57
3.1	Methodology objectives	57
3.2	Methodology assumptions	58
3.3	Previous work : Co-design verification approach	59
3.4	Methodology formal objects and concepts	60
3.4.1	Stakeholder	60
3.4.2	Requirements and properties	63
3.4.3	Goals and Key Performance Indicators (KPIs)	64
3.4.4	Contract (assumption and guarantee)	65
3.4.5	The implementation	69
3.4.6	Architecture and architecture description	69
3.4.7	Observation model	70
3.4.8	Binding	71
3.4.9	Test scenarios	72
3.4.10	Refinement	72
3.5	CReMA methodology - Global overview	73
3.5.1	Overview on CReMA methodology levels	73
3.5.2	Methodology phases	77
3.6	Methodology metamodel	77
3.6.1	Definition of a metamodel	77
3.6.2	CReMA methodology metamodel	78
3.7	Methodology levels	81
3.7.1	Stakeholder intentional level	82
3.7.2	Stakeholder interaction level	86
3.7.3	Formal requirement level	90
3.7.4	Design justification level	98
3.7.5	Architecture description level	102
3.7.6	Behavioral level	106

3.7.7	Modular construction of the verification model	109
3.8	Summary	112
Chapter 4	Validation of the methodology - the PowerGrid case study	115
4.1	PowerGrid story board	116
4.1.1	Overview	116
4.1.2	PowerGrid development phases	117
4.2	Phase 0: PowerGrid urban energy planning	118
4.2.1	Stakeholders, objectives and tested scenarios	118
4.2.2	Stakeholder intentional level	119
4.2.3	Stakeholder interaction level	119
4.2.4	Formal requirement level	119
4.2.5	Design justification level	121
4.2.6	Architecture description level	121
4.2.7	Behavioral level	123
4.2.8	Verification model	125
4.2.9	Results	125
4.2.10	Summary	127
4.3	Phase 1: PowerGrid dynamic sizing	128
4.3.1	Stakeholders, objectives and tested scenarios	128
4.3.2	Stakeholder intentional level - Strategic rationale models	129
4.3.3	Stakeholder intentional level - Strategic dependency models	136
4.3.4	Stakeholder interaction level	144
4.3.5	Formal requirement level	145
4.3.6	Design justification level	154
4.3.7	Architecture description level	163
4.3.8	Behavioral level	165
4.3.9	Observation models and bindings	167
4.3.10	Verification models	174
4.4	Summary	188
Chapter 5	Conclusion and future works	191
5.1	Overview	191
5.2	Contributions	193
5.2.1	Limits	195
5.3	Methodology application	196
5.4	Future works	197
5.4.1	Potential enhancements concerning CReMA methodology	197

5.4.2	Potential enhancements concerning the CReMA methodology application	198
5.5	In a nutshell	199
	Bibliography	200
	Appendix A: Model checking	211
A.1	Characteristics of Model Checking	211
A.2	Model Checking processes	211
A.3	Comparison between temporal logic languages	213
A.4	Strengths and weaknesses of model checking	215
	Appendix B: Systems Engineering methodologies	217
B.1	Comparison criteria of SE methodologies	217
B.2	Evaluation of SE methodologies	219
B.2.1	An MBSE Approach to Pass from Requirements to Functional Architecture	219
B.2.2	A Model-Based Design Methodology for Cyber-Physical Systems	222
B.2.3	OpenMETA / CyPhy	224
B.2.4	INTO-CPS	225
B.2.5	UnCoVerCPS	226
	Appendix C: A formal contract based framework	231
	Appendix D: Formal properties in ReqSysPro	233
	Appendix E: Observation models and bindings	241

List of Figures

0-1	Niveaux de la méthodologie CReMA	iv
1-1	Cyber-Physical Systems elements inspired from [70]	4
2-1	Systems Engineering history from [18]	16
2-2	Dependency types of the i* framework	20
2-3	Example of a strategic dependency model from the health care domain (from [73])	21
2-4	Example of a strategic rationale model from the health care domain (from [73])	22
2-5	Example of a STIMULUS Requirement (from [89])	28
2-6	Example of FORM-L requirement	30
2-7	Basic graphical elements of CAE (from [116])	39
2-8	Basic graphical elements of CAE (from [116])	39
2-9	RFLP V-model and tools used for each view	44
2-10	SLIM methodology framework (from[50])	48
2-11	B method for software design	49
2-12	Modeling techniques used at different levels of system maturity	54
3-1	Verification model (from [63])	59
3-2	Abstract overview on CReMA methodology levels	60
3-3	Stakeholder description model including five bricks	62
3-4	FORM-L contract (from [113])	66
3-5	Example of contract composition including three stakeholders	67
3-6	Positioning the concepts of observation model and bindings in the construction of the verification model [63]	70
3-7	Methodology levels	74
3-8	CReMA methodology levels with underlying techniques	76
3-9	Global metamodel of CReMA methodology	79
3-10	Metamodel of the stakeholder intentional level	83
3-11	Intentional level	84
3-12	Metamodel of the stakeholder interaction level	87

3-13 Stakeholders Interaction level	88
3-14 Metamodel of the formal requirement level	91
3-15 Formal requirements level	92
3-16 Example of a requirement expressed in FORM-L language	94
3-17 Example of a requirement built using ReqSysPro library (corresponds to the requirement in Figure 3-16)	96
3-18 Basic graphical elements of CAE (from [116])	100
3-19 Metamodel of the architecture description level	103
3-20 Architectural description level	104
3-21 Metamodel of the behavioral level	107
3-22 Behavioral level	108
3-23 Modular verification model	111
4-1 Observing goal change in the strategic rationale model of the town council .	119
4-2 Modeling of kpi1 in FORM-L (from [60])	120
4-3 PowerGrid electric architecture	121
4-4 PowerGrid heat architecture	122
4-5 PowerGrid economic architecture	122
4-6 Main interface of the EnergyLogic PowerGrid Demonstrator Tool (from [74])	124
4-7 Evaluation of the town council’s kpi1 and kpi4 (made by EIFER)	126
4-8 Evaluation of the town council’s kpi2 and kpi3 (made by EIFER)	127
4-9 Consumers Strategic Rationale Model	129
4-10 Producers Strategic Rationale Model	130
4-11 E-DSO Strategic Rationale Model	131
4-12 Strategic Rationale Model of NeighbourPower as the H-DSO	133
4-13 Strategic Rationale Model of NeighbourPower as a supplier	134
4-14 Strategic Rationale Model of NeighbourPower as a balance manager	135
4-15 TSO Strategic Rationale Model	135
4-16 Consumers - H-DSO dependency model	136
4-17 Consumers – Supplier dependency model	136
4-18 Producers - H-DSO dependency model	137
4-19 Producers - Supplier dependency model	138
4-20 E-DSO - Consumers dependency model	138
4-21 Producers - E-DSO dependency model	139
4-22 E-DSO - Supplier dependency model	139
4-23 TSO - E-DSO dependency model	140
4-24 Balance manager - E-DSO dependency model	140
4-25 Balance manager - TSO dependency model	141

4-26	Balance manager - supplier dependency model	141
4-27	Supplier - H-DSO dependency model	142
4-28	Global SD model	143
4-29	Stakeholders global interaction model	144
4-30	Consumers KPI_1 written in FORM-L	148
4-31	KPI_1 represented using ReqSysPro blocks in Modelica	149
4-32	Global contract model	151
4-33	E-DSO - Consumers dependency model	152
4-34	E-DSO subcontract written in FORM-L language	155
4-35	absoluteLimit requirement translated into Modelica using using ReqSysPro .	156
4-36	CAE structure of the justification of the safe distribution of electricity to consumers	157
4-37	CAE structure of the justification of the reliability of a design hypothesis . .	159
4-38	CAE structure of the justification of the reasonable simulation time of the E-DSO model	160
4-39	PowerGrid electric architecture	163
4-40	PowerGrid heat architecture	164
4-41	Parameters characterizing the architecture of the PowerGrid buildings	165
4-42	PowerGrid economic architecture	166
4-43	Behavioral model of the 20 buildings [75]	167
4-44	Partial overview of the distribution grid's behavioral model for the 20 buildings [75]	168
4-45	The observation model for KPI_1 and KPI_2 written using FORM-L language	170
4-46	Modelica diagram view of the observation model for KPI_1 and KPI_2	171
4-47	Modelica text view of the observation model for KPI_1 and KPI_2	172
4-48	The elements composing the consumers description model with the bindings connecting them	173
4-49	Consumers' verification model	176
4-50	Simulation results of one specific consumer guarantees	179
4-51	Simulation results of the KPIs of one specific consumer	180
4-52	Maximum power range overrun	181
4-53	EDSO's verification model	185
4-54	Maximum voltage range overrun	187
B-1	Core Control Context Diagram (from [101]) - Capella	220
B-2	Modes and transitions diagram (from [101]) - Capella	221
B-3	Illustration of the sequence diagram (from [101]) - Capella	222
D-1	Budget KPI represented using ReqSysPro blocks in Modelica	233

D-2	Green KPI represented using ReqSysPro blocks in Modelica	234
D-3	Local consumption KPI represented using ReqSysPro blocks in Modelica . .	234
D-4	The requirements corresponding to consumers guarantees and the E-DSO as- sumptions	235
D-5	The requirements corresponding to E-DSO guarantees and the consumers as- sumptions	236
D-6	AbsolutLimit requirement represented using ReqSysPro blocks in Modelica .	237
D-7	AverageLimit requirement represented using ReqSysPro blocks in Modelica .	237
D-8	AverageLimit3 requirement represented using ReqSysPro blocks in Modelica	238
D-9	AverageLimit3 requirement represented using ReqSysPro blocks in Modelica	238
D-10	ComplyMaxP requirement represented using ReqSysPro blocks in Modelica .	239
D-11	ComplyMaxQ requirement represented using ReqSysPro blocks in Modelica .	239
D-12	ComplyPQconstraint requirement represented using ReqSysPro blocks in Mod- elica	240
E-1	Modelica text view of the observation model for KPI_3 and KPI_4 (part 1) .	241
E-2	Modelica text view of the observation model for KPI_3 and KPI_4 (part 2) .	242
E-3	Modelica diagram view of the observation model for KPI_3 and KPI_4	243
E-4	Python code developed for the instantiation of the first section of the obser- vation model used for consumers KPIs evaluation (part1)	243
E-5	Python code developed for the instantiation of the first section of the obser- vation model used for consumers KPIs evaluation (part2)	244

List of Tables

2.1	Requirements classification according to the SEBoK	25
2.2	Summary table of requirements languages and tools from the state of the art	35
2.3	Summary table of methodologies from the state of the art	52
3.1	Graphical elements used in the intentional level	85
3.2	Graphical elements used by the Stakeholder Interaction Model (SIM)	89
3.3	Graphical elements used by the formal requirement level	93
3.4	Truth table of the logical conjunction (from [61])	95
3.5	Truth table of the logical negation (from [61])	96
3.6	Graphical elements used by CAE framework	100
3.7	Graphical elements used for building architecture description model	105
4.1	Relative errors comparing EPSL and PowerSysPro results at initialization time (from [134])	162
4.2	Comparing the number of unknowns/equations between EPSL and Power- SysPro on the PowerGrid demonstrator (from [134])	162
4.3	Summary of the simulation results of the consumers verification model . . .	178
4.4	Summary of the simulation results of the E-DSO verification model	185
B.1	Methodologies comparison criteria	218
B.2	Summary table of methodologies from the state-of-the-art	229

List of Abbreviations

AFIS	Association Française d'Ingénierie Système
ASCE	Assurance and Safety Case Environment
CAD	Computer-Aided Design
CAE	Claim - Argument - Evidence
CAPEX	CAPital EXpenditure
CBD	Contract Based Design
CFD	Computational Fluid Dynamics
CHP	Combined Heat and Power
CPS	Cyber Physical Systems
CReMA	Common Requirement engineering Modeling Approach
CRML	Common Requirement Modeling Language
CTL	Computation Tree Logic
DAE	Differential-Algebraic Equations
DHN	District Heat Network
DLM	Data Lifecycle Management
DS	Dassault Systemes
DSL	Domain-Specific Languages
EDF	Electricité De France
E-DSO	Electric Distribution System Operator
EIFER	European Institute for Energy Research
ETL	Extended Temporal Language
FEM	Finite Element Method
FMU	Functional Mockup Unit
FORM-L	Formal Requirements Modelling Language
FT	Fault Trees
GSN	Goal Structuring Notation
HCI	Human-Computer Interaction
H-DSO	Heat Distribution System Operator
HV	High Voltage
HVAC	Heating, Ventilation, and Air-Conditioning

I&C	Information and Control
INCOSE	International Council on Systems Engineering
IoT	Internet of Things
KBE	Knowledge-Based Engineering
KPI	Key Performance Indicator
LTL	Linear temporal logic
LV	Low Voltage
M&S	Modeling and Simulation
MBSE	Model-Based Systems Engineering
ME-CPS	Multi-Energy Cyber Physical Systems
MES	Multi-Energy Systems
MV	Medium Voltage
NCOSE	National Council on Systems Engineering
NEWTON	NExt Wave of smarT On-demand eNergy services based on efficiency and flexibility
NPSH	Net Positive Suction Head
OCL	Object Constraint Language
OME	Organization Modelling Environment
OMG	Object Management Group
OPEX	OPerational EXpenditure
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
PV	Photovoltaics
RE	Requirement Engineering
RFLP	Requirement, Functional, Logical, and Physical
SD	Strategic Dependency model
SDPD	System Driven Product Development
SE	Systems Engineering
SI	International System of units
SIM	Stakeholder Interaction Model
SoS	Systems Of Systems
SR	Strategic Rationale model
STL	Signal temporal logic
SUTD	System Used To Design
TSO	Transmission System Operator
V&V	Verification and Validation

Chapter 1

Introduction

1.1 Context

In a context of a shifting energy landscape, and due to the deregulation of the energy sector in Europe, more and more actors are being active on the energy grid in concurrence with the traditional centralized production units (nuclear, fossil fuel, hydro. . .). Besides, renewable energies have now a high priority in terms of future investments and are taking increasing parts of the energy market. Furthermore, electricity consumers are now given the possibility to become also electricity producers. The increase in the number of new players in an open power market results in short and long-term increasing uncertainties in terms of energy price fluctuation, increased grid instabilities generated by intermittent renewable energy sources, new regulations giving priority to renewables on the grid. Uncertainties are further amplified by the uncoordinated energy policies between European countries.

The infrastructure of the current electric grid was conceived decades ago for a stable national network with large production units and accurate predictions about electric demand and production with few uncertainties. This infrastructure was neither designed for the current energy transition with an energy mix featuring a high level of renewable, nor the future electric mobility both requiring a stronger and smarter grid architecture. A possible solution for increasing grid stability is to divide it into smaller grids of various sizes, from medium-size to small-size. Hence, future network architectures will likely be more decentralized with new organization levels going down as far as micro-grids.

To prepare at minimum cost the transition towards the future energy system, it is necessary to develop new prediction methods to understand the impact of various possible energy scenarios on future investments. To be credible, predictions should be based on a minimum set of assumptions and take into account all constraints and uncertainties having a significant impact on the energy system under study, those items constitute the so-called scenarios of the study. It is essential to assess different alternatives for a given scenario to choose the

best energy system architectures.

Hence, new methods and tools should particularly take into consideration the following points:

- The energy mix of Multi-Energy Systems (MES) combining different types of producers (e.g., nuclear, hydro, wind...) and consumers (e.g., industries, household...) interacting at various levels (e.g., country, city, district...) with optimal criteria that may differ from one stakeholder to another.
- Multiple stakeholders act on the system all along its lifecycle including end-users and customers that are willing to participate from a very early stage in the system design to establish, modify and verify requirements, and understand the outcomes of design choices at each step of the system's engineering lifecycle.
- The energy system is becoming more and more decentralized with individual management and partial independence, thus evolving towards a System of Systems (SoS).
- In order to minimize the impact of uncertainties and assumptions taken on the environment, it is essential to use validated models for the description of system architecture and behavior (such as physical models).

The current thesis comes in this context and aims at contributing to modeling tomorrow's energy system in which EDF has a significant stake. This work is expected to have a significant contribution to cost reduction of large infrastructure energy system projects (around 30%). It is expected that the results of this thesis will lead to the development of new tools for Model-Based Systems Engineering (MBSE) and their systematic use in engineering divisions where Systems Engineering (SE) is currently only partially deployed.

1.2 Complex systems – an introduction

The current trend of systems development can be summarized by the industrial revolution Industry 4.0 and the development of technologies such as the Internet of Things (IoT). Today's complex systems contain increasing numbers of elements evolving independently in their environment. The future is to assign some kind of intelligence to these independent elements so that they can efficiently cooperate to perform high-value missions. The objective is to have highly flexible and reliable systems dealing with the need to individualize services and to adapt to the constant changes in the environment while using environment-friendly resources. Typical examples of such systems are cities (or districts), transportation systems, energy systems, which can interact with each other. A typical high-value problem is how to double the size of a district without doubling energy consumption.

Intelligent systems are systems that can react efficiently when challenged by other interacting systems. Such systems induce massive inter-connections and interactions between elements at a cyber level using internet or other vectors for data exchange. At the same time, they induce major physical interactions between elements as well as social interactions between human actors who are systems stakeholders. The latter can be developers, operators, end-users, insurers, public authorities... that need to cooperate and without whom intelligent systems wouldn't have any added value.

New concepts have emerged to describe the assets of present and future complex systems. They are increasingly being cited in recent research works. Amongst them we can mention "Cyber-Physical-Systems", "Socio-Cyber-Physical Systems", "Systems of Systems", and even "Cyber-Physical Systems of Systems" [135].

Before starting to work on how to handle these complex systems, it is important to start by presenting the essential characteristics of these concepts and positioning our work with regard to them.

1.2.1 Introduction to CPS, S-CPS, SoS

1.2.1.1 Cyber-Physical Systems (CPS)

Cyber-Physical Systems are integrations of real-time computing and physical systems with tight interactions. Compared to traditional embedded systems, they have a stronger emphasis on interactions between the physical world and the computing world [95][100]. Processors control physical processes via sensors that provide feedback. CPS elements are equipped to various extent with processors, sensors, and communication technology that provide them with some kind of autonomous intelligence and the ability to communicate and cooperate.

Thus, the engineering of CPS gathers multiple classical disciplines such as mechanics, electronics, safety and dependability, system control, human factors, etc. The usual issue is how to coordinate efficiently the different disciplines that come with their own methodologies and tools which are in general not conceived to be interoperable across engineering domains.

1.2.1.2 Socio Cyber-Physical Systems (S-CPS)

Socio-Cyber-Physical Systems are the fusion of social and human factors with CPS [115]. The human factor plays a key role in S-CPS operation and is in charge of taking major decisions. S-CPS components have different levels of automatic control and autonomy, depending on the involvement of human intervention and control. The decision process is generally distributed among teams of operators and managers who make use of available decision supports. The way the system is managed and operated is the result of the interplay of the human actor and computerized systems and is influenced by many socio-technical

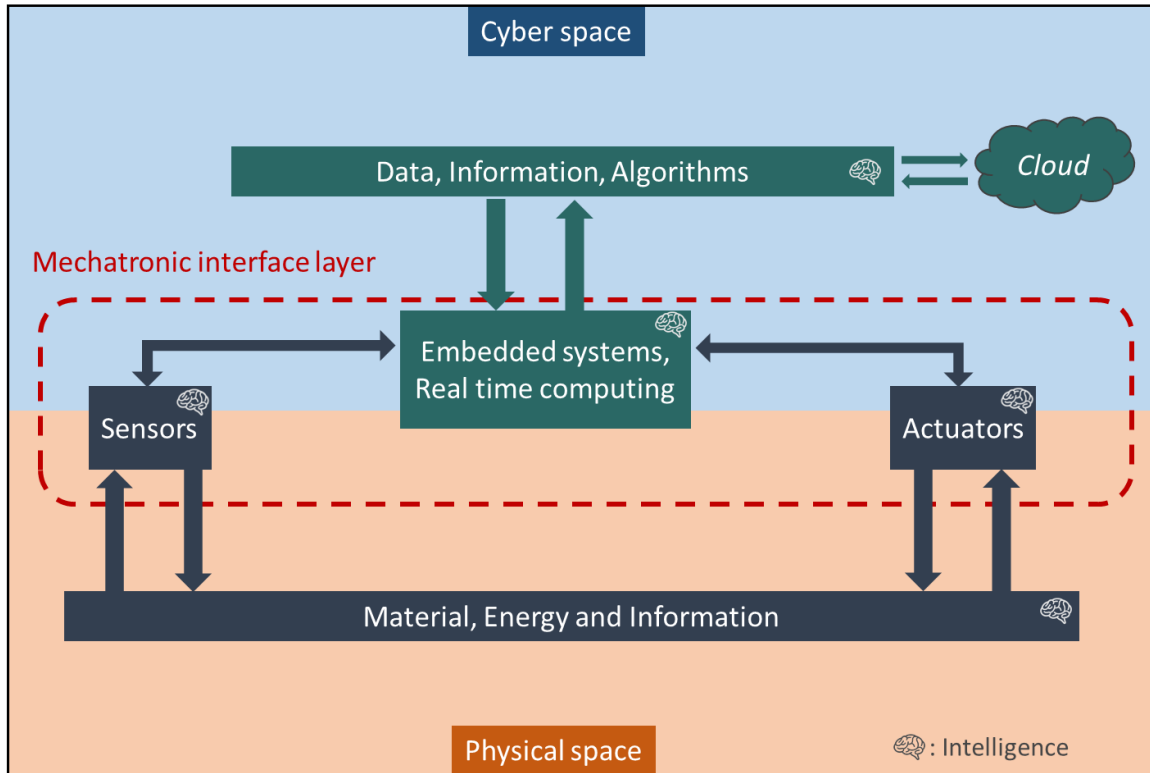


Figure 1-1: Cyber-Physical Systems elements inspired from [70]

factors. If a computer-based solution is not intuitive or not represented transparently, human tends to intervene and alter the decision suggested by a computer according to his experience. Thus, while being able to recognize problematic or abnormal situations early and reacting to them based on knowledge and experience, a human may also introduce an additional source of unpredictable and non-linear behavior in the system. The role of the human actor in the management and operation of complex systems and his interplay with computer-based decision support systems or optimization algorithms is an area that needs deeper investigation. Additionally, studies must be done concerning human awareness and reactions when facing different situations [135].

1.2.1.3 Systems Of Systems (SoS)

Systems of systems are an integration of systems that are partially independent in terms of managerial and operational processes. They are gathered to achieve a global function or a service that cannot be provided by a component individually, or cannot be optimally and efficiently provided being isolated [135]. According to Maier in [104], systems of systems are particularly characterized by:

- Operational independence of components: each system of an SoS has the ability to operate independently if it is separated from the other systems.
- Managerial independence of components: Components of Systems of systems are owned by separate entities and are managed independently and generally maintain a continuous operational existence.
- Geographical distribution: SoS are often geographically distributed with material or/and information connections.
- Emergent behavior: the emerging behavior of the overall system is not necessarily the expected goal of the system because there are always risks associated with unintended behaviors resulting from combining complex systems.
- Evolutionary development processes: large systems are developed to operate over long periods and are continuously improved and adapted over the system lifecycle. More detailed descriptions of SoS are presented in [135], [104] and Annex G of the standard ISO/IEC/IEEE 15288 [42].

1.2.2 Focus on Multi-Energy Systems (MES)

1.2.2.1 MES overview

Multi-energy systems (MES) can be defined as the combination of diverse production units (renewable or non-renewable) with possible storage capacities, transport and distribution systems, telecommunication and so on, which interact together for the sake of optimal performances in terms of economic, technical, or environmental aspect. This is in contrast with the classical energy systems where sectors are operated separately or with only a few real-time interactions [105].

MES may incorporate different kinds of energy vectors such as electricity, heat, fuel, cooling, etc. in the same network and some of them can be produced by the same unit which is called a “co-generation unit”. Doing so leads to a significant increase in energy efficiency by:

- Coordinating a higher variety of production units that are subject to different operational constraints (e.g., startup and shutdown delays, number of cycles, minimum and maximum loads, fuel costs, electricity prices, pollutant emissions, wind, and solar resources...) and therefore exhibit different optimal conditions,
- Diminishing heat pollution (i.e. heat released to the cold source) by turning heat waste into a valuable product,

- Allowing consumers to participate in the stabilization of the grid by interacting with production units, by being themselves local energy producers, and by participating in consumption cut-off programs.

Therefore, MES can be seen as an extension of the classical energy mix towards a higher variety of means to ensure grid stability while promoting smarter and greener energy consumption.

1.2.2.2 MES evolution

Until recently, energy systems were designed for stable energy needs at all levels up to the national scope. Centralized production units were sized and operated on that basis in order to meet the energy demand at all times. From this perspective, uncertainties affecting the daily operation of plants could only come from climatic variations, but seasonal statistical predictions coupled to weather forecast allowed fairly good forecasts for the day-ahead planning, ensuring fairly good network stability. Obviously, this paradigm was adequate in terms of economic, technical, and management aspects. However, it is no longer adapted to new societal trends regarding environmental, economic, and political issues:

- The need to break away from the dependency of primary energy sources that are very irregularly distributed around the earth, strongly related to political aspects, limited in terms of available resources and not ecologically friendly.
- The deregulation of the energy sector in Europe, with more active actors on the energy grid in competition with the existing centralized production units.
- The necessity to integrate a larger share of renewable to reach a higher percentage of renewable energy production (or consumption) in line with national policies.
- The development of electric mobility (15 million vehicles in 2035 [43]).
- The necessity to ensure a secure and affordable energy supply.

These elements will have a huge impact on the electric grid that will need to be reinforced. For each scenario, it will be necessary to assess many alternatives. As an example, for its forecast on the possible evolution of the energy mix over the period 2025-2035, RTE (*“Réseau de Transport d’Electricité”*) identified 5 scenarios totaling 15,000 possible variants, of which it only analyzed 50. Over 1000 simulations were performed for each variant. This evolution concerns only the energy mix in the current grid architecture. One can then anticipate that the number of scenarios and variants to be analyzed will be much larger when considering modifications in the architecture of the energy system (e.g., the introduction of microgrids).

1.2.2.3 MES characteristics

MES involve multiple stakeholders. MES gather large numbers of stakeholders with different but complementary roles and views on the same system. Stakeholders act at different stages of the engineering lifecycle of a MES and have generally conflicting expectations from it. In regards to the same energy grid, investors have profitability objectives, whereas the grid operator ensures the grid stability and the electricity regulator guarantees equitable access to the grid to the various competing electricity producers and consumers. Thus, system stakeholders need dedicated methods allowing, at the same time, taking into consideration all points of view in a co-design framework, while enabling an autonomous design per discipline at different stages of the system lifecycle.

MES are long-lived and are subject to constant transformation. Large distributed MES are developed to operate continuously over long periods that can go up to decades for nuclear plants, and even eternity for the electric grid. Nobody ever thinks of dismantling the latter, only gradual maintenance and renovations operations are planned. These systems are therefore constantly subject to improvements and modifications at different stages of their lifecycle.

It is believed that MES are always constrained by the legacy energy system. Any so-called “new energy system” is, in fact, an evolution of the old one because energy supply can never be interrupted, even for very short periods of time. A MES can be considered as an ecosystem that is never rebuilt from scratch but is in constant transformation. MES differ from other complex systems with less stringent availability constraints (such as rail networks) and require a middle-out approach considering the legacy system instead of the usual top-down approach suitable when starting from scratch. Existing approaches are generally “top-down” starting with the definition of systems missions and requirements, followed by functional and logical decomposition, and finally components allocation. These paradigms are not well suited for the “transformation” of large MES with an important legacy side.

In order to continuously ensure the smooth transformation, MES need to be supported by a “digital twin” that will evolve throughout their whole lifetime. The digital twin here is not restricted to behavioral representation of the system, but it also refers to what is expected from it. At any step of a system lifecycle, the so-called digital twin will assist engineers to have a precise view on it and have the capacity to evaluate the impacts of transformations while testing a large panel of realistic scenarios. The main goal is to secure the acquired performances and values and to respond to the new emerging challenges.

MES exhibit high safety stakes. Energy systems have high safety stakes and several safety-critical components that are carefully and regularly inspected by safety authorities.

One of the biggest challenges facing the development of an energy system such as a nuclear plant is the constant obligation to demonstrate that safety-critical elements satisfy all safety requirements at each stage of their life cycle, including design, manufacture, operation and dismantling. The proof of the correct functioning is equally as important as the functioning itself. If the developer fails to provide rigorous justification and demonstration that safety requirements are met, the system will not be licensed to enter the operational phase or continue its operation. It is the case of the French nuclear reactor EPR which has felt behind its schedule and exceeded the costs originally estimated. Among other reasons was the difficulty to demonstrate that some safety requirements are respected. The digital twin mentioned above supported with adapted methods for modeling complex systems can have a major role to tackle this challenge. It provides rigorous means for justification and demonstration that can be based on deterministic or stochastic paradigms in order to prove that all system requirements and safety properties, in particular, are satisfied with the performances required by regulations.

A second challenge emerges in parallel with the necessity of justification and demonstration is the pressing need for productivity and the need to enter the system operational phase within limited time and cost. A typical example of failure related to this challenge is the Boeing 737 max. The need to put into the market an airplane with specific performances in no time have led engineers to underestimate a function that turned out to be safety-critical. They did use safe engines and a safe airplane structure for the 737 max, but the overall integration was not well-conceived. Not to mention that an important stakeholder which is safety authorities was left aside during the design.

In large complex systems, it is common that some demonstrations are done in parallel with the development of the physical system (such as the EPR where the manufacturing of the system is launched before the end of the design). However, it is essential to have rigorous means to make tenable V&V at early design steps in order to identify possible inconsistencies in the system before proceeding to its development.

MES complexity comes from their physical dimension. The evolution of MES sketches up a landscape with increasing communication, control, and information technologies into the existing physical world. The goal is to have “Smarter” systems that guarantee more flexibility, efficiency, reliability and security, and introduce new services and applications throughout a systematic embedding of cyber dimension within the physical world [85][76][103]. The idea is not to intensively implement modern sensors and actuators to have a large amount of data for energy systems operators, but the goal is rather to develop methodologies with the right integration between the cyber dimension and the physical world that represents the core of the complexity of Multi-Energy Cyber-Physical-Systems (ME-CPS)

[85]. Indeed, MES are essentially made up of multi-physical systems with complex processes in constant interaction exchanging energy and mass. One may cite physical disciplines such as electricity, heat, chemistry, mechanic, electronics, etc where each requires dedicated paradigms to be modeled and analyzed. Thus, one of the main challenges facing the optimal design of MES is the interoperability between these disciplines as such and further integration of the cyber dimension. New methods should afford means to integrate the “classical” aspects of energy systems with the new communication and computing cyber components.

MES are subject to environmental uncertainties. MES are “open” systems operating in environments that cannot be controlled or to a very low degree, and to which the behavior is strongly tied up and is highly sensitive to its change. The environment does not only refer to the natural context, but it also refers to the societal environment, political environment, regulations, etc.

During development phases, it is very complex or even impossible to precisely model the MES environment, which depends on a large number of parameters. The environment is generally represented using stochastic techniques which always have a significant part of uncertainties. One can never exactly predict the exact behavior of the weather, the climate changes, the market, regulations modifications, availability of resources, or even a pandemic such as COVID-19, etc. which all have a huge impact on MES.

Therefore, considering environmental aspects is a major concern for the development of future energy systems. The uncertainties should be analyzed with dedicated paradigms allowing to systematically evaluate large numbers of realistic scenarios and assess their impact on future MES.

1.2.2.4 Summary

To sum up, MES are not fundamentally different from the CPS, S-CPS, and SoS as they exhibit similar characteristics. However, MES have a strong emphasis on physical and societal aspects, the cyber dimension being progressively embedded in the present and the future.

This thesis will essentially focus on developing a methodology aiming at conceiving future energy systems, which will be referred to as Multi-Energy Cyber-Physical-Systems (ME-CPS)

1.3 Work objectives

This work aims at proposing a co-design methodology that is motivated by the fact that *large numbers* of stakeholders are in constant interactions for the design and operation of complex systems such as ME-CPS. When many stakeholders with different objectives, perspectives,

and culture are involved, it is difficult, and often impossible, to reach a common agreement on the purpose of the system and its scope *before* starting to work on the system. This is in stark contrast with the classical systems approach where the system goals and scope are considered to be defined first.

The purpose of this thesis is to propose a framework where the design of a complex ME-CPS will emerge throughout the negotiations between stakeholders, each stakeholder putting forward its own objectives and interests, but being open to negotiation in order to reach a common agreement with other stakeholders. The difficulty being certainly more than proportional to the number of stakeholders (maybe exponential), these negotiations need to be formalized in contracts, whose achievement shapes the ME-CPS. Negotiations here do not necessarily mean that stakeholders are in conflict, it may simply refer to discussions between teams for the sake of optimizing the overall system

One of the main purposes behind the development of this methodology is also preventing stakeholders from over-specifying their requirements regarding others. It is generally due to the uncertainties in regard to the behavior of the each stakeholder's environment. The idea here is to use rigorous Modeling and Simulation (M&S) all along with systems development and operation phases. The methodology shall overcome challenges that characterize industrial projects including multiple stakeholders such as confidentiality and the fact that interacting systems are managed and operated independently by various entities.

The methodology that will be introduced later on in this thesis is named Common Requirement engineering Modeling Approach (CReMA).

1.4 Scientific challenges

In order to ensure the objectives introduced above and overcome the challenges of future ME-CPS design, this thesis needs to provide a rigorous methodology that is based on the formal modeling of requirements and systems' behaviors that can be exploited and verified through simulation all along the systems lifecycle. The idea is assisting stakeholders throughout negotiations with means enabling them to verify, on the one hand, that the contracts they are negotiating answer their objectives, and on the other hand, their ability to commit to their obligations in regard to the other stakeholders'.

To do so, the methodology will need to propose a framework that is centered and driven by the concept of "Stakeholder" instead of the classical vision driven by "System" missions and goals. Moreover, the methodology needs to provide stakeholders with a rigorous framework to elaborate, negotiate and make formal Verification and Validation (V&V) of contracts at all stages of the system lifecycle (from early design phases to operation and maintenance), while taking into consideration the concerns of each one of them, which are generally in

contradiction between each other. A formal algebra that rules this framework and guides the elaboration of the verification models needs to be developed.

In addition, and for the sake of making different kinds of analysis, this framework is required to be modular and allow stakeholders to easily adjust their models depending on the desired level of granularity of the analysis.

Furthermore, the methodology needs to foster the changeability and transformation of systems by having the capacity to evaluate the impacts of changes at any moment of time in order to operate them with minimum effort.

In this thesis, we consider that the final system will emerge from the conform implementation of all the contracts that have been harmoniously negotiated and validated by the involved stakeholders all along the design phases.

1.5 Research background

The state of art established in this thesis was oriented towards the existing works dealing with the notions of stakeholders, stakeholders coordination, goal oriented design, System Engineering (SE) methodologies for the design of CPS and SoS with a focus on formal techniques for Requirement Engineering (RE) and V&V.

The concepts of stakeholder and stakeholders' coordination are central to the challenge that we are willing to undertake in this work. In literature, they are strongly highlighted by Systems Engineering (SE) processes [42] as being critical elements for developing complex systems. However, in Freeman's stakeholder pioneering theory [78][123], now recognized as a reference, the capture of stakeholders' requirements is considered from a single designer or analyst point of view. According to this theory, requirements are oriented toward social and ethical aspects, and the impact of industrial production on the environment. This approach has led to valuable societal results as companies now consider a wider scope than the sheer interest of their own stockholders for making strategic decisions. However, Freeman's approach is not intended to answer one of the biggest problems facing ME-CPS design, which is the coordination challenge between multiple stakeholders concerned by a common system. When many stakeholders with different objectives, perspectives and backgrounds are involved, it is generally difficult if not impossible to satisfy everyone if they are not involved in a framework where negotiations and discussions are possible.

Yu et al. have introduced in [141][140] a framework called the "i star (*) framework" that focuses on social aspects and intends to integrate the social characteristics into software and information systems engineering [142]. It emphasizes the necessity and value of modeling social actors, each one having intrinsic intentions and goals. The approach aims to model different means-ends allowing actors to achieve their goals and also to identify what

relationships to establish between them to optimize their intentions. This framework was partially integrated into CReMA methodology and was adapted to energy systems in order to structure the relationships between stakeholders at conceptual phases.

Regardless of social aspects, and focusing on the existing approaches for the design of CPS, and more specifically to ME-CPS, a detailed state of the art concerning SE methodologies was summarized in [47]. It emphasizes the lack of rigorous design methodologies capable of addressing all the particularities of ME-CPS (long lifetime systems with harsh physical constraints and stringent safety regulations that are organized more into a system of systems distributed over a network of multiple stakeholders...), especially when dealing with formal requirements and thus the automatic V&V. Amongst the few identified works dealing with the rigorous capture of requirements, T.Nguyen has introduced in [115][114] a formal requirement modeling language with a design methodology for S-CPS that is based on the concept of “Formal contracts”.

The notion of formal contracts was embraced in CReMA methodology as an essential concept for stakeholders’ coordination. In literature, it was first introduced by B.Meyer in [108] in order to put an emphasis on the quality of programs and to develop bug-free software. In this paradigm, a contract for a given module is defined as a couple (precondition, post-condition). The precondition states the prerequisites to be satisfied by the inputs of the module. The post-condition specifies the conditions to be satisfied by the outputs. The contract states the obligations of the module with respect to its environment, and the obligations of the environment with respect to the module. A formal characterization of contracts was later introduced by Inria in [53][54]. This work proposes an Assume/Guarantee paradigm where Assumes are the pre-conditions and Guarantees are post-conditions. It introduces a unified treatment for Contract Based Design (CBD) with a methodology having a rigorous algebra for contract abstraction, refinement, decomposition, conjunction and verification.

1.6 Our contribution

The idea of the CReMA methodology is to switch from a vision centered on “Systems” to a vision centered on “Stakeholders”. This does not mean that the concept of system is dropped. It just means that the design of systems is driven by the concept of stakeholders. Instead of starting from the technical standpoint of the classical system’s definition, which mainly focuses on what a system shall do and how well it should perform, our starting point is the high-level needs of stakeholders that represent their primary motivations for committing to the overall system.

CReMA methodology introduces a co-design framework composed of two main views: the Stakeholder view and the System view, each of them containing multiple modeling levels

that are detailed in section 3.7.

The Stakeholder view focuses on capturing and formalizing the social aspects of stakeholders. The social aspects here refer to stakeholders' intrinsic interests, goals, and values on the one hand, and to the relationships and interactions between stakeholders on the other hand. In this view, stakeholders specify their needs and refine them throughout multiple steps in which they identify the alternative means-ends to achieve their goals while keeping track of the rationales behind their decisions. Throughout this refinements, stakeholders identify the relationships (that will be later called dependencies) that will enable them to meet the objectives that they cannot achieve by their own (e.g. a pure consumer cannot satisfy his need for electricity on his own. He will look forward to creating a relationship with a stakeholder that could supply him with electricity).

In order to formalize these aspects (high-level goals and relationships) and integrate them into a rigorous design approach, CReMA methodology was embedded with the formal requirement modeling language named FORM-L that allows bridging the gap between requirement engineering and disciplinary models, and thus giving the possibility for systematic V&V. Using FORM-L language, stakeholders' goals are formalized into "Key Performance Indicators (KPIs)" that are used for decision making and to evaluate alternative solutions. The requirements and obligations related to the relationships between stakeholders are also formalized and integrated into formal contracts. In a contract, the obligations of a stakeholder towards others are named guarantees, the expectations of the stakeholder regarding the others are named assumptions. A contract has two (or more if more than two stakeholders are involved in the contract) points of view depending on the stakeholder standpoint. Formal contracts have the particularity of encompassing all what a stakeholder needs to know about its environment (including other stakeholders). Indeed, the assumptions describe an envelope of all possible external behaviors that are considered to be always satisfied. From that point on, stakeholders may independently design the system that answers their guarantees (obligations toward other stakeholders) as well as their own goals.

This brings us to the second view of CReMA methodology: the "System view". This view aims at describing a digital twin of the systems under the responsibility of stakeholders. The digital twin is considered in our methodology as including dynamic behavioral models of a system as well as corresponding requirement models (defined in the previous view). These two elements are necessary to ensure that the system meets the goals for which it was developed. This is even more valuable when dealing with safety-critical systems for which proving the correct operation of the system is as important as the design itself. The digital twin undergoes several modifications during the different phases of a system engineering lifecycle. Different design steps require different levels of granularity of behavioral models to challenge the requirements that are induced at every stage.

Finally, CReMA methodology is equipped with a formal algebra to rigorously define concepts such as contracts, contracts implementations, environment, KPIs, etc. and also operations on these concepts such as contract verification, contract signature, etc. This algebra was inspired by the works of Inria [53] and adapted to fit our challenges.

1.7 Manuscript organization

The thesis manuscript is organised as follows:

Chapter 2 introduces a state of the art and discussions on existing methodologies related to the challenges introduced above. We also discuss for each work its strong assets and limitations that we will try to overcome.

Starting from the identified limitations of existing methodologies, and while being inspired by some works from literature, CReMA methodology is introduced in chapter 3.

A formal algebra for contract definition and operations is proposed in appendix C.

In order to validate CReMA methodology, an industrial case study of a Smart-grid was used and results are shown in chapter 4.

Finally, a conclusion with further perspectives is presented in chapter 5.

Chapter 2

Complex systems design - a state of the art analysis

In this chapter, we will introduce the main works from the literature that deal with the challenges presented in the introduction. The aim is to position our methodology in regards to existing works and also to acknowledge the works that have inspired some parts of our methodology.

The results of the state of the art work were partially published in the IEEE International Systems Conference, SysCon 2019 using the title “A Survey on Systems Engineering Methodologies for Large Multi-Energy Cyber-Physical Systems” [47].

2.1 An overview of Systems Engineering

2.1.1 SE history

The term “Systems engineering” have first emerged in the early 1940s in the context of the development of the first airspace defense system in the USA by the Bell Telephone laboratory. The first application of systems engineering was achieved on a large scale project in the military framework. The growing complexity of military systems soon required formal systems engineering processes in the form of standards which led to the MIL-STD 499:1969 standard [36] in 1969. Requirements and specification processes, interfaces management, and milestones respect were then known. In the early 1980s, systems complexity increases, and systems engineering starts to spread in different fields and becomes international since the 1990s. Worldwide organizations took inspiration from standards such as IEEE1220:1994 [37], EIA 632:1998 [72] and ISO/IEC/IEEE 15288:2015 [42]. The National Council on Systems engineering NCOSE promoting SE in the USA becomes international in 1995 under the name of “INCOSE”, being thus the first world organization in SE both by its creation date and its

size. Figure 2-1 shows a global overview on the evolution of these standards.

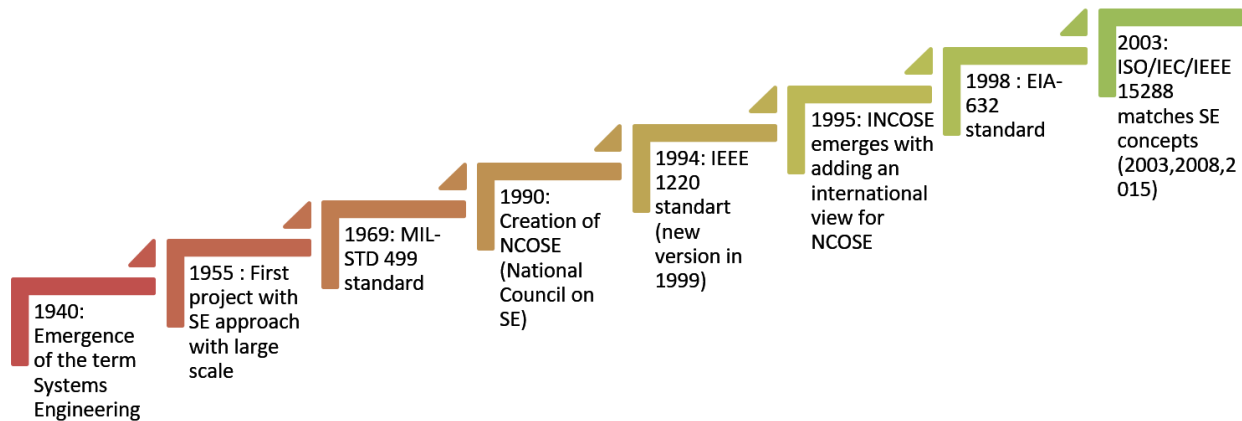


Figure 2-1: Systems Engineering history from [18]

2.1.2 Introduction to SE

According to the INCOSE, SE is:

“an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem: Operations, Performance, Test, Manufacturing, Cost & Schedule, Training & Support, and Disposal. Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.” [138]

Systems engineering standards in general and more specifically ISO/IEC/IEEE 15288:2015 [42] describe a set of processes to be implemented at different steps of a system lifecycle in order to maximize the chances of success of complex systems. However, SE does not describe any specific methodology, languages nor tools to deal with complex systems as it is meant to be cross-disciplinary and applicable to all kinds of complex systems. Before introducing SE processes, it is very important to make a difference between processes, methods, and tools.

A **process** is a “*set of interrelated or interacting activities that transform inputs into outputs*” referring to ISO 9000:2015 [19] clause 3.4.1.. SE processes describe the “What” of the system, i.e. what the system shall do, or what should be done to create the system. They describe the activities to be realized and define the inputs and outputs to produce. Processes do not define how the activities should be performed and how outputs will be

produced. For instance, the process to be performed by an HVAC (Heating, Ventilation, and Air-Conditioning) system bears on having a required level of thermal comfort and indoor air quality. It can also specify which specific rooms need cooling and heating.

Methods are techniques for performing activities: they define the “How”. By considering a set of inputs and desired outputs, the methods describe how the system shall perform in order to reach its goal, and define the technical implementations of activities while relying on tools and languages for better handling the complexity of systems.

Tools represent all instruments used in order to improve the efficiency of methods. They require specific skills and training for efficient use. The aim of tools is encompassing and supporting processes and methods [86].

Finally, a **methodology** can be defined as a detailed combination of processes, methods, and tools dedicated to a specific kind of systems with certain similarities.

SE processes are decomposed into four categories according to the ISO/IEC/IEEE 15288:2015 [42] standard:

- Technical processes: they concern the activities to be done to develop the concrete system, or what we call the SoI (System of Interest).
- Technical management processes: they concern the project management activities to be implemented for a good elaboration of the final system.
- Agreement processes: they describe the way of handling a contractual relation between system owners and suppliers.
- Organizational project-enabling processes: they concern the organization that should be done at the level of the company to enable the elaboration of the system.

The last three processes concern the SUTD (System Used To Design) which comes as a secondary system in parallel to SoI in which SE gives an important value for the overall system success. It is about handling activities, tasks, and resources.

2.2 Stakeholders and stakeholders’ coordination

In literature, we have frequently met the concepts of stakeholder and the capture of stakeholders’ needs. They are considered to be key success factors for the design of complex systems involving multiple entities and affecting large numbers of individuals. The notion of stakeholder has rapidly spread in literature in the 80s [46], and the most widespread definition and recognized work is that of Freeman [78][79]. In Freeman’s stakeholder theory, pioneering in the subject, stakeholders are defined as “*any group or individual who can affect*

or is affected by the achievement of the organization's objectives." in [78]. This definition was later revised to become: *"those groups who are vital to the survival and success of the corporation"* [80]. This work spotlights the necessity of considering social and ethical aspects in the management of companies and especially when making strategic decisions. This approach has led to valuable societal results as companies now consider a wider scope than the sheer interest of their own stockholders for making strategic decisions.

These notions are also strongly highlighted in SE processes [42] and in consequent methodologies as being critical elements for developing complex systems. Referring to the INCOSE Handbook [138], a stakeholder is: *"any entity (individual or organization) with a legitimate interest in the system."* Amongst the technical processes of SE, we have identified the *"Stakeholder needs and requirements definition"* process that claims that *"When nominating stakeholders, business management will take into account all those who may be affected by or able to influence the system"*. This process considers stakeholders as a sources of requirements that should be part of the system specifications and that need to be addressed by the chosen solutions.

Similarly, the SE referential of the French organization for Systems Engineering (AFIS) [45] emphasizes the necessity of identifying stakeholders as well as their needs and requirements in early conceptual studies before being decomposed into system requirements and translated into functions to be allocated to organic elements.

The main limitation identified in existing works in regards to nowadays complex CPS challenges is that stakeholders are considered from a unique analyst point of view and these works try to develop a system that meets its owners' objectives while taking into account the external stakeholders' requirements when taking strategic decisions.

This point can be further observed in the standard ISO 42010 section 5.3 [87], where stakeholders identification is limited to system users, operators, acquirers, owners, suppliers, developers, builders, and maintainers.

In this work, we propose to handle stakeholders in a more integrated paradigm. Indeed, as we are dealing with SoS [104] having multiple stakeholders that are operationally and manager-wise independent, we propose to provide them with the means to specify their own needs, and based on their decomposition negotiate the terms of contracts that will be established between them. These contracts will allow structuring the relationships and interactions between them.

This work takes into account the challenges emerging from gathering stakeholders from different entities and disciplines such as confidentiality of knowledge, lack of common language, and the heterogeneous visions on the same system.

Existing frameworks such as 3DEXPERIENCE [69] and Teamcenter [34] allow via Product Lifecycle Management (PLM) and Data Lifecycle Management (DLM) platforms to

guarantee customized access to large projects contributors according to their profiles and roles, with data management possibilities throughout the product lifecycle. However, they offer no methodology to coordinate stakeholders in the sense of assisting them to seek and agree on common objectives. They can be assimilated to information systems that support the different actors of an organization by collecting, processing, storing, and distributing information [124].

2.3 i* (star) framework

As part of the coordination of ME-CPS stakeholders, and for the sake of capturing and structuring their objectives, and also for keeping track of the rationales behind their choices, CReMA methodology got partially inspired by the book “Social modeling for requirement engineering” [73]. This work presents the “i* framework” that covers the goal-oriented and agent-oriented modeling approaches. This approach considers the world as having actors, intentions, goals, and rationales behind any behavior. Actors are not only considered to interact physically with each other but are also related to each other at an “intentional level” which is captured by the framework. At this level, no predefined sequences or behaviors are specified, actors interact through their goals and commitments to others.

The particularity of the i* framework is that it introduces a richer modeling framework that is not limited to modeling complex systems processes, but provides means to capture the “Why” dimension of processes. The classical process models tend to represent systems using activity diagrams and workflows between components. The i* framework pushes the analysis further to characterize actors using properties such as goals, beliefs, ability, and commitment [73].

The i* framework consists of two modeling elements that were adopted by CReMA methodology, the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

2.3.1 Strategic Dependency (SD) model

The Strategic Dependency (SD) model represents a network of dependency links between actors. It models the world using a set of nodes representing the actors and connections corresponding to the dependencies. The depending stakeholder is called “Depender”, the stakeholder depended upon is the “Dependee” and the dependency element is called “Dependum”.

The dependencies are categorized into four types:

- Goal dependency: the depender depends on the dependee to carry out a certain state to the world. The depender does not specify how to do it. The dependee has the

freedom to choose his way. (e.g. a patient depends on his doctor to be cured, while the doctor has the freedom to choose the means to cure the patient).

- **Task dependency:** The depender depends on the dependee to achieve a specific activity with a precise way of doing it. The goal of the task is not shared with the dependee. (e.g. electricity supplier depends on the producers to adapt their production according to his demand).
- **Resource dependency:** The depender depends on the dependee to afford him a physical or virtual element that has limited availability, and that will be used for a specific purpose of the depender. (e.g. electricity, money).
- **Soft-Goal dependency:** In this dependency, the goal is not precisely known and the conditions to reach it are specified throughout the accomplishment of the task or goal. (e.g. being eco-friendly).

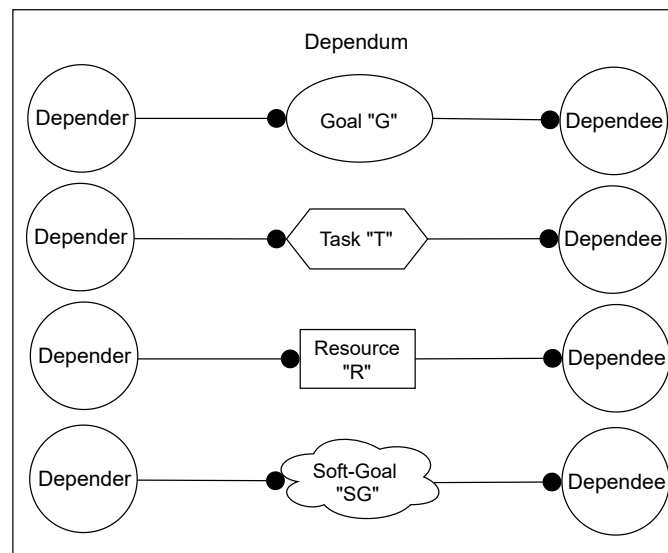


Figure 2-2: Dependency types of the i* framework

Figure 2-2 shows how the i* framework represents the four kinds of dependencies using graphical nodes and arrays going from the depender to the dependee, passing through the dependum.

These dependencies are characterized using three vulnerability degrees that allow each actor to assess the viability of his dependencies: (i) Open dependency, (ii) Committed dependency, and (iii) Critical dependency. A formal definition of these aspects can be found in [73]. CReMA methodology will not include these aspects as we assume that when stakeholders sign a contract, it is understood that they have necessarily justified their ability to

commit to their engagements. We also consider that stakeholders will further commit to their engagements.

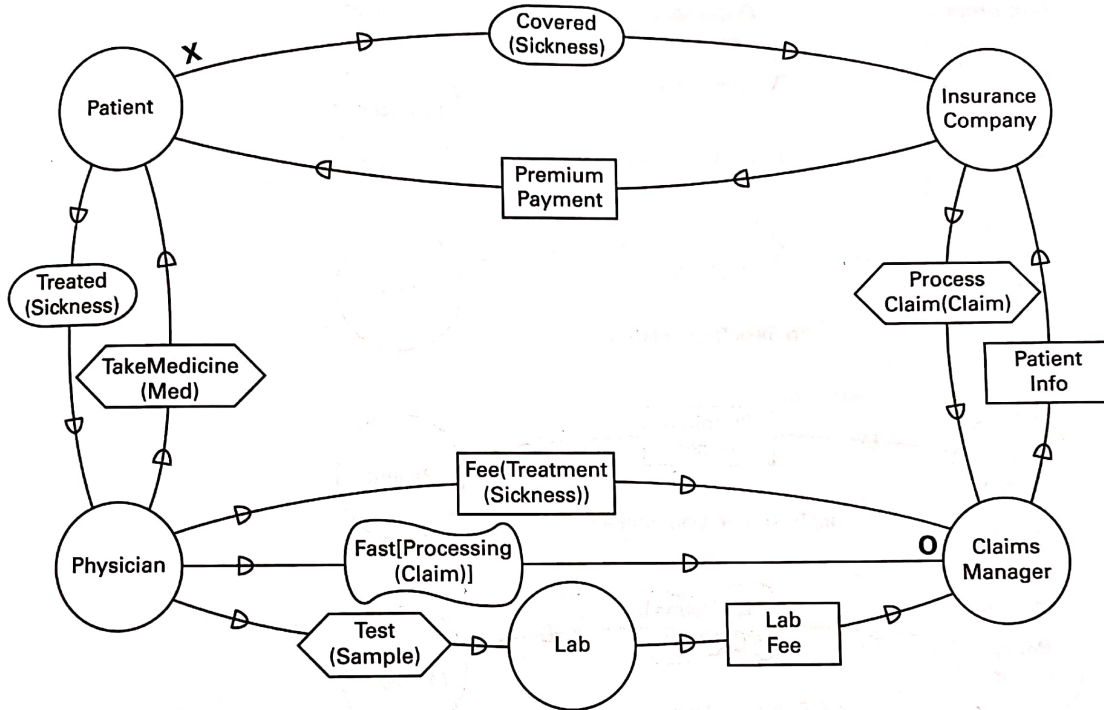


Figure 2-3: Example of a strategic dependency model from the health care domain (from [73])

Figure 2-3 shows an example of an SD model that was taken from the health care domain, and represents, using directed links, the dependencies of various types between stakeholders.

2.3.2 Strategic Rationale (SR) model

The Strategic Rationale model (SR) describes on the one hand the rationales behind the dependencies between actors, and on the other hand, the means-end used by the actors to carry out their own goals and soft-goals. This model focuses on the internal intentional relations of actors by answering the “why” and “how”, as well as the alternative solutions by showing different possible means-ends of a goal or a task.

An example of strategic rationale model that was taken from [73] is shown in Figure 2-4. It shows the reasoning behind the dependencies between three actors from the same health care example of Figure 2-3. The dashed circle around each set of goals and tasks delineates the scope of each stakeholder. This model shows the high-level goals, soft-goals, and tasks

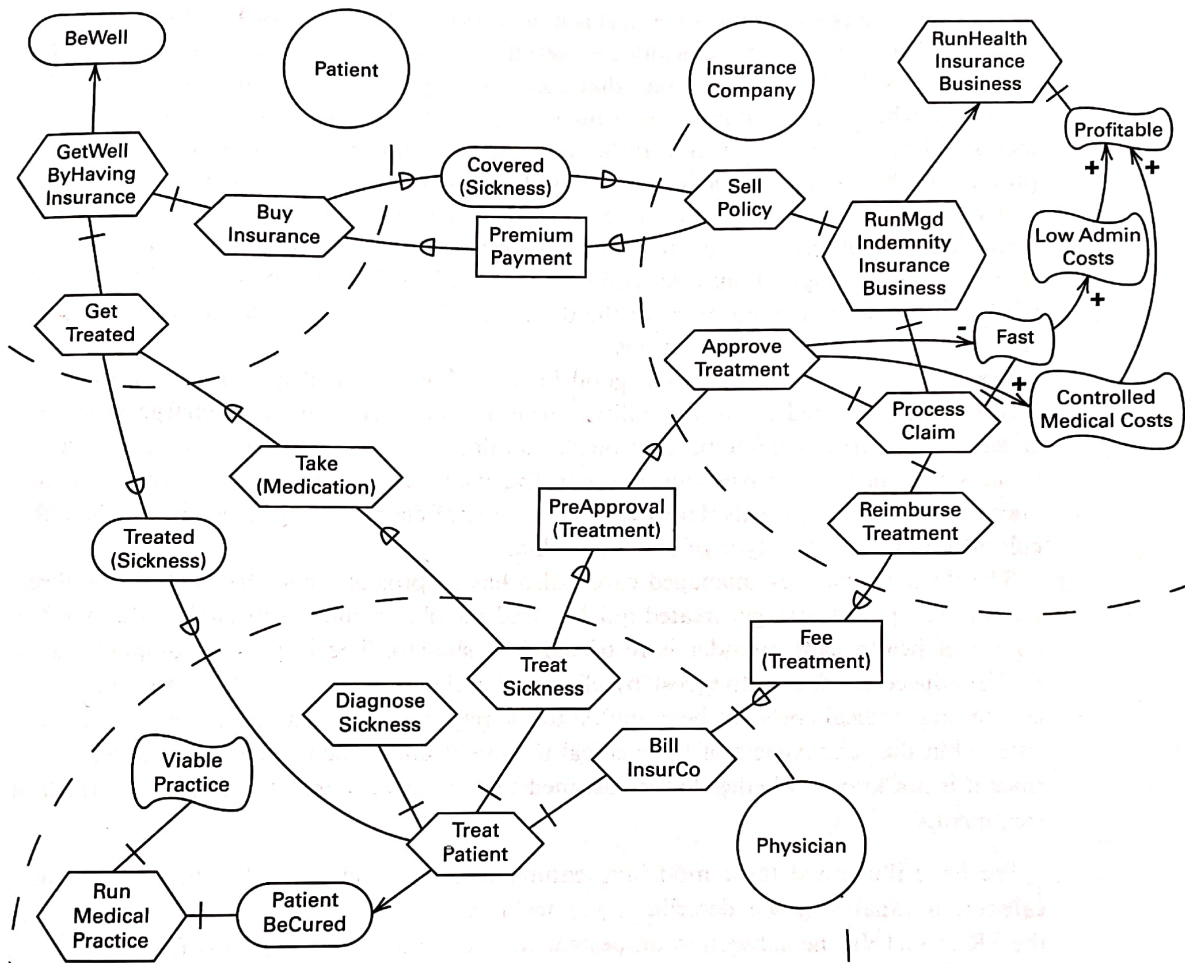


Figure 2-4: Example of a strategic rationale model from the health care domain (from [73])

to be achieved by each actor. These properties are thereafter decomposed into sub-elements that represent the “means-end” which leads to dependencies regarding the other actors.

In CReMA methodology, the intentional level will allow us to analyze the roles, the objectives, and the reasons behind the dependencies between stakeholders before going to the formalization of these elements that will turn into stakeholders’ KPIs and requirements integrated into contracts.

2.4 Requirement engineering (RE)

Requirements are considered to be a pillar of SE practices. Requirements engineering (RE) gathers systems engineering processes dealing with requirements definition and management in the engineering design phase. Requirements engineering defines how requirements should

be constructed and considered all along a system lifecycle in order to ensure a correct capture of stakeholders' needs. The standard ISO/IEC/IEEE 29148:2011 [38] focuses on RE and presents in detail all activities and guidelines concerning SE requirement processes. In our document, an overview of good practices for capturing requirements and managing them is presented, we invite readers interested in more detail to refer to the section 5.2 of ISO/IEC/IEEE 29148:2011 [38] standard concerning "Requirements fundamentals".

A requirement is *"a statement that identifies a system, product or process characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability."* INCOSE [138].

2.4.1 Characteristics of good requirements

The standard ISO/IEC/IEEE 29148:2011 [38] dissociates between characteristics of requirements and of a set of requirements which is a gathering of several requirements. Indeed, certain characteristics have to be considered on sets of requirements related to elements as stakeholders, systems, or subsystems. Each individual requirement shall respect the following characteristics:

- Necessary: The requirement should be created for an essential characteristic whose omission leads inevitably to deficiencies that cannot be mitigated by other capabilities.
- Implementation free: The requirement should avoid any constraints on the architectural design and consider the system as a black box.
- Unambiguous: The requirement should be specified in a way such as only one interpretation is possible.
- Consistent: The requirement should not be contradictory to other constraints.
- Complete: The requirement as stated should not need more details or refinement to be understood because it is measurable and provides enough information.
- Singular: The requirement should not be a combination of multiple requirements.
- Feasible: The requirement should be technically achievable and fit within acceptable constraints and risks.
- Traceable: The requirement should be traceable to an upstream requirement or source. It should also be traceable downstream to specific requirements, functions, or components.

- **Verifiable:** The requirement should express characteristics that can be proved to be verified by the system with one of the following methods: inspection, analysis, demonstration, or test [138].

The characteristics for sets of requirements that need to be respected are:

- **Complete:** The set of requirements should contain all necessary and pertinent information for the definition of the system.
- **Consistent:** The set of requirements should have no contradictory or duplicated requirements. The terms used in the requirements come from a unique glossary.
- **Affordable:** The set of requirements can be satisfied with a solution that is achievable while respecting the system lifecycle constraints in terms of cost, schedule, technical, legal, regulatory, etc.

The standard ISO/IEC/IEEE 29148:2011 [38] also introduces requirement attributes that come as analysis support for requirements and intend to ease their understanding and handling. Attributes are attached to each requirement and carry information such as Identification, Priority, Dependency, Risk, Source, Rationale, Difficulty, and Type. Furthermore, the standard also presents some language criteria and considerations when writing requirements. Please refer to section 5.2.7 of the standard for more details.

2.4.2 Requirements classification

Several classifications of requirements are found in the literature regarding requirement definition methods and considered design methodology. An example of requirements classification taken from the SEBoK [56] is shown in Table 2.1.

2.4.3 Requirements modeling approaches

In literature, various languages and tools were developed for modeling, managing, and verifying systems requirements with different degrees of formalizing: Informal, Semi-formal, and Formal. Some approaches use natural language for requirements capture. Others use templates to guide their formulation, while the third kind uses mathematical expressions for doing so.

2.4.3.1 Non-formal and semi-formal requirements modeling

A large majority of tools being used for RE in the industry are either non-formal or semi-formal supported with templates. These tools are essentially used for managing the large

Table 2.1: Requirements classification according to the SEBoK

Types of requirements	Description
Functional requirements	Describe qualitatively the function that shall be achieved by the system.
Performance requirements	Define quantitatively how well the system shall perform the functions.
Interface requirements	Define how the system is required to interact or exchange with neighboring systems or internal elements supporting interactions.
Operational requirements	Define the operational conditions or properties that are required for the system to perform correctly as human factors, ergonomics, availability, reliability, security. . .
Modes and/or states requirements	Define various operational modes of the system in use and events conducting to transitions between modes.
Adaptability constraints	Define the potential extension, growth, or scalability during system lifecycle.
Physical constraints	Define constraints on system physical aspects such as weight, volume, dimensions.
Design constraints	Define the limits of the options that are available to a designer of a solution.
Environmental conditions	Define the environmental conditions to be encountered by the system in its different operational modes. It encompasses natural, societal and other environments.
Logistical requirements	Logistical conditions include sustainment, shipping, transportation. . .
Policies and regulations	Define relevant and applicable organizational policies or regulatory requirements that could affect the design, operation or performance of the system.
Cost and schedule constraints	Define the cost of the system and the expected delivery dates, milestones . . .

number of system requirements arising from stakeholders' needs and induced from the design choices made through the engineering lifecycle.

- Informal approach: This approach consists in using natural language for describing systems requirements. It is used in several tools such as DOORS [27] and SysML [119] that are widely used in the industry as natural language that does not need any specific training (provided that the technical expressions are known). Moreover, unlike formal methods, natural languages allow expressing all possible ideas [99]. However, this does not mean that no expertise is needed. Understanding the requirements issued by experts from other disciplines requires some proficiency in the field in order to avoid silly mistakes. Also, natural language is potentially ambiguous as a given word or sentence

may have different meanings depending on the context, and consequently may lead to different and possibly contradictory interpretations. Such misunderstandings may result in heavy financial losses (e.g. due to system over-sizing, or failure to comply with regulatory requirements), and therefore call for more formal methods. Last but not least, automatic verification of properties is impossible when using natural language, which makes the exploration of design alternatives cumbersome.

- Semi-formal approach: This approach comes as a candidate solution to improve natural language requirement management by introducing restrictions on the way of formulating requirements. Some semi-formal tools consist in using template languages that control requirements capture using gapped sentences in the form: “The *{system}* shall *{do action}* with *{a given performance}*”. Other languages choose to keep the natural language and add hyper-links to the requirement text [99]. Links ensure a one-to-one mapping between a word or a group of words and their meaning. As a consequence, humans, and machines have a unique way of interpreting the requirements, provided that the meanings are well formulated and ideally standardized. This approach avoids misunderstandings but is still not suitable for the automatic verification of requirements because templates with textual words as arguments cannot be mathematically evaluated.

Amongst the languages and methods for non-formal and semi-formal requirements that were identified in the literature, we can cite:

- Lebeaupin et al. presented in [99] a methodology for specifying requirements in order to mitigate the ambiguity of natural language requirements by creating a one-to-one correspondence between every mentioned element, parameter, and variable in the text with a single meaning. Thus, each requirement would have a unique interpretation, and each element will have a unique representation in all requirements. The authors of this work consider that not all requirements can be formalized and give examples: “*The system equipment or structural items shall remain within their own design and integration envelope after a failure leading to a partial or total part detachment*” [99]. However, this takes us further in another phase of requirements engineering, and especially in requirement capture where requirements shall respect a number of criteria, as presented by SE processes (see section 2.4.1). It is certainly not an easy task to specify requirements while sticking to complex requirements engineering processes, but it is capital to specify requirements in a rigorous and verifiable manner to be able to handle the complexity of large CPS.
- DOORS[27] is a tool developed by IBM which is widely used in industry for managing

non-formal requirements. It is used in order to capture, trace, analyze, and handle requirement modifications.

- ReqIF[28] (Requirements Interchange Format) is a open requirement format developed by the OMG which allows requirements exchange between multiple software tools from different editors.
- SSPL (System Specification Pattern Language) [82][83] is a template language integrated into the SAVONA [30] platform. It has the particularity of being highly expressive, readable, and easy to use. The verification which is possible using SSPL concern the coherence of the connections between blocks in term of flux and inputs/outputs.
- Other non-formal languages and methods were found in the state of the art: (i) Requirement Specification Language (RSL) [126] introduced within the CESAR European project[9]. (ii) [88] presents an application of model-based methodology to formalize requirement of railway systems.

2.4.3.2 Formal requirements modeling

The formal requirements modeling approach consists in representing requirements using mathematical and logical expressions. It allows having a unique formal meaning for each property. It opens the possibility of executing requirements in order to automatically perform important tasks such as consistency checks or design verification. Thus formal approaches are necessary for the design of next-generation ME-CPS as they open the possibility of evaluating large numbers of alternatives in the search of best solutions that comply with complex requirements sets. The drawback is that the expressiveness of the requirements is limited to the semantics of the modeling language. In practice, formal approaches must be combined with non-formal or semi-formal ones to cover the full scope of SE.

The literature on requirements engineering shows that there are two main formal requirement paradigms.

The first paradigm was inspired by software development and is based on logical representations of requirements temporal logic languages such as [49][77][92]. For the sake of making formal V&V of requirements, this paradigm calls for representing system behavior using transition systems and state machines under the same temporal logic language. This approach is called model checking and will be detailed further in section 2.5.2.1. The main limitation of temporal logic is that it only works on systems with finite predefined discrete states (i.e. known at model design) which is not adapted for analyzing and evaluating the dynamics of physical systems that can have an unlimited number of continuous and discrete states. Indeed, logical representations do not take into consideration the physical aspects

of systems. They partially consider the real-time aspects and mainly focus on the logical aspects of control systems by exploring the transitions between states.

The second paradigm intends to formalize requirements using languages and tools, having a well-defined syntax and semantics, allowing from the one side to verify the consistency and the correct establishment of requirements, and from the other side to observe behavioral models and evaluate system specifications using simulation. The particularity of this paradigm is that it does not restrict engineers to use a specific modeling approach to be in line with requirements. However, it considers requirements as an independent layer that needs to be connected to the analytical layer to make systematic V&V (refer to section 2.5). Doing so gives designers the freedom (within the limits of the language) to use domain-specific languages that are adapted to the discipline they are working on.

Two main works were identified:

- STIMULUS [89] is a tool developed by ARGOSIM and is currently part of the 3DEXPERIENCE [69] toolchain. It allows requirements debugging in early design phases in order to exhibit requirement expression problems such as inconsistencies, incompleteness, ambiguity, or missing requirements.

STIMULUS uses a boilerplate language for capturing requirements. The sentences obtained by filling the boilerplates are translated into an internal formal language that enables the requirements to be simulated for verification. However, a substantial manual transformation of the requirements originally expressed in natural language must be done by the user in order to express them as a timed automata, which is the core paradigm used by STIMULUS to determine whether requirements are satisfied or not. An example of a requirement written in STIMULUS is shown in Figure 2-5.

```
When ( ( ( switch is 'AUTO' ) and ( headLight was 'OFF' ) and ( lightIntensity is below ( 60 % ) ) ) ) during more than 2 [second] , headLight shall be 'ON'
```

Figure 2-5: Example of a STIMULUS Requirement (from [89])

The strength of STIMULUS is its ability to automatically generate test sequences compliant with the assumptions made on the system environment. These test sequences represent a simplified (or envelope) behavior of the environment of the system that can include stochastic aspects (transition probabilities between states) and even non-deterministic aspects (such as random walk within a given interval). This is indeed useful when little information is known about the behavior of the environment, whether

the environment is the natural environment (i.e. weather), or is made of subsystems that are not known yet or in very preliminary design stage.

Finally, in more advanced design steps, the requirements can be challenged using other kinds of behavioral models such as finite state machines or physical models (e.g. Mod-*elica* models) that can be imported into STIMULUS provided that they comply with the Functional Mockup Unit (FMU) standard. Observers can then be plugged onto the behavioral models to feed in the variables used in the requirements with the data computed by the behavioral models (the observers switch from observing the test sequences to more precise deterministic behavior).

The main drawback is that it is not easy to make sure that requirements are correctly expressed in STIMULUS because they must be expressed in the STIMULUS programming language which is not close to natural language when it comes to timed automata, and because the principles for the generation of test sequences are not public, so it is very difficult, if not impossible, to assess how relevant the test sequences are to the problem at hand. Also the tool is closed in terms of pairing with other platforms useful for the handling of cross-domain systems. Moreover, it does not take part in a more holistic methodology taking into consideration other processes of the engineering cycle. The main interest of STIMULUS is not in requirements engineering (structuring, formalization, traceability...) but rather in requirements debugging.

- The second work dealing with formal requirements modeling and simulation is the language named FORM-L (FOrmal Requirements Modeling Language) [114] that was developed in the framework of ITEA2 European project MODRIO [20]. An initiative for the standardization of this language is in progress within the ITEA3 project EM-BrACE [21], under the name of CRML (Common Requirement Modeling Language). The relevance of this language lies in its capacity to formally express and organize requirements as sets of constraints on objects physical time and spatial location using a syntax that is intuitive enough for engineers, to the contrary of temporal logic languages that use mathematical notation. The ability to express formal requirements in a language that is close enough to natural language is important in order to ensure that they indeed represent the intent of the stakeholder. With FORM-L, requirements such as "*Any pumps in the system must not cavitate more than once in a sliding time period of one year.*" can be formally expressed. An example of this requirement in FORM-L is illustrated in Figure 2-6.

In order to formally describe the semantics of the temporal aspects of FORM-L, a temporal language called Extended Temporal Language (ETL) was introduced by Bouskela et al. in [61]. It is based on a 4-value logic (true, false, undecided, undefined) in order

to estimate the status of a requirement while taking into account the time periods. When a requirement is evaluated in a time period where it is not applicable, the requirement is “undefined”. For example, if the requirement “*When the system Alpha is in operation, the temperature of the water must be higher than 20 degrees Celsius*” is tested in a time period where the system Alpha is not in operation, the decision would be “undefined”. When the decision for knowing whether a requirement is satisfied “true” or violated “false” is still pending, the requirement is then “undecided”. For example, the evaluation of the requirement “*The average electric consumption over one year must not exceed 100kW*” is “undecided” during every moment of that year. Once the year is completed, the decision can be made whether or not the requirement is “true” or “false”.

ETL was implemented in the multi-domain modeling language Modelica [81] giving birth to the library called ReqSysPro. So far, the translation between FORM-L and ReqSysPro is done manually using the ReqSysPro blocks. A compiler is under development to automate this task in partnership with Inria/Sciworks Technologies and with the EMBrACE consortium.

```

requirement R1 is for all pump in system.pumps
    during any 1*year while (system.inOperation)
    check count (pump.cavitation)<=2;
```

Figure 2-6: Example of FORM-L requirement

It has to be emphasized that the formal dimension of requirement engineering is essentially when dealing with systematic V&V. Therefore, a deeper overview will be given to model checking and V&V using simulation in section 2.5.2.

2.5 Verification and Validation (V&V)

2.5.1 Definitions

SE have introduced two distinct processes: the Verification process and the Validation process.

As stated in the SEBoK [56] and in conformance with ISO/IEC/IEEE 15288 [42]:

“System Verification is a set of actions used to check the correctness of any element, such as a system element, a system, a document, a service, a task, a requirement, etc. These types of actions are planned and carried out throughout the life cycle of the system.”

“System Validation is a set of actions used to check the compliance of any element (a system element, a system, a document, a service, a task, a system requirement, etc.) with its purpose and functions. These actions are planned and carried out throughout the life cycle of the system.”

The verification process intends to answer the question “*Am I building the system right?*”, in other terms “*Am I correctly building the system?*”. Whereas, Validation intends to answer the question “*Am I building the right system?*” and using other words “*Am I building the system that responds to stakeholders requirements?*” [59]. Thus, in the validation process, the system is considered as a black box, as we verify that the behavior at the output of the system meets stakeholders’ requirements. The verification process deals with the system as a white box where we go inside the system and verify that technical behavior meets the technical requirements of the system.

V&V processes can be applied at any stage of system design. In order to be efficient and avoid heavy costs due to modifications in advanced system lifecycle phases, which are essentially caused by errors in requirement analysis or design phases, it is recommended to start V&V from early design activities such as requirement analysis. Let us take examples of what could be the activities of each process.

Concerning verification:

- The verification of a stakeholder requirement or a system requirement consists in verifying that the requirement respects the structure and characteristics of well-captured requirements defined by related processes and also checking the compliance with the approach (formal or informal) considered by stakeholders for managing requirements.
- The verification of the architecture of a system is done by checking the correct use of modeling tools and models/diagrams.
- The verification of a system or system element is done by checking that the properties and characteristics meet what is specified in requirements, architectures, and design documents.

Concerning validation:

- Validation of a requirement: checking that the set of requirements is consistent and meets stakeholders’ needs.
- Validation of a system: verifying that the system satisfies its missions and the stakeholders’ expectations under all operational scenarios [138].

2.5.2 Methods for V&V

Verification and Validation processes use the same techniques, but they have different purposes. Verification deals with the satisfaction of the internal technical requirements of the system, whereas validation deals with the external aspects of the system and makes sure that the latter meets the intended missions. According to the INCOSE Handbook [138] and (IEEE 1012:2012 [39]; ISO/IEC/IEEE 29119:2013 [40]; ISO/IEC/IEEE 29148:2011 [38]) standards, V&V techniques are classified as follows:

- **Inspection:** Visual verification of the conformity between the system and requirements. It incorporates stakeholders' reviews which are useful for spotting errors and making decisions.
- **Analysis:** This technique is performed using formal languages with a semantic developed on a mathematical, logical, or probabilistic basis. Formal methods allow to objectively verify the correctness of system properties.
- **Demonstration:** It consists in observing a system behavior and characteristics in its operational context without using any instrumentation. The demonstration can be done on a prototype, on the final system, and even on the process that led to creating the system.
- **Test:** The validation is done by executing the system while controlling its operating conditions, and comparing the results to the expected behavior.
- **Analogy or similarity:** This technique is performed when having the possibility of affording an equivalent element on which other V&V techniques can be applied.
- **Simulation:** This technique is based on the elaboration of a system model on which verification will be done.
- **Sampling:** It is based on verifying systems using samples. The characteristics of samples and intended results are specified before doing the tests.

In this thesis, we focus on the analysis and simulation techniques for V&V. These techniques are supported by tools with different formalizing degrees that assist engineers during the design. These tools can help to avoid several human errors by allowing systematic V&V all along the engineering cycle of systems, thus enabling the development of structured models that respect rules and required characteristics.

In literature, several works were identified dealing with formal techniques for V&V, and can be classified into two categories, model checking, and simulation (as stated in section 2.4.3.2).

2.5.2.1 V&V using model checking

Model-checking is the most widespread formal verification technique found in literature, which was inspired from software engineering. Model-checking operates on abstractions of system requirements and system behavior, both captured using finite state machines expressed in temporal logic. It is a verification approach that explores exhaustively all states of the system for all possible scenarios to check whether the system meets its given specifications. When executing a verification, the method returns “Yes” when the system satisfies the requirements and counter-examples when they are violated. Errors are debugged by analyzing the counter-examples. Model-checking allows verifying properties such as reachability, fairness, invariance, etc. [49]. However, it cannot be used on models that are not expressed with temporal logic.

In general, temporal logic is a mathematical logic characterized by rules and symbols for representing and reasoning on propositions allowing the description of the temporal ordering of events by adding temporal terms. We can then express statements such as “The phone will ring until someone answers” or “The phone will eventually ring”. The term “temporal” is used as an abstract sense of the real-time behavior of reactive systems. Temporal logic never operates explicitly with time as an absolute duration. Between two events, there may be a time period spanning from seconds to hours or even years because the value of the duration cannot be expressed in most of model checking languages LTL [49], CTL [77] and OCL [120]. Therefore, properties like “After 5s the phone must ring” cannot be expressed. Some attempts were made to create temporal logic extensions for model checking by considering the notion of duration such as STL [127], but they turned out to be computationally costly and require elaborated techniques.

Several temporal logic languages can be found in the state of the art, we can cite LTL [49], CTL [77], CTL* [65], STL [127], OCL [120], HyLTL [64], etc. The main difference between them remains in the temporal operators of language, the rules by which finite state machines are represented and structured, and the capacity to represent temporal aspects. An overview of these languages is reported in Appendix A.

Model-checking has proven to be a successful technology to verify temporal logic models against their specifications and have been applied to several industrial complex systems. For instance, it has revealed several serious design flaws in the control software of a storm surge barrier that protects the main port of Rotterdam against flooding [49].

The main limitation of model checking is that only systems expressed as finite-state machines can be checked for correctness, and it is rapidly limited when dealing with systems with continuous-time aspects and/or including high numbers of states leading to a combinatorial explosion. Techniques were then created for developing finite-state abstractions for these kinds of systems but they are still not suitable for physical systems having a large

number of continuous states.

2.5.2.2 V&V using simulation

This paradigm consists in verifying system properties by launching system simulations in the scope of the real context of the system and checking whether the requirements are satisfied or not. Two kinds of verification can be undertaken using simulation: standalone requirements verification to check the consistency of the requirements, and system behavior verification against requirements to check the system design.

Compared to model checking, the V&V using simulation techniques has the special characteristic of identifying some unpredictable behaviors that can emerge from potential components failures or any combination of perturbations. In contrast, model-checking requires logical representations including an exhaustive knowledge of the context and all possible behaviors of the system without having the possibility to consider perturbations.

The two state-of-the-art works that use the requirements simulation for V&V are STIMULUS and FORM-L language [114][63]. These works were already introduced in the formal requirement section (see 2.4.3.2)

2.5.3 V&V and requirement modeling languages and tools - Summary

To sum up, Table 2.2 gathers the different requirements modeling languages identified in the literature and that are mentioned above. They were compared based on criteria taking into consideration the ease of handling the tool and the power of the provided features.

Natural language is of course the most legible and expressive among all developed formal or semi-formal languages as it is understood by everyone and all ideas can be expressed using it. However, it has multiple drawbacks and is not adapted to ME-CPS design which requires executable and formal expressions.

Two existing formal paradigms supported by various languages and tools are available in the literature: model checking and paradigms using tools such as STIMULUS and FORM-L.

Model checking is a powerful paradigm that allows verifying properties involving time quantifiers (such as the system will never reach a given state), which is not the case for other paradigms based on 0D/1D simulation that always perform verification on finite time periods. Thus, model checking is essential for reachability analysis. However, it cannot be sufficient for analyzing large CPS where physical aspects are an essential part of the system and need to be modeled and analyzed in real-time. Paradigms using tools such as STIMULUS and FORM-L are more adapted for V&V of ME-CPS as they can be coupled to OD/1D modeling and simulation tools.

Languages & Tools Comparison criteria	Natural Language (e.g. DOORS)	Template languages (e.g. SSPL)	LTL/CTL/CTL*	STL / HyLTL	OCL	TOCL	STIMULUS	FORM-L / ETL / ReqSysPro
Readability	++++	++	-	-	++	+++	++	+++
Expressiveness	++++	+++	+	++	+	+++	++	+++
Formalization (executable)	-	-	+	+	+	?	+	+
Debugging features	-	-	+	+	+	?	++	-
Adapted for verification and validation of physical systems	-	-	-	-	-	++	+	++
Adapted for reachability analysis	-	-	+	+	-	+	-	-
Openness for formal bindings with domain specific tools	-	+	+	+	?	?	++	+++
Ease of use / learn	++++	+++	-	-	+	?	++	+++

Table 2.2: Summary table of requirements languages and tools from the state of the art

2.6 Focus on justification frameworks and traceability

In order to propose a design methodology that allows, at any point in time, to verify that objectives and requirements are met, and to rigorously demonstrate that the reasoning chain of the design process is dependable, a focus is made on traceability and justification frameworks, in particular Claim, Argument, Evidence (CAE) [15][94]. The notion of justification frameworks, introduced in [55] and [116], concerns approaches that highly focus on capturing and structuring the reasoning chain that justifies how high-level requirements will be met by a system based on pieces of evidence.

Before addressing how these frameworks and traceability can support continuous V&V and dependable systems, let's introduce an overview of the concept of traceability from the literature point of view.

2.6.1 Traceability

Traceability refers to means by which we keep track of how was implemented a high-level objective or requirement into all development phases of a system [116]. The concept of traceability essentially aims to ensure that all system requirements are addressed and taken into consideration in the design by means of links (or allocations). This does not mean that they are systematically satisfied by the designed system, it only means that requirements are processed, linked to sub-elements.

Traceability can also be seen as a means to structure the progressive development of a complex system, and also as a means for capitalization of the acquired know-how by engineers during this phase, especially for long-lived systems such as ME-CPS. During re-engineering phases where modifications are operated into a system, traceability plays a major role in keeping track of the rationales behind every design element as well as the hypothesis of the system. Traceability aims to ensure a smooth transformation on systems as it makes it easier to identify the impacts of changes in regards to the existing elements, and also to make sure that all the impacted ones will be suitably treated. A good implementation of traceability in design processes can be a considerable time saver.

It is to be said that one of the most known languages providing traceability features is SysML language [119]. In the latter, the traceability is supported by three possible representations: graphical, tabular, and tree-shaped. SysML provides different kinds of links between its components: (i) links between requirements such as containment, derive, copy or trace and (ii) links between requirements and implementations such as satisfy, verify, and refine [52], and (iii) links between implementations from different levels (functional, logical, physical, etc.). Furthermore, in SysML, one can define and allocate test cases to each requirement or a set of requirements. These test cases will be the pieces of evidence proving that the requirement(s) is (are) satisfied or not.

However, ensuring traceability with links is not always sufficient to justify the consistency of the design process and keep track of the rationales behind choosing a solution and decision making in general [116]. It is particularly important for systems having safety-critical functions and requirements where the correct operation must be demonstrated all along the system lifetime. Justification methods are also essential for systems with long-lived lifecycles, especially in phases where designers are no longer in charge, and several engineers succeed to one another on the same system. Capitalization of rationales is crucial for further system changes and renovation [116].

In the current practice, when a property such as a requirement is refined by other requirements or allocated to a function or a system, they are connected using traceability links. However, no element allows justifying this relationship between them. Moreover, there is no stringent property that guarantees that the transition between properties from different

refinement levels is consistent (e.g. when doing decomposition, we need a property that guarantees that the sum of the sub-elements is equivalent to the high-level property, and does not lead to an undesired emerging behavior).

SysML, as a language, can be indeed supported by a methodology with means to justify traceability links and to structure the justification processes followed. However, the methods that were identified in the state of the art such as [112] are limited to establish derive or justify links across different components and diagrams. Moreover, SysML presents a lack of rigorous means to verify the consistency of the design by verifying the correctness of refinement steps across different modeling layers.

T.Nguyen has presented in [116] an advanced traceability methodology including the notion of justification framework

2.6.2 Claim - Argument - Evidence (CAE) - justification framework

2.6.2.1 Overview

As mentioned earlier, the notion of justification framework [55] [116] refers to structuring and capitalizing the reasoning chain that has been followed throughout the design process of a complex system. The goal is to ensure that the argumentation and the reasoning chain provide a sufficient confidence level that guarantees that the system meets its high-level missions and requirements.

In complex systems, justifying a high-level property such as “the system must be safe” requires a very complex argumentation chain and a large number of factual pieces of evidence from different natures (simulations, demonstrations, testing, statistical analysis, etc.). The problem faced in old and today’s practices is that the reasoning chain followed throughout the design is not well documented through deductive and logical steps [58]. A well-defined argumentation process is needed in order to capture the whole justification chain without any missing links.

An overview on the state of the art works dealing with confidence in decision making has conducted to identify what is called “Assurance (security and safety) cases” [58], which are defined as “*documented bodies of evidence that provide valid and convincing arguments that a system is adequately dependable in a given application and environment*”, referring to Y.Matsuno in [106].

Other techniques were also identified which mainly focus on the argumentation of systems safety properties such as the Goal Structuring Notation (GSN) [93], and the Claim, Argument, Evidence (CAE) practices which are highly recommended by safety-critical systems regulators [106].

In the sequel, we will focus on the CAE approach that was subjected to standardiza-

tion works. One can refer to the system and software assurance and engineering standard ISO/IEC 15026-3:2015 [41], and to the Object Management Group (OMG) work [32].

CAE was used as a justification framework in the European project HARMONICS [16] that addresses software systems performing safety-critical functions from the highest category. The project deals with three design concepts: V&V, safety justification and qualitative evaluation of reliability

Even-though the concepts called Safety justification frameworks introduced in this project were first developed to justify the safety of software systems, however, they turned out to be more general than that, and can be further applied to disciplines other than safety. Applications in various fields can be found in [15].

2.6.2.2 CAE framework

The CAE approach is made of three main elements that are defined in [44] as follows:

- **Claims**, *“which are assertions put forward for general acceptance. These are typically statements about a property of the system or some subsystem. Claims that are asserted as true without justification become assumptions and claims supporting an argument are called sub-claims.”*
- **Arguments**, *“which link the evidence to the claim. These are the “statements indicating the general ways of arguing being applied in a particular case and implicitly relied on and whose trustworthiness is well established”[136], together with the validation for the scientific and engineering laws used. In an engineering context, arguments should be explicit.”*
- **Evidence**, *“is used as the basis of the justification of the claim. Sources of evidence may include the design, the development process, prior field experience, testing (including statistical testing), source code analysis or formal analysis.”*

In this framework, stakeholders start by making claims that are assumed to be correct (e.g. that the system complies with a specific requirement), and that will be further justified using pieces of evidence that are supported with arguments. The claim has value only through the appropriate pieces of evidence supported by arguments that justify the viability of the claim.

These elements are captured using the graphical representations that are shown in figure 2-7.

Throughout the design of complex systems, a high-level claim generally goes through several refinement steps before being decomposed into more manageable sub-claims. CAE approach introduces the notion of “Side-claim”, which is a type of claim that plays a major

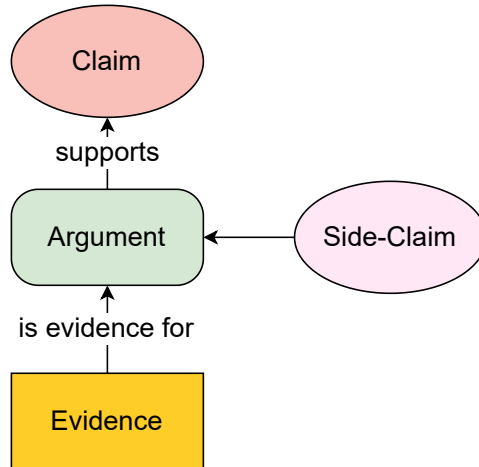


Figure 2-7: Basic graphical elements of CAE (from [116])

role to justify that the argumentation itself is consistent. To do so, side-claims rigorously define properties on the consistency between different refinement levels to verify if they are legitimate. These properties depend on the type of argumentation used for the refinement between two levels. An example of the use of side-claims is shown in Figure 2-8. The side-claim property states that the decomposition is consistent when the conjunction of the sub-claims ensures the top-level claim. Based on empirical analysis of safety cases, five kinds

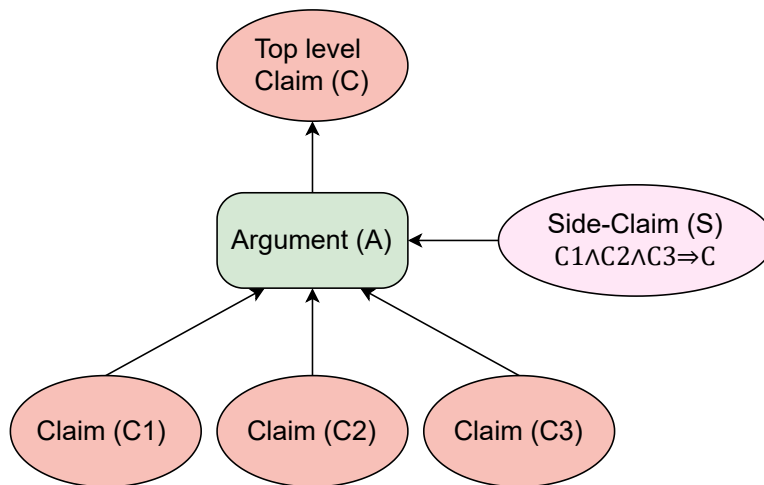


Figure 2-8: Basic graphical elements of CAE (from [116])

of argumentation building blocks were detailed in [57]. A concise definition for each of these blocks is as follows :

- **Concretion blocks:** “*This block is used when a claim needs to be given a more precise*

definition or interpretation. This is often the case of top-level claims, which generally need to be expressed in more measurable, less abstract, terms. For example, a claim on a system's dependability can use a concretion block to introduce sub-claims for each of the dependability attributes. Although this kind of block is necessary, it needs to be clearly identified since it breaks the reasoning in the claim tree..”, taken from [44].

- **Substitution blocks:** *“Another common type of claim expansion involves transforming a claim about an object (or property) into a claim about an equivalent object (or property), which can be viewed as a form of substitution. For example, one might claim that the designed system has a certain property, and therefore the production system has this property too, assuming that the production system is equivalent in some clearly defined way to the designed one. Another example would be the substitution of an executable software code with a source code, as long as the translation of the source code into executable software does not introduce effects that would invalidate the argument built on the source code.”, taken from [44].*
- **Decomposition blocks:** *“A decomposition block is used to claim that a conclusion about the whole object, process, property, or function can be deduced from the claims or facts about constituent parts. More formally, this block is used to show that property $P(X)$ of object, function or process X can be demonstrated by reference to properties $P_1(X_1) \wedge P_2(X_2) \wedge \dots \wedge P_i(X_n)$ of its sub-objects X_1, X_2, \dots, X_n from which it is composed. The sub-objects can be artefacts, processes, environments, configurations, functions, organisations etc. ”, taken from [57].*
- **Calculation blocks:** *“This block is used to claim that the value of a property of a system can be computed from the values of related properties of other objects (e.g. its subsystems). One application of the block is to provide a quantitative argument when the value of one property can be calculated from the values of other specific properties. For example, the availability of a system can be calculated from its reliability and its failure recovery time. As another example, the average time of data retrieval from a database can be calculated from the probability that the data are in the cache and the time of data retrieval if they are not in the cache.”, taken from [44].*
- **Evidence incorporation blocks:** *“This block is used at the edge of the CAE structure to incorporate evidence into the assessment. It is used to demonstrate that a sub-claim is directly satisfied by its supporting evidence.”, taken from [44].*

The CAE approach was integrated as part of CReMA methodology in order to support the design process by structuring and organizing the reasoning chain followed by stakeholders. A connection between CAE approach and some engineering practices has been made in

this thesis and will be introduced later in chapter 3. The idea is to ensure an evolutionary approach that goes from requirement engineering at early design phases to detailed analytical phases where engineers use disciplinary design and modeling tools. This approach will allow on the one hand the stakeholders to design their complex systems using state-of-the-art means for reasoning on complex systems, and on the other hand to guarantee to other stakeholders the viability of the contract that is established with them through justifications using a solid and well-established argumentation. This is essentially meaningful when establishing contracts with safety authorities that require stringent pieces of evidence to justify the right operation of safety systems. Last but not the least, this argumentation can also be of major added value during negotiations between stakeholders. A stakeholder can for example demonstrate using this approach that to guarantee a level of system performance, the other stakeholders will have to make compromises in regards to time delay or cost.

2.7 Systems engineering methodologies for complex systems

2.7.1 Overview

This section of the state of the art was done to identify SE methodologies that are used for the design of large complex systems and cover all stages of their lifecycles. A focus was made on V&V dimensions of the methodologies as it is an essential asset to overcome challenges facing the design of ME-CPS (important short and long terms uncertainties, a large number of scenarios, need to coordinate stakeholders using rigorous and systematic techniques).

SE methodologies come to support and assist engineers in designing complex systems. They aim at proposing guidelines for designers in order to use platform implemented processes emanating from SE standards without being overwhelmed by their complexity. Indeed, a key concept for the success of a methodology is its capability to hide the complexity without affecting the pertinence of the design by using guidelines and refinement steps for developing the optimal system.

The methodologies found in the state of the art address SE processes in different manners depending on the available tools supporting them, and also on what parts of the systems engineering cycle are in the scope (preliminary studies, requirement engineering, architecture design, detailed design, etc.). As a matter of fact, few existing methodologies manage to expand their scope to deal with all SE processes, and rare are the ones that respect all SE recommendations.

2.7.2 Comparison and evaluation of existing methodologies

A large panel of SE methodologies was explored in literature and evaluated on the basis of multiple criteria. The full criteria list with the meaning and the quality level of each one of them is reported in Appendix B. Among them, we have chosen the following ones for comparison matters in the current section: (i) the appropriateness for modeling physical aspects, (ii) the formal handling of requirements, (iii) the capacity of making automatic verification and validation, (iv) the openness for pairing with other tools and platforms dedicated to specific domains, (v) the license terms and (vi) the ease of use. This section was a subject of a conference paper [47] that was published in the framework of this thesis presenting a survey on SE methodologies for the design of ME-CPS.

In this section, and to be more concise, we have restricted the list of methodologies studied and we have classified them into 4 categories. We have chosen the ones presented in this survey as a result of the following reasoning:

We have first started by exploring Product Life Management (PLM)-based methodologies such as System Driven Product Development (SDPD) [33] proposed by Siemens PLM Software or Requirement, Functional, Logical, and Physical (RFLP) approach [69] that were the most likely to answer our problems as they are SE software leaders and incorporate multi-domains tools. We have then oriented our focus towards Model-Based Systems Engineering (MBSE) methodologies based on languages such as SysML [109] and Capella [128], which are used in multiple industries. However, in both PLM and MBSE, we have noticed considerable limitations concerning the formal handling of requirements, and thus automatic verification and validation. Moreover, we observed a wide gap between the early design phases such as requirement capture and the subsequent detailed design phases when disciplinary modeling and simulation are involved. We have then examined knowledge-based methodologies such as SysDICE [68] or SLIM [50] that propose formal methods that narrow the gap between early and detailed design and enable creating links between different tools.

In order to deepen our knowledge concerning formal requirements and automatic verification and validation, we have focused on control-like methodologies that were inspired from software design and use similar paradigms such as B method [66] and Contract-Based design [117] [53]. However, these approaches are limited when dealing with physical systems such as ME-CPS.

An overview of each of these approaches is given in the sequel. A more exhaustive list of SE methodologies as well as a global summary table can be found in Appendix B.

2.7.2.1 Teamcenter / SDPD approach

Teamcenter [34] is a PLM platform from Siemens PLM Software that is widely used in industry for handling complex systems over different steps of their lifecycle from design to disposal phases.

Teamcenter aims to go beyond the technical processes of SE and incorporate in the same platform the technical management processes, agreement processes, and organizational project-enabling processes. In our thesis, we will only focus on the technical processes of the platform.

Siemens PLM Software developed an approach called System Driven Product Development (SDPD) [33] that deals with the technical processes of SE and specifically the design part of the PLM platform. SDPD orchestrates system design by providing analysis tools, guidelines, diagrams, and views for design phases such as requirements definition, functional decomposition, organic allocation, traceability, physical definition, etc. all implemented in the Teamcenter platform.

The SDPD approach has the particularity of capturing requirements using two ways in a single framework by combining the expressiveness of the natural language paradigm with the rigor of semi-formal requirements integrated into a table with attributes. A very interesting aspect of the current methodology concerns the traceability features that track system design evolution throughout its lifecycle. Moreover, impact analysis with automatic reporting is proposed by the platform that hosts the methodology.

However, this methodology does not offer automatic verification of requirements. Furthermore, it has a major limiting aspect that concerns the commercial aspect of the Teamcenter platform hosting the methodology which requires important investment for industries.

2.7.2.2 3DEXPERIENCE platform / RFLP methodology

3DEXPERIENCE is a PLM platform developed by Dassault Systemes (DS) [69]. It gathers all Domain-Specific Languages (DSL) proposed by DS in order to bring together different views of a system and provides a global vision as well as enhanced analysis possibilities. Platform applications are separated into four categories: (i) Social and collaborative applications, (ii) 3D and Model-based applications, (iii) Simulation applications, and (iv) Information intelligence application.

The platform ensures data management functionalities across multiple domains and possible stakeholders. All systems and projects data are saved in a unified database hosted in the cloud, with different access capabilities depending on the roles attributed to the stakeholders. Data management ensures unique and updated information to all stakeholders.

The RFLP approach is based on the V-shaped cycle presented in figure 2-9.

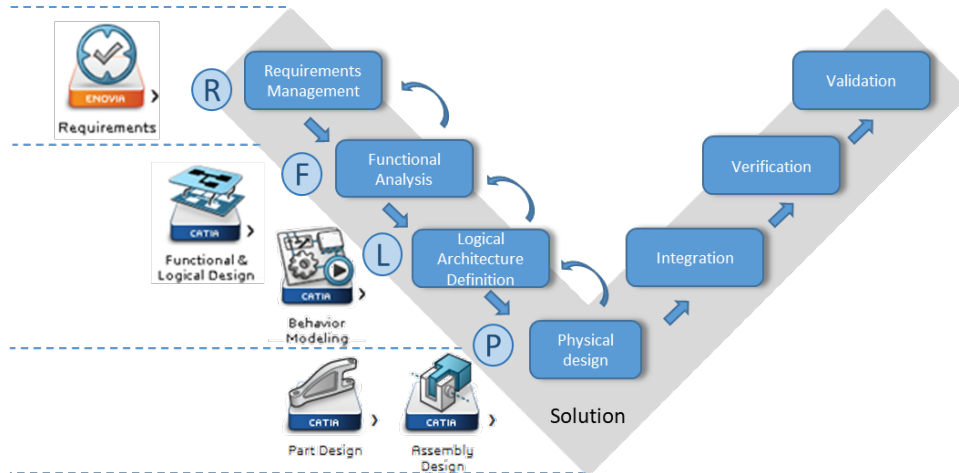


Figure 2-9: RFLP V-model and tools used for each view

RFLP [69] is introduced as a framework supporting SE approach by integrating its processes into a unified platform using four fundamental views: requirement, functional, logical, and physical views. RFLP is supported by the 3DEXPERIENCE platform with its integrated DSLs for representing four main views as well as for ensuring cross-views relationships and traceability. It consists of the following steps:

- Requirements are first written in natural language in the “Requirement” application. They are characterized using the attributes: type, state <active or inactive>, <locked, unlocked>, difficulty level, priority, and classification. For each requirement, attributes and a test case with the conditions under which the system will be validated shall be provided.
- The functional architecture of the system is then created by decomposing functional requirements using sub-functions and flows that the later defined components will exchange. The functional architecture view is similar to the Activity Diagram used in SysML.
- Create a logical architecture that defines how the system shall perform in order to accomplish its missions/functions. At the top level of the logical architecture, relationships, and interfaces between systems components are defined. At lower levels, a behavioral model is created for each component in the behavioral application (this application has the same features as Dymola).
- Finally, after doing system analysis using platform integrated tools and developing a 3D digital mock-up, a relationship between the 3D design and logical design can be created in order to visualize a 3D kinematic of the represented behavior.

3DEXPERIENCE ensures a large set of model-based applications and simulation capabilities. However, it does not afford the formal aspect of the methodology for designing large distributed ME-CPS. Indeed, the RFLP approach is limited in terms of methodological guidelines and some technical aspects as it shows important gaps between different design steps such as the absence of explicit links between requirements, functional and logical (behavioral) architectures. Technically speaking, using non-formal requirements is a limitation for automatic verification and analyzing modifications impacts.

2.7.2.3 SysML-based methodology

In [109], Mhenni et al. presented a SysML-based methodology that affords a holistic modeling approach and enables encompassing several aspects of a system over its lifecycle using graphical representations and views with an integrated syntax. It allows formal mechanisms for organizing and handling data over different views representing the same system.

The methodology is based on a functional analysis and is decomposed into two complementary phases, a black box analysis to produce from scratch an exhaustive book of specifications, and a white box analysis for system design with the possibility of having concurrent architectures (functional and physical). However, SysML can only model system behavior using sequences of events or state machines. Continuous-time dynamics are supported neither by the language nor by the methodology. Requirements are captured using natural language which leaves space for ambiguity and cannot be executed for automatic verification, even if connections between requirement views and other diagrams are possible.

SysML is based on a functional decomposition of requirements and allocations into organic components. Consequently, real-time physical aspects cannot be supported with this approach. With these limitations, a wide gap can be observed between the early system analysis phases and modeling and simulation phases.

2.7.2.4 Capella / ARCADIA methodology

Capella is an MBSE open-source platform inspired by UML/SysML. It was developed to cope with the difficulty that engineers with no software background face when using generic languages such as UML/SysML. Capella is supported by graphical tools for systems modeling and provides processes in accordance with the ARCADIA method.

ARCADIA refers to the Architecture Analysis and Design Integrated Approach, which is an engineering method centered on architecture decomposition and model-driven design. It is based on a set of functional decomposition processes and allocation of functions to components. It supports preliminary system analysis consisting of capturing and decomposing stakeholders' needs following a set of processes that enables constructing multiple views and

architectures of the system.

ARCADIA presents four main steps for system design:

- **Operational Analysis:** Define stakeholders' needs using textual expressions and identify environmental entities, actors, activities, and roles using specific diagrams.
- **System Analysis:** Identify system boundaries and define what the system should accomplish using functional decomposition and dynamic behaviors (limited to sequence diagrams and state machines). This phase is referred to as "Requirement formalizing" in the documentation.
- **Logical Architecture:** Define how the system will work in order to achieve expectations. Build architectural breakdown candidates and choose the best compromise.
- **Physical Architecture:** Define how the system will be developed and built. Considering the reuse of existing elements, design physical architectural reference, and validate it.
- **Development contracts:** Define what is expected from each developer, how verification and validation will be done, and what integration strategies will be used.

The ARCADIA method obviously catches up with SysML-based methods in diverse aspects. They both explicitly separate black box and white box analyses. Both also capture requirements using informal language and both are based on functional decomposition. Finally, they both model system behavior using finite state machines.

ARCADIA and SysML-based methodologies allow good handling of complex systems and especially ease and guide subsequent integration of subsystems. However, non-formal requirement handling, functional decomposition, and limited modeling features for physical aspects are not adapted for ME-CPS.

2.7.2.5 Combining SysML/UML to Modelica

Different works were found in the state of the art aiming at combining both benefits of SysML and Modelica into tools such as ModelicaML [130], SysML4Modelica [122], etc. These works aim to combine the descriptive power of SysML/UML using diagrams and graphical representations with the formal language Modelica that allows to model dynamic systems using differential equations and discrete events as well as rigorous analysis capabilities. ModelicaML was created in order to generate executable Modelica code from SysML graphical representations.

Modelica and SysML/UML are both object-oriented modeling languages [131]. They both aim at modeling a system with objects but use different paradigms for describing their internal behaviors. SysML focuses on capturing (non-formal) requirements, uses cases,

functional and organic architectures with the possibility to integrate context elements, etc. Modelica on the other hand uses graphical icons and connections together with an integrated textual language for modeling executable behavior which enables efficient numerical simulation. Combining Modelica and SysML as presented in [122] is done by first extending SysML into SysML4Modelica in order to generate Modelica constructs using SysML extensions. The concept is first to create the SysML model as usual, then to choose the part that needs to be analyzed using Modelica, and finally to apply the SysML4Modelica profile on it to generate an analytic model that is translated into a Modelica model to be simulated.

The development of SysML4Modelica was based on finding the similarities in terms of constructions between Modelica and SysML, extensions of the latter were done in order to have as many similarities as possible.

However, developing this methodology turns out to be very complex as ModelicaML is not able to support the complete Modelica specifications, and at the same time, it only uses a limited set of UML/SysML functionalities [130]. Other issues need to be solved before being able to use these methods for encompassing the complexity of CPS.

2.7.2.6 Knowledge-Based Engineering (KBE)

Knowledge-based methodologies were introduced in order to deal with the interactive process between engineers and tackle the continuous change of parameters in different system disciplines that use heterogeneous models and DSLs. They afford means to capture and capitalize data in a structured way for synchronization purposes between stakeholders [107].

These methodologies generally use SysML as a central language affording a multi-disciplinary architecture that gathers different system views and components. Methodologies such as SysDICE [68] and SLIM [50] were developed to reduce the wide gap between SysML[119] descriptive architectures and DSLs for each system facet.

The SysDICE [68] methodology introduces means to extract formal mathematical data from SysML representations by using profile views that were specifically developed for this purpose. Stereotypes were added to the existing SysML requirement block by including numerical attributes such as a value and a priority. These stereotypes are considered as the evaluation objectives information. The methodology introduces a tool adapter used to link formal extractions with engineering-specific tools for further analysis.

The SLIM [50] approach proposes a similar paradigm compared to SysDICE. It consists in adding a layer to SysML in terms of relationships with DSLs such as DOORS, CATIA, Matlab, Amesim, Teamcenter, etc. as shown in Figure 2-10.

The SLIM approach is not a design methodology as it does not propose any guidelines for conceiving systems. Instead, it proposes a framework that gathers different systems facets and components with different views and different tools, and ensures features to manage ar-

chitectures as well as synchronize data between SysML and different DSLs or PLM platforms. Data is centralized in an SQL database.

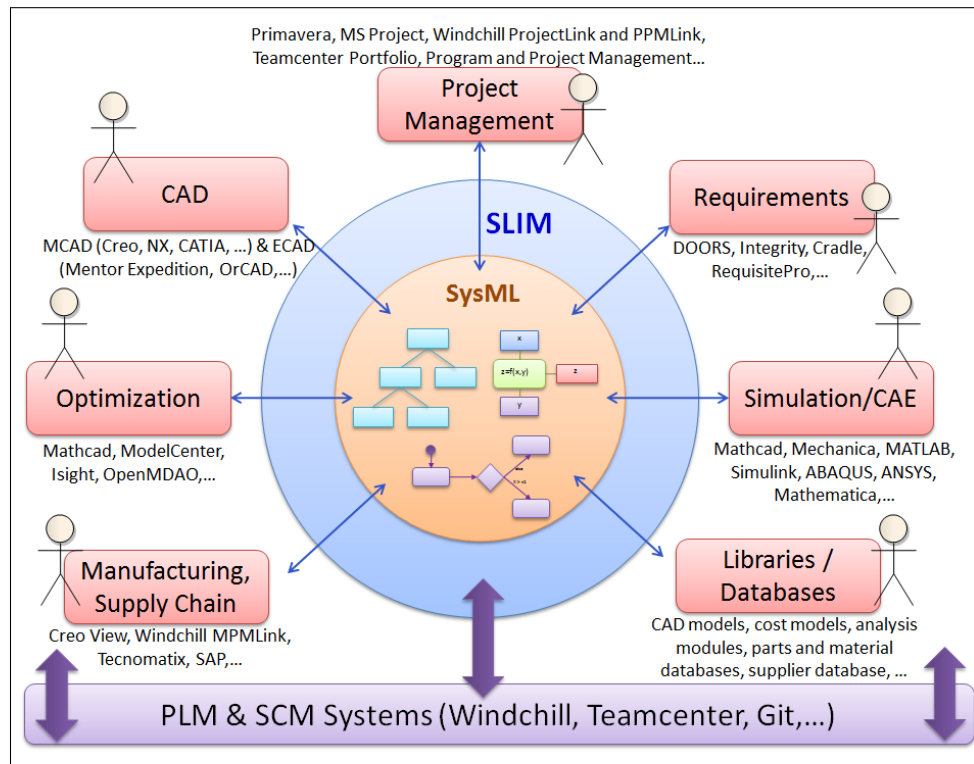


Figure 2-10: SLIM methodology framework (from[50])

This approach has partially succeeded in reducing the gap between SysML and other DSLs as it gives the possibility to connect the non-executable SysML diagrams to executable ones in order to feed them with the latest configurations. However, SysML limitations can negatively impact the design of the kind of systems we aim to develop (ME-CPS). Other works related to knowledge-based are worth to be mentioned, in particular MOKA [118] (Methodology and tools Oriented to Knowledge-based engineering Applications), COLIBRI [96] (Constraint Linking Bridge), and KCM [110] (Knowledge Configuration Model).

To sum up, knowledge-based methodologies propose means to automatically check the consistency and conformance of system key variables computed in different DSLs with the desired properties. The main limitation of these methodologies concerns the types of requirements and properties that can be checked. Indeed, they are limited to checking arithmetic constraints with fixed values, and cannot encompass requirements that are expressed in terms of probabilistic aspects, state machines, properties requiring mathematical transformations, etc. Moreover, they do not offer any guidelines to support the overall system design process. They only propose the framework enabling the handling of the interactions and coordination

between different disciplines and engineers.

2.7.2.7 B method

B is a method for designing software using mathematical proof and successive refinements of specifications in order to develop a correct by construction product. B method introduces automatic refinement processes to get over the time consuming manual programming [66]. The automatic refinement allows rapid software development by providing the means to automatically translate an abstract B model into an implementable one written in ADA or C++. This is done through an iterative process in which a high-level state machine is transformed into a set of state machines based on a set of refinement rules that are verified at each iteration.

B aims at developing correct by construction software by the formal modeling of the system (hardware and software), its environment (other systems, infrastructures, etc.), the properties to be satisfied, and simplifying code production. The key of doing so is first formally capturing the specifications and applying multiple successive refinements steps with verifications and approval procedures for each one of them until a concrete level similar to code for B and components (including I&C) that can be implemented is reached. Figure 2-11 gives an overview of the global process of B method.

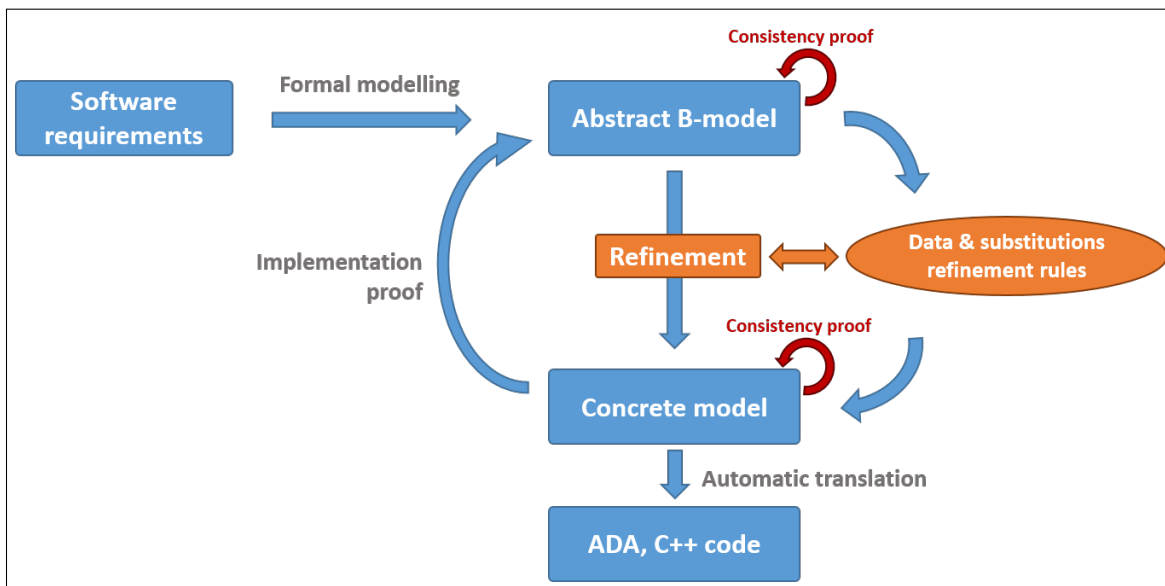


Figure 2-11: B method for software design

Taking a look at B is interesting for our work even if it is only appropriate for software development because its design operation mode is similar to ours, albeit applied to a different product. However, it is not appropriate for CPS because it is especially suited for the software

world where manufacturing costs are absent, where there is no system wear, and where there is always the possibility of duplication at zero cost. Moreover, this paradigm requires a strong mathematical background and a precise knowledge of formal languages with special rigorous syntax and semantics. Also, B method only deals with systems modeled using finite state machines, which narrows the scope of system properties that can be represented, especially for systems including physical aspects.

2.7.2.8 Contract Based Design (CBD)

CBD, also known as contract programming is an approach that was and is still essentially used for software design but has been adapted in the last years to the formal modeling of CPS. The term “contract” comes originally from the conceptual idea of business contracts with conditions and obligations between a client and a supplier. It proved to be beneficial for structuring the communication and relationship between what is needed and what is provided.

Historically, the design by contract was first introduced by B. Meyer in [108] in order to put an emphasis on the quality of programs and to develop bug-free software. B. Meyer integrated this paradigm into the programming language Eiffel [14]. In this paradigm, a contract for a given module is defined as a couple (precondition, post-condition). The precondition states the prerequisites to be satisfied by the inputs of the module. The post-condition specifies the conditions to be satisfied by the outputs. The contract states the obligations of the module with respect to its environment, and the obligations of the environment with respect to the module. For instance, let’s take as an example a function that computes $y = \sqrt{x}$, the precondition (or assumption) of the function that needs to be processed before making any operation on the input x is: “ $x \geq 0$ ” and the post-condition (guarantee) to be checked after executing the function computation is $y^2 = x$.

A formal mathematical theory was proposed by Inria in [53] [54] which characterizes contracts using the assume/guarantee paradigm in reference to pre-conditions/post-conditions respectively. This work proposes a unified treatment for Contract-Based Design (CBD) with a methodology having a rigorous algebra for contract abstraction, refinement, decomposition, conjunction, and verification.

CBD consists in adding a formal modeling layer (and sometimes semi-formal but it is not within the scope of our work) that wraps the system’s components using a contract to define specifications on the inputs and outputs [129]. The goal is to make sure that each system element guarantees the mission for which it was conceived and within a defined tolerance. A contract is therefore a form of shared interfaces between system components where they agree on the values of inputs/outputs and variables. CBD approach consists in developing systems through multiple refinement steps going from top-level abstraction to

implementations with low-level components. At each step, the refined contracts are intended to be verified for consistency with previous levels to make sure that the designed system will answer the high-level goals.

Formalizing requirements using contracts at early design stages enables early detection of inconsistencies by explicitly defining the assumptions on the environment and the guarantees of each system component. Once the contracts of all these components are set up, a consistency analysis is done to validate the requirements. The goal of the decomposition process is to reduce the system complexity into more manageable tasks at the component level with a set of assumptions on their operational environment and guarantees they need to provide [117].

CBD can be cost-effective for handling updates and system modifications by avoiding the need of verifying the whole design process after each modification. Indeed, based on compositional and refinement methods, it is sufficient to verify that the new or modified subsystems and components fit into the dedicated contract. The idea is that every decomposition can be verified independently from the rest without having the necessity of knowing the concrete implementation.

In the framework of this thesis, CReMA methodology will be based on the CBD approach which will be adapted to ME-CPS challenges. CBD was chosen for its ability to allow a rigorous definition of the interfaces and relationships between interacting elements. Stakeholders are considered to be the central elements of CReMA methodology, CBD will be thus used to structure the relationships between them and will assist them to converge towards common agreements.

2.7.2.9 A Platform-Based Design Methodology with Contracts and Related Tools for the design of CPS

Nuzzo et al. present in [117] a contract based methodology for CPS design in order to create components abstraction and handle their interactions while ensuring a formal support during all design phases. The design is performed under successive refinement steps of system specifications until low-level components implementation.

The methodology starts with formalizing top-level requirements using contract-based design. Top-level requirements are captured using formal automata paradigm and integrated into contracts in terms of assumptions and guarantees. At this design phase, the system behavior is not yet modeled even though requirements consistency checks can be launched. Consistency checks allow to detect conflicting requirement properties and thus give an idea about the feasibility of the system while avoiding considerable specification errors.

Based on top-level requirements, different types of contracts can be created. When dealing with system architecture specifications, static contracts can be used in order to integrate

discrete or continuous properties of the system using mathematical constraints on Boolean, discrete or continuous variables representing static geometric or interconnection properties. Otherwise, when coping with control algorithm specifications, the dynamic behaviors are captured in contracts using temporal logic. Different languages can be used depending on the behaviors to be modelled. Linear Temporal Logic (LTL) is a logical formalism that handles time as a succession of discrete states in a transition system. Signal temporal logic (STL) extends temporal logic by giving the possibility to handle continuous-time signals. STL was introduced as being more adapted to CPS where discrete and continuous variables are inevitably present and tightly connected (such systems are called hybrid systems).

The main idea of the methodology is to separate out at the early stages the system architecture constraints from the control specifications, each part being integrated into two different contracts. When structuring system decomposition into architectural design and control design while verifying compatibility and refinement properties with top-level contracts, the system can be developed in a compositional way while guarantying that it can be correctly assembled. This methodology catches up with most of the existing formal methods for handling CPS design that represents systems and their requirements using state machines and uses a model checking approach for verifying system behavior against desired properties.

2.7.3 Summary and discussions

Comparison criteria	Appropriateness for physical systems	Requirements formalizing	Automatic testing and V&V	Openness for pairing with other tools	License terms	Ease of use
Methodologies						
TeamCenter / SDPD	Yes	Semi-formal	No	Yes	Commercial	Middle
3Dexperience / RFLP	Yes	Semi-formal	No	No	Commercial	Middle
SysML based methodology	No	Semi-formal	No	No	Open source	Middle
Capella / Arcadia	No	Semi-formal	No	No	Open source	Middle
Combining SysML/UML to Modelica	Yes	Semi-formal	No	No	Open source	Middle
B Method	No	Formal	Model checking	No	Open source	Complex
Contract-based design	No	Formal	Model checking	No	Open source	Complex
Knowledge-based methodologies	Yes	Formal	Limited simulation	Yes	Open source	Middle

Table 2.3: Summary table of methodologies from the state of the art

Table 2.3 presents a summary of the methodologies found in the state of the art. This

work is far from being exhaustive but gathers commonly used and known practices in the industrial and academic worlds. Each method has its pros and cons.

In the literature, different Systems Engineering communities proposed a large number of MBSE (Model-Based Systems Engineering) methodologies. These methodologies are based on general modeling languages such as SysML [109] [119] or Capella [128] that are used for their descriptive power and expressiveness by means of standardized graphical representations which afford a holistic system view encompassing different aspects of a system over its lifecycle. However, these approaches present some limitations. First, requirements are captured using natural language, which leaves space for ambiguity and does not allow them to be executed for automatic verification and validation (V&V). Secondly, MBSE methodologies are based on functional and logical decomposition of requirements, followed by allocations of organic components. Consequently, systems behaviors are only modeled using sequences of events, state machines or parametric diagrams. Neither these methodologies nor the associated languages support continuous-time or dynamic physical aspects. These aspects are thus neglected or at best developed in a separate corner. These shortcomings widen the gap between the early design phases where the logical aspects of the system are analyzed and the detailed design phases where the physical aspects are considered. Finally, the current V&V approaches are generally limited to (i) verifying that requirements are well mapped to functions and components, and (ii) the validation of system design is not based on any domain-specific tool and is limited to state machines modeling. Thus the physical aspects are not well integrated into these approaches.

SE methodologies introduced by the software editors Dassault Systèmes and Siemens respectively named RFLP and SDPD, and which are based on their respective PLM platforms 3DEXPERIENCE and Teamcenter were also studied. They offer collaborative features, effective data management, traceability over the systems lifecycle as well as several analysis options, with the possibility to utilize domain-specific tools for the detailed modeling and simulation of physical systems. However, none of them offer, until now functionalities for the formal handling for requirements, and thus do not allow automatic verification.

During the state of the art, a striking similarity was observed between most of MBSE methodologies. In fact, it is common that they are generally based on several tools that are used at different levels of a system lifecycle, and we have noticed that paradigms that are used in early design phases and advanced design phases are deeply different and heterogeneous. A wide gap can clearly be observed between these two phases[50]. In early phases, most practices use (i) natural language or template languages for their requirement engineering and (ii) general modeling languages such as SysML or Capella in order to analyze at an abstract level the complexity of systems in terms of interactions with their environment as well as their internal interactions. Systems behaviors are limited to state machines representations.

In more advanced phases, engineers use domain-specific languages in order to model and analyze the dynamics of systems. These languages are based on formal modeling paradigms that cannot easily be connected to the ones used earlier for an executable V&V. Figure 2-12 shows an overview of the tools that are used at different stages of a system lifecycle. We emphasize that it is far from being exhaustive.

Some tools have tried to bridge this gap by proposing transformation mechanisms between tools such as ModelicaML that connects SysML and Modelica. However, it had mitigated success, which we suppose is due to the lack of methodological aspects that would support connecting these two tools. Knowledge-based methodologies on the other hand have tried

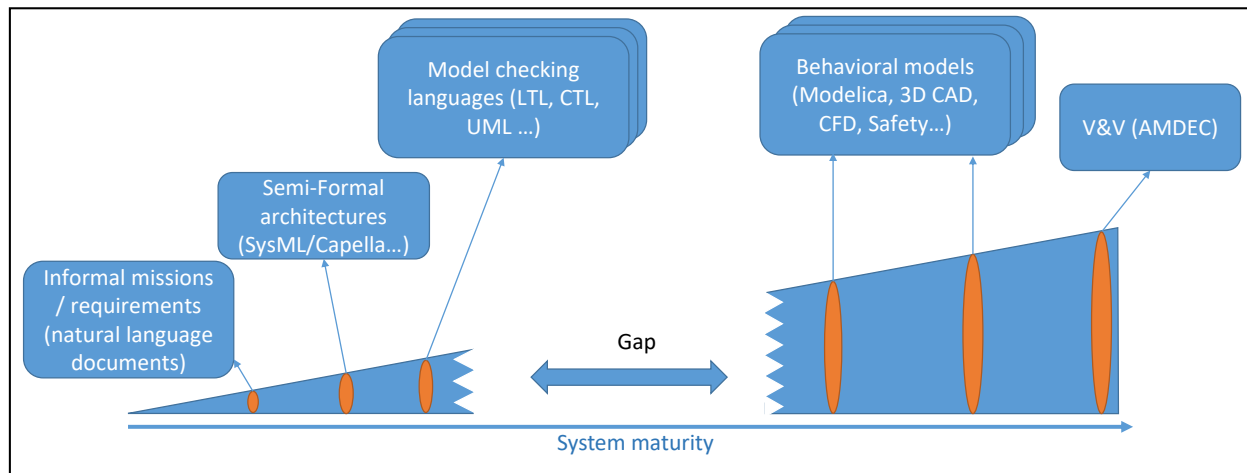


Figure 2-12: Modeling techniques used at different levels of system maturity

to narrow this gap by introducing means to connect analytical tools used at early conceptual phases such as SysML with domain-specific languages. They enable to automatically check the consistency and conformance of system key variables and properties between different DSLs. The main limitation of these methodologies concerns the types of requirements and properties that can be checked. Indeed, they are limited to checking arithmetic constraints with fixed values, and cannot encompass requirements that are expressed in terms of probabilistic aspects, state machines, properties requiring mathematical transformations, etc. Moreover, they do not offer any guidelines to support the overall system design process. They only propose the framework to handle the interactions and the coordination between different disciplines and engineers.

Both contract-based design approaches proposed by Inria [54] and the University of California [117] are rigorous methods aiming at developing correct by construction systems. They offer deductive guidelines and verification features that help to check the correctness of system design at different abstraction levels. However, these methods are only used in specific applications in software design or small mechatronic systems. They are not used

by engineers working on large physical systems. This stems from two main reasons. First, using this paradigm requires a strong mathematical background and a precise knowledge of formal languages with special rigorous syntax and semantics. Second, existing formal approaches only deal with systems modeled using finite state machines, which narrow the scope of system properties that can be represented. In particular, the physical aspects are restricted to their logical representation, this being contradictory with their essential continuous real-time nature potentially involving a very large (quasi-infinite) number of states. In fact, no method can deal with an infinite number of states, but finite-state machines are not suitable to represent the evolutions of continuous-time dynamic systems that are better described using algebro-differential equations (DAEs), which are properly discretized using advanced numerical methods.

Chapter 3

Complex systems design – a new methodology

This chapter introduces the new CReMA methodology developed in the framework of this thesis. In the following, we start by giving a brief recall of the methodology objectives and assumptions. Secondly, we give an overview of the prior work underlying CReMA methodology which concerns the formal verification of requirements. Then, an abstract overview of CReMA methodology levels is given followed by an exhaustive and detailed introduction of the formal objects and concepts composing the different levels. Later, CReMA methodology is exposed with more details. A schematic form of the information model gathering and organizing the interactions between CReMA’s concepts is presented. Finally, a proposed unified characterization of CReMA methodology levels is presented and applied level-wise.

The methodology introduced in this chapter has been submitted for publication in the IEEE Systems journal with the following title : “A Model-Based Engineering Methodology for Stakeholders Coordination of Multi-Energy Cyber-Physical Systems” [48].

3.1 Methodology objectives

We recall in this section the objectives of the Common Requirement engineering Modeling Approach (CReMA) methodology, which were stated in section 1.3.

This work aims at proposing a co-design methodology that is motivated by the fact that a *large number* of stakeholders are in constant interaction for the design and operation of complex systems such as ME-CPS. When many stakeholders with different objectives, perspectives, and culture are involved, it is difficult, and often impossible, to reach a common agreement on the purpose of the system and its scope *before* starting to work on the system. This is in stark contrast with the classical systems approach where the system goals and

scope are considered to be defined first.

The purpose of this thesis is to propose a framework where the design of a complex ME-CPS will emerge throughout the negotiations between stakeholders, each stakeholder putting forward its own objectives and interests, but being open to negotiation in order to reach a common agreement with other stakeholders. The difficulty of reaching an agreement highly increases with the number of stakeholders (maybe exponentially proportional). These negotiations need to be formalized in contracts, whose realization shape the ME-CPS. Negotiations here do not necessarily mean that stakeholders are in conflict, it may simply refer to discussions between teams for the sake of optimizing the overall system.

Among the main purposes behind the development of CReMA methodology is also preventing stakeholders from over-specifying their requirements regarding others. It is generally due to the poor knowledge that a stakeholder has on his interfaces with other interacting stakeholders. The idea here is to implement techniques allowing a rigorous characterization of the interfaces between stakeholders. The methodology shall overcome challenges facing industrial projects involving multiple stakeholders such as confidentiality and the fact that interacting systems are managed and operated independently by various entities.

To name a few examples in the energy field where CReMA methodology can be applied, we mention:

- The renovation of the energy system of a district or a consumption structure such as a large train station in order to support a higher share of renewable energies with possible self-consumption and smart energy management.
- The sizing and construction of a nuclear plant involving owners, designers, regulators, investors, political entities, etc.
- The development of new strategies of energy management at a large scale involving consumers, producers, suppliers, energy distributors, etc. while adding new uses such as electric vehicles and smart charging.

3.2 Methodology assumptions

In this work, we assume that stakeholders are correctly coordinated and that a common agreement is reached when *formal contracts* are signed between them (formal contracts will be defined later). We consider that stakeholders are willing to negotiate and to adapt the terms of the contracts in order to converge towards common agreements. Stakeholders will be ready to sign when they are sure that they can commit to their obligations as stated in the contracts and that these contracts allow them to fulfill their own objectives.

Furthermore, we consider that stakeholders already have the knowledge and know-how that is required to design the system they are in charge of when they have a well-defined book of specifications. However, stakeholders have coordination difficulties to design and operate inter-connected systems of systems.

3.3 Previous work : Co-design verification approach

CReMA methodology takes as a starting point the formal verification approach introduced by Bouskela et al. in [63]. This work intended to fill the gap between the early conceptual phases where non-formal or semi-formal languages prevail for requirement engineering and the analytical phases where engineers use multiple domain-specific languages for modeling their systems. Figure 3-1 from [63] shows the main concepts that need to be assembled

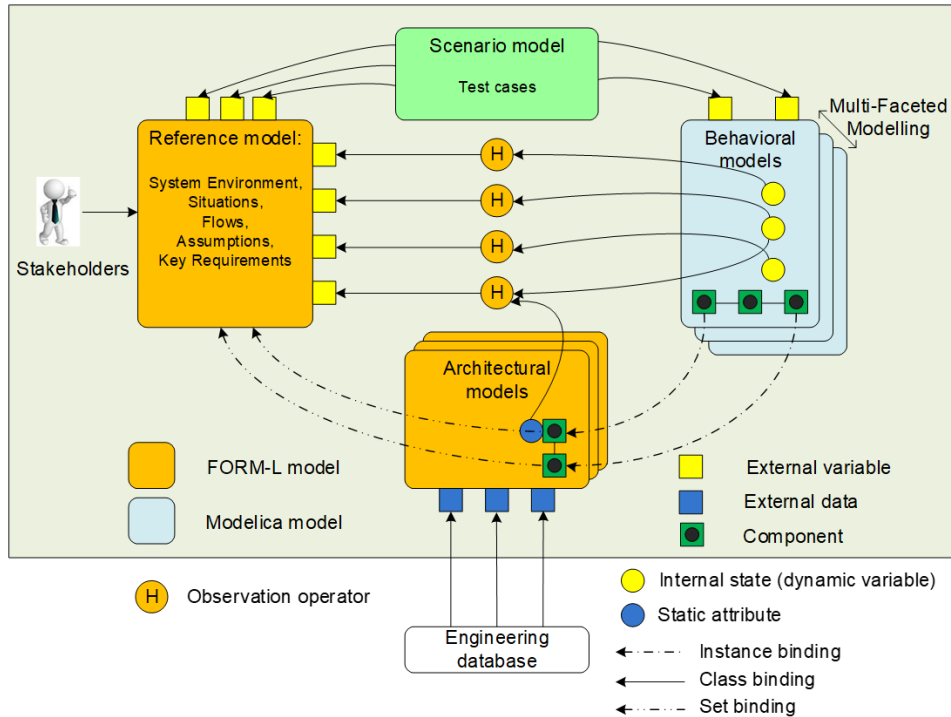


Figure 3-1: Verification model (from [63])

in order to create an executable verification model that can be simulated under multiple scenarios. These concepts are (i) the requirement model, (ii) the architectural model and (iii) the behavioral model. The goal is to challenge the system design against requirements under multiple scenarios and for multiple alternative architectures. These models are connected through the concepts of “bindings” (see section 3.4.8) and “observers” (see section 3.4.7).

CReMA methodology is an extension of the above-mentioned approach and seeks to scale up to ME-CPS involving a large number of stakeholders. As a matter of fact, when dealing with a large ME-CPS, there is no unique book of specifications that gathers all system requirements. On the contrary, each system stakeholder has its own expectations and perspectives in mind, which generally leads to contradictory requirements between system actors. Consequently, the idea of the new methodology is to switch from a vision centered on the system to a vision centered on the stakeholders. This does not mean that the concept of system is dropped. It just means that the concept of systems is driven by the concept of stakeholders.

3.4 Methodology formal objects and concepts

CReMA methodology is built upon six modeling levels as displayed in Figure 3-2. These levels, which will be introduced in detail in Section 3.5, are built by assembling multiple objects such as stakeholder, requirements, contract, etc and concepts such as refinement which are defined one by one in this section.

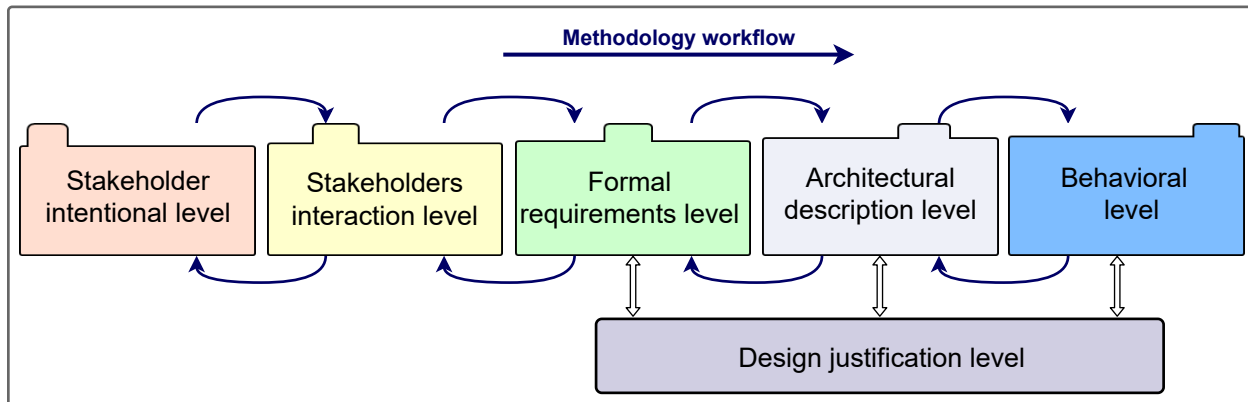


Figure 3-2: Abstract overview on CReMA methodology levels

3.4.1 Stakeholder

3.4.1.1 Definition

Referring to the INCOSE Handbook [138], a stakeholder is: “*any entity (individual or organization) with a legitimate interest in the system. When nominating stakeholders, business management will take into account all those who may be affected by or able to influence the system—typically, they would consider users, operators, organization decision-makers,*

parties to the agreement, regulatory bodies, developing agencies, support organizations, and society at large (within the context of the business and proposed solution).”.

A legal entity or a physical person is thus considered as being a stakeholder taking part of a ME-CPS when it/he has specific intents and goals to be achieved, or interests to be defended. If these intents cannot be satisfied or can no longer be satisfied after being secured, a stakeholder will never engage or won't stay involved in a contract that compels him to commit to other parties' requirements towards him.

Throughout a ME-CPS lifetime and particularly throughout its engineering cycle, several stakeholders join and leave the scope of the system depending on the design phase:

- In early conceptual phases, a complex system boils down to a set of thoughts, objectives, and values to which few if not only one stakeholder is involved. This stakeholder can be a political entity or a company director that aims at realizing his ideas in order to fulfill some specific objectives. The feasibility and viability of the idea are then analyzed using limited resources.
- During design phases, stakeholders having interests in the system progressively join the project such as sponsors, operators, project teams, etc. Other stakeholders are also involved in the design as they represent the laws that should be respected and the norms that should be followed such as legislators, safety authorities, etc. End-users or consumers representatives can also be part of the design.
- During the operational phase, stakeholders such as end-users, suppliers, and sub-systems operators including multiple teams from different possible entities can be involved.
- During the re-engineering or dismantling of a system, the arrangement of stakeholders can widely vary as (i) new stakeholders can be handed responsibilities, (ii) the scopes of the existing stakeholders can be substantially modified, and (iii) some of these existing stakeholders can leave the scope of the system for reasons such as the non-fulfillment of their interests, their inability to perform some required tasks, or even the rise of concurrent stakeholders.

3.4.1.2 Stakeholder in the CReMA methodology

Stakeholder is considered as a central concept of the CReMA methodology and is characterized from different angles. A stakeholder description is thus built by assembling different bricks, each one representing a specific dimension. These bricks are progressively fed with detailed pieces of information concerning the stakeholder and the system under his scope. They are shown in Figure 3-3 and can be described as follows:

- The strategic rationale (SR) model that describes using natural language the main goals and tasks of a stakeholder as well as their decomposition into elementary concepts that will either be guaranteed by himself or otherwise by another stakeholder. The last option will lead to a dependency relationship.
- The internal Key Performance Indicators (KPI) that are considered as properties that are specific to the stakeholder and are not shared with others. They represent the high-level concerns of the stakeholder and are the key elements for making decisions. These KPIs come as a result of the decomposition of the intentional level goals and tasks.
- An architectural part (Arch) that represents the scope which will be affected to the stakeholder from the architectural model of the overall system.
- A behavioral model (Behav) describing the behavior of the elements captured in the architectural model. Different models and tools can be used for modeling different facets of ME-CPS.
- A set of observers (Obs) that allows transforming the system physical behavioral states such as mass flow-rates into functional states that are needed for the internal requirement model.
- Bindings that connect models of different natures in order to exchange information for the purpose of verification using simulation. In this case, it assembles the three elements introduced above (that is the behavioral, the architectural and the KPI parts).

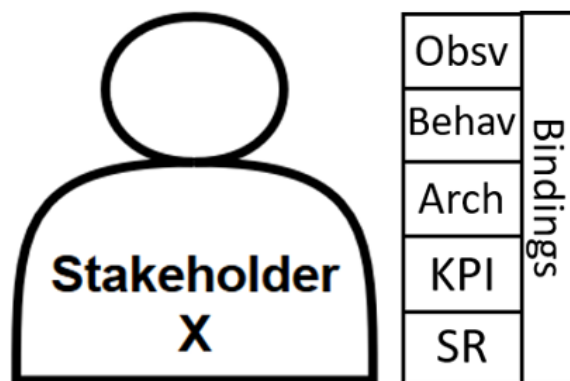


Figure 3-3: Stakeholder description model including five bricks

These elements will be introduced in more detail in the description of the CReMA methodology levels in section 3.7.

3.4.2 Requirements and properties

3.4.2.1 Definition

Properties are spatio-temporal expressions that state constraints on a specific element and can take the values "true", "false", "undecided" or "undefined" in order to estimate the status of a requirement while taking into account the time periods [61] [114]. When a requirement is evaluated in a time period where it is not applicable, the requirement is “undefined”. For example, if the requirement “*When the system Alpha is in operation, the temperature of the water must be higher than 20 degrees Celsius*” is tested in a time period where the system Alpha is not in operation, the decision would be “undefined”. When the decision for knowing whether a requirement is satisfied “true” or violated “false” is still pending, the requirement is then “undecided”. For example, the evaluation of the requirement “*The average electric consumption over one year must not exceed 100kW*” is “undecided” during every moment of that year. Once the year is completed, the decision can be made whether the requirement is “true” or “false”.

Properties can either refer to:

- Assumptions which are assertions that are considered to be satisfied by definition and will not be violated at any moment in time. They can be assimilated to boundary conditions of an item [63].
- Requirements (or “guarantees” inside contracts as will be called in the sequel) which are desired properties that should not be violated by a particular item and need to be satisfied during a specific time period. Requirements can be seen as assumptions in the eyes of the external element(s) that claimed the property[63].

3.4.2.2 Requirements characterization

Bouskela et al. have characterized in [63] properties constraints using four elements:

- **Spatial Locators (WHERE):** refer to the component or set of components that are concerned by the constraint.
- **Time Locators (WHEN):** refer to the time periods or specific instants where the requirement needs to be satisfied. There are two kinds of time locators:
 - *Discrete Time Locators (DTLs):* define specific instants for the occurrence of property events and have no duration in time.
 - *Continuous Time Locators (CTLs):* define continuous periods that are delimited using a starting and ending events with a positive duration. The time periods can overlap with each other.

- **Condition to Be Fulfilled (WHAT):** refers to the condition that needs to be satisfied by the spatial locators during the specified time periods. The evaluation of the condition can take four possible values *true*, *false*, *undecided* when the condition is still not fulfilled and the time period is not yet finished, and *undefined* if tested outside the time period.
- **Probabilistic Constraint (HOW-WELL):** refers to a probabilistic constraint on the satisfaction of the property. (e.g. the probability of crossing the threshold must be less than $1e^{-4}$)

3.4.3 Goals and Key Performance Indicators (KPIs)

A goal is a desired property that a stakeholder wants to be fulfilled. During early conceptual design phases, stakeholders cannot usually define precise numerical targets for their goals as the target values can only be precisely evaluated following an in-depth analysis where the available technologies are identified and agreements between involved stakeholders are made. An example to illustrate this point can be a political entity that aims to renovate a neighbourhood in order to make it more ecologically responsible. A precise target for the eco-friendly objective is very difficult if not impossible to be fixed at conceptual phases given that it depends on a large number of unknowns that are clarified and refined during the design process and emerge as a result of negotiations between stakeholders.

Goals are fundamentally different from requirements. The achievement of a requirement depends on a “hard” constraint with targets and boundaries that should never be crossed and that is evaluated using deterministic and sometimes probabilistic approaches. In contrast, the achievement of a goal depends on a “soft” constraint that only the stakeholder can decide whether or not it suits him. This does not mean that targets cannot be allocated for stakeholders’ goals, it only means that even if this target is not reached 100%, it can still be considered as satisfied by the concerned stakeholder. For instance, in the last example of the political entity that aims to renovate a neighbourhood to be more ecologically friendly, considering a target for the goal “*The total energy provided to neighborhood consumers should have at least 30% of renewable energy*”, if the designed system reaches a value of 27% of renewables, the objective is most likely to be considered as satisfied by the stakeholder.

The Key Performance Indicators (KPIs) are considered as “metrics” that are defined by stakeholders in order to evaluate their goals and assist them to decide whether or not they are satisfied. In CReMA methodology, the term “KPIs” will be frequently used to refer to the formal characterization of stakeholders’ goals.

In the various stages of the engineering cycle of a ME-CPS, KPIs are evaluated using different methods. In early design phases, some specific tools can be used to assess systems

behavior using simple flow models and get a rough picture of what can be accomplished using the available resources. Among the tools enabling to perform these analyses we can find AnyLogic [7] which is a multi-method simulation environment. As the system design progresses and a clearer idea is drawn on ME-CPS elements, a sharper picture emerges on the real system behavior and consequently on the stakeholders' key indicators. Domain-specific tools can be used in this context to make an in-depth analysis of systems behavior. These tools will be further discussed in section 3.7.6.

In CReMA methodology, for the sake of automatic verification, we will consider KPIs expressed as thresholds to be satisfied such as "*having more than 25% of renewable energy*". Therefore, every stakeholder KPIs will be characterized using :

- The scenario under which the test is done.
- The target of the KPI in SI unit: val_{target} .
- The final state of the property (satisfied, violated or untested).
- The final value of the KPI in SI unit : val_{KPI} .
- The final margin in regards to the target as a percentage (%), defined by $ratio = \frac{val_{KPI} - val_{target}}{val_{target}}$

3.4.4 Contract (assumption and guarantee)

3.4.4.1 Reminder of related works

The notion of contract in CReMA methodology was mainly inspired by the *design by contracts* introduced by B.Meyer in [108] and the *contract based design* introduced by Inria in [53][54]. As a reminder of section 2.7.2.8 in the state of the art chapter, B.Meyer defines a contract for a given module as a couple (precondition, post-condition). The precondition states the prerequisites to be satisfied by the inputs of the module. The post-condition specifies the conditions to be satisfied by the outputs. Inria on the other hand has introduced a formal mathematical theory that characterizes contracts using the assume/guarantee paradigm in correspondence to pre-conditions/post-conditions respectively. This work proposes a unified treatment for Contract-Based Design (CBD) with a methodology having a rigorous algebra for contract abstraction, refinement, decomposition, conjunction, and verification.

T. Nguyen has discussed in [113] how the notion of contract is implemented in the formal requirement modeling language FORM-L. Similarly to Inria's work, the contract is composed of assumptions and guarantees. However, the particularity of the FORM-L contract is that it is established between two parties and each one of them has its own assumptions and guarantees as shown in Figure 3-4. The guarantees of one party are considered to be the

assumptions of the other one and vice versa. Moreover, T. Nguyen uses contracts not only to establish the requirements and assumptions of each party but also as a means of structuring the interaction between the different parties. The contract is used as the unique interface gathering all the physical, cyber, and legal interactions between two parties.

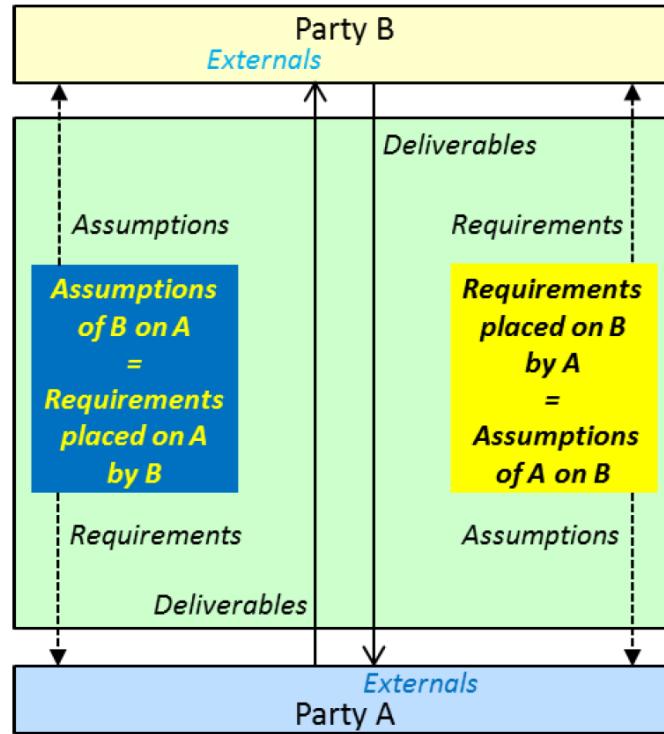


Figure 3-4: FORM-L contract (from [113])

3.4.4.2 Contracts in CReMA methodology

For the sake of coordinating stakeholders, the concept of contract in CReMA methodology will be closer to the classical contract notion that captures the obligations of its stakeholders. The definition of contract that will be considered in this framework is thus closer to the one of T.Nguyen [113], but limited to the assumptions and guarantees of each stakeholder.

In CReMA methodology, we also consider that contracts are limited in time, and that the period of time during which stakeholders must comply with their engagements must be specified in the contract. Therefore, a contract is defined with a set of assumptions A , guarantees G and a period of time P : $\mathcal{C} = (A, G)_P$.

Figure 3-5 shows an example of a contract composition in CReMA methodology involving three stakeholders, each one having a set of assumptions A_i and a set of guarantees G_i . For each stakeholder, the assumptions represent the properties that he considers to be satisfied by

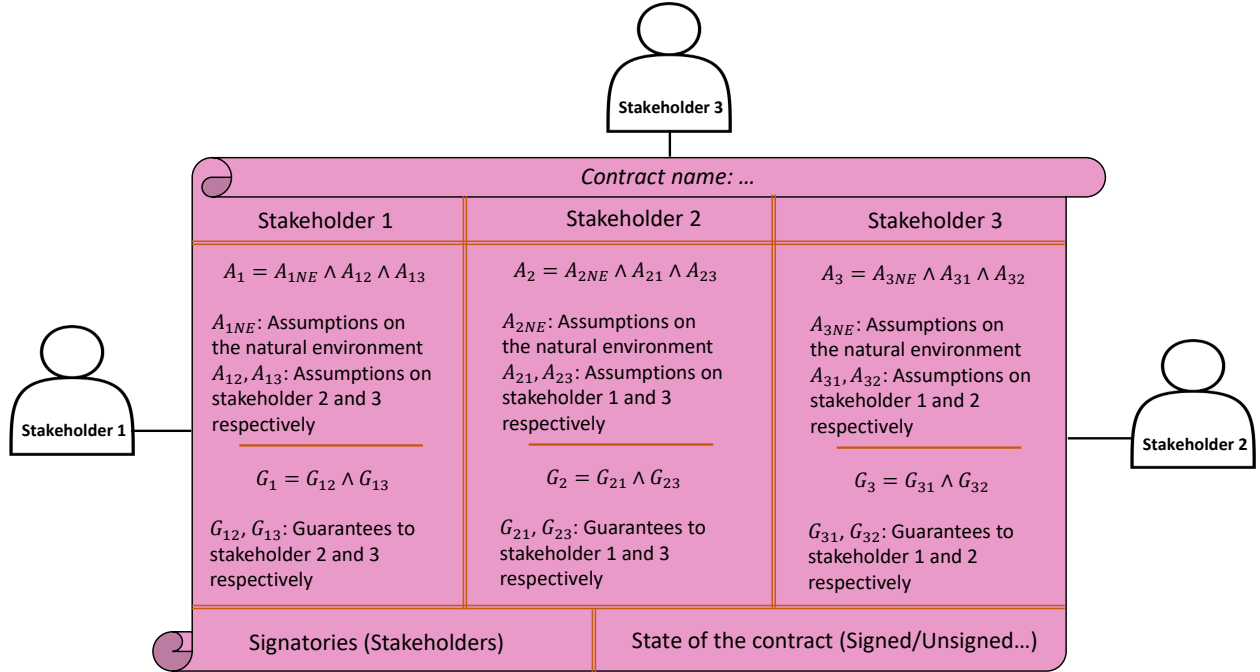


Figure 3-5: Example of contract composition including three stakeholders

definition. These assumptions are either satisfied by the natural environment or guaranteed by the other stakeholders. In the assumptions sections, each stakeholder “ i ” defines the hypothesis that were made on his natural environment “ NE ” : A_{iNE} , which are design hypotheses on which should agree the other stakeholders. For instance, assumptions can be made on the average outdoor temperature range for a given month, outside of which the stakeholder is no longer required to satisfy his dues. In the same section, a stakeholder “ i ” also defines the assumptions made on what is guaranteed by the other stakeholders “ j ” : A_{ij} . For instance, an electricity supplier making assumptions on the behavior of consumers during a yearly period represented as a maximum energy consumption curve. These assumptions need to be consistent with the guarantees of the other stakeholders: G_{ji} . The guarantee G_{ji} refers to the constraints that should be satisfied by the a stakeholder “ j ” to the benefit of stakeholder “ i ”. The guarantee of consumers in the early example would be to ensure that the maximum energy consumption is never exceeded.

The *natural environment* referred to as “ NE ” should not be confused with the notion of *environment* “ E ”:

- The *natural environment* stands for the living and non-living things in earth which are exclusively governed by natural phenomena and not by civilized human actions (or at least without a massive impact of human) such as the weather, natural resources, ecological units, etc.[23]. In this thesis, we will enlarge the definition of natural envi-

ronment to include all the elements surrounding a stakeholder that cannot be placed under the responsibility of any of the involved stakeholders and cannot be controlled by him. To cite examples we mention the electricity market, the financial market that can have considerable impact on ME-CPS design, but is out of the scope of the involved stakeholders (similarly to the classical natural environment).

- When it comes to the concept of a stakeholder *environment* “ E ”, a wider scope is considered. It encompasses all the surrounding elements of a stakeholder including the natural environment and also the other stakeholders and their systems. This definition can be formalized as: $A_i = A_{iNE} \wedge_i A_{ij} = A_{iE}$. This notion was set up in the context of the formal contract algebra that was developed in this thesis and introduced in Appendix C.

It is reminded that both assumptions and guarantees are expressed in the same requirement language. However, assumptions are not considered as requirements in the design process of a stakeholder. They are considered as system boundary conditions that are necessary for the design process. The guarantees on the other hand are considered as system requirements that should be satisfied by the parties participating in a given contract. As explained earlier, each party in the contract assumes that the guarantees from the other parties in the same contract are satisfied, possibly with some probability of failure.

It is also emphasized that the parties involved in a contract specify its properties, however, (generally speaking) they do not specify the means through which those properties will be ensured.

3.4.4.3 Comparison between CReMA contract and Inria contract [54]

If we make a correspondence between the contract in Figure 3-5 and the definition given by Inria to a contract [54], one can identify three assume/guarantee (Inria’s) contracts in the figure, one for each stakeholder.

Partitioning a contract between two or more stakeholders into sub-contracts will be used in CReMA methodology in order to enable stakeholders to carry out their analysis and design in a more autonomous manner. In other words, each stakeholder can proceed with the design of the system under his scope using self-contained sub-contracts, under the assumption that the other stakeholders will commit to their sub-contracts.

3.4.4.4 CReMA contract attributes

Contract lifecycle As contracts are always limited in time, negotiating new contracts to replace the old ones eventually leads to consequent modifications on the ME-CPS of interest.

Faster pace of change can be obtained by amendments to existing contracts. To this end, we have defined the notion of contract “lifecycle” which includes the different states that a contract undergoes over time. A contract lifecycle includes four states :

- In edition: the contract is being established by stakeholders.
- Unsigned (in negotiation): the contract is edited and challenged with behavioral models in order to make verification using modeling and simulation.
- Signed: all stakeholders have agreed the terms of the contract and the contract satisfies all its stakeholders.
- In amendment: the terms of the contract are being modified and re-negotiated.

Along a ME-CPS lifetime, contracts will undergo multiple amendments, the challenge for each stakeholder is then to ensure that he will always be able to fulfill his obligations under the assumption that the other stakeholders satisfy their own commitments, and prepare mitigation actions in case of breach of contract by one (or maybe several) of them.

3.4.5 The implementation

The term implementation represents the global achievement made by a specific stakeholder in order to fulfill his obligations. The architectural description and behavioral modeling that are made throughout the design are intended to set up representations of this implementation. On another note, the verification models aims to check that the implementations provided by stakeholders are faithful to the obligations they need to satisfy and for which the implementations was developed. This term is mainly used in the formal logic of CReMA methodology presented in Appendix C.

3.4.6 Architecture and architecture description

Referring to the ISO 42010 standard[87], the architecture of a system is defined as “*fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*”. It should not be confused with the architecture description that is defined as a “*work product used to express an architecture*”. It describes the system from different views in order to answer the concerns of its stakeholders and designers. These views are governed by viewpoints (as specified in the standard) that establish the rules, disciplinary conventions, languages, interpretation, etc in order to characterize the interactions between the elements of a complex system.

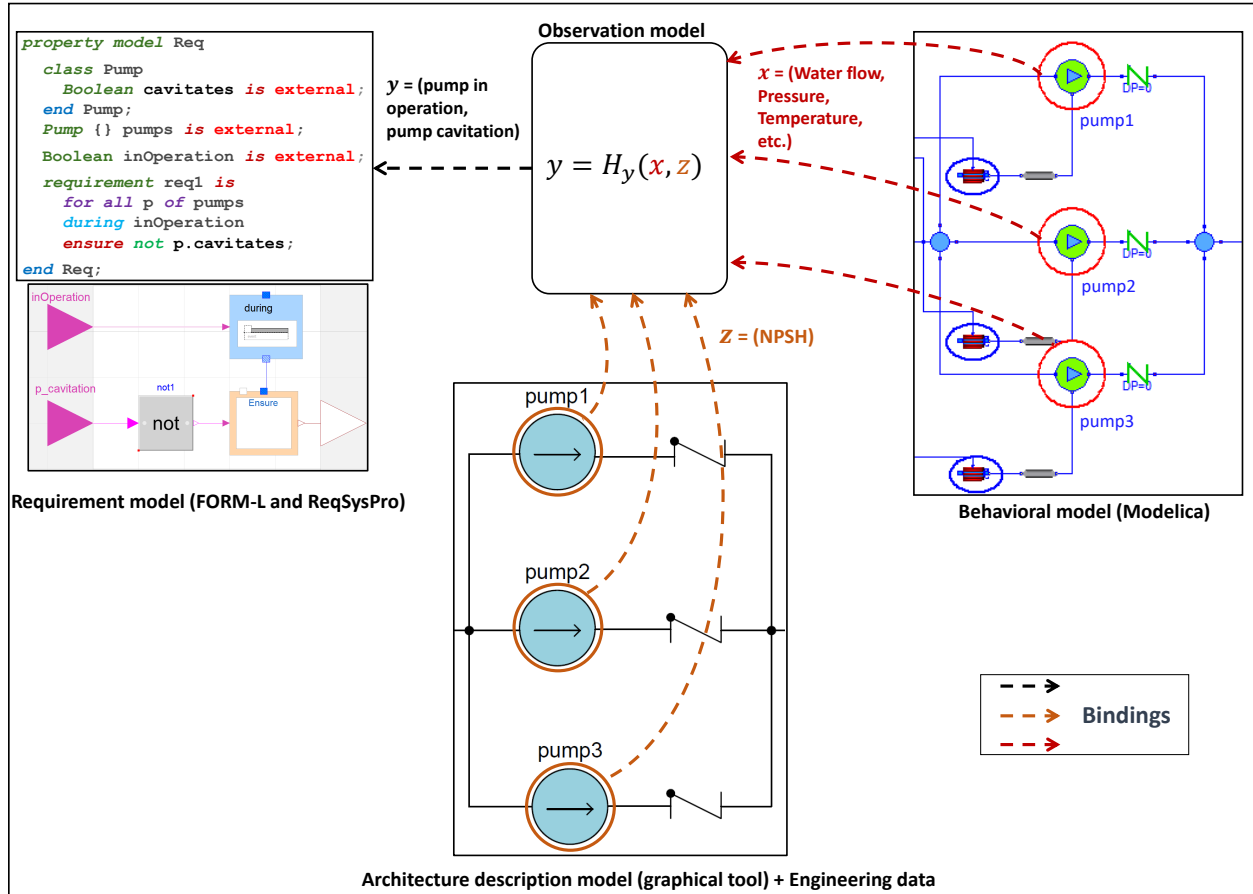


Figure 3-6: Positioning the concepts of observation model and bindings in the construction of the verification model [63]

3.4.7 Observation model

The verification of requirement models usually requires external variables that carry pieces of information on the system that should fulfill the constraints. These variables are provided by external models that may represent (i) different views on the system that are developed at various stages of the design process such as the architectural description view (with static data) and behavioral modeling view (with dynamic physical states), (ii) different disciplines such as physics (electricity, heat, etc), safety, financial, etc. (iii) different subsystems or components, and (iv) different design alternatives [75].

Therefore, the goal of the observation model is to observe one or multiple models and extract data that will be transformed in order to assess the system functional states. The functional states are subsequently used (through the observation operators) as input properties of the requirement models. The observation operators act as non intrusive sensors on behavioral models (in the sense that there is no need to modify these models) and do not

affect their simulation results. This means that designers can use their existing behavioral models developed using domain specific tools without having the obligation to develop new “consenting” models.

Figure 3-6 displays an example of verification model for the requirement: “*for each p in pumps such that p is in operation, ensure that p does not cavitate*”, which was introduced in [62], [63]. In this requirement, two external dynamic variables are mentioned: “pump cavitation” and “pump in operation” which are functional states of the pump. Whereas, on the one hand, the behavioral model represents the system using physical equations that compute physical state variables such as the water flow, temperature or pressure, on the other hand, the architectural description of the system contains the characteristics of its components such as the NPSH (Net Positive Suction Head) of the pump, which may be stored in a separate database. If we consider y as a vector of external variables of the requirement model which are functional states, x as a vector of physical states computed by the behavioral model, and z as a vector of characteristics captured from the architecture model, the observation model transforms the physical states and system characteristics into functional states using the observation function H_y that can be described as follows:

$$y = H_y(x, z)$$

If we consider the external variable y_1 =“pump in operation”, observing the physical states of the system such as the supply current, the angular velocity or the mechanical torque are sufficient to determine whether the pump is in operation or not: $y_1 = H_{y_1}(x)$. However, if we consider the external variable y_2 =“pump cavitation”, both the physical states such as the pressure and the flow, and the architecture characteristic NPSH are required for the cavitation assessment : $y_2 = H_{y_2}(x, z)$.

3.4.8 Binding

The goal of bindings is to automate the process of connecting multiple formal models developed by different teams for the sake of automatically generating verification models [63]. The central idea of bindings is the establishment of the links that bound the static data of the architectural model, the dynamic variables of the behavioral model, the interfaces of the observation model, and the requirement model. Figure 3-6 gives an example of a verification model including bindings (in dashed arrows) that connect its elements. The requirement example used in that figure, notably “*for all p in pumps such that p is in operation, ensure that p does not cavitate*” [62] represents a typical illustration for which the use of automatic bindings is very accurate. The quantifier “for all” requires using operators that enable connecting a requirement to an external set of components instead of a manual instantiation

that can be time-consuming if conducted manually. In this example, we only consider three pumps, which can be easily managed by hand, however, this number can be much higher in large complex systems. Refer to [63] for more details on the bindings operators.

3.4.9 Test scenarios

The test scenarios represent a modeling component embedding the assumptions made on the environment of the system in the form of an envelope gathering all the trajectories of possible behaviors, with the corresponding parameters characterising them. Using a script, multiple simulations can be launched randomly or in a more controlled manner by considering different configurations from the specified envelope. The test scenarios can be connected to any object of the verification model including the stakeholders description models, contracts and observation models in order to set, for every simulation, the parameters corresponding to the scenario to be tested.

3.4.10 Refinement

The design of complex systems is a progressive and iterative process that goes through several engineering steps. Throughout this process, design patterns such as stakeholders' needs, requirements, architecture, and behavioral modeling are continuously refined through formalization, decomposition, substitution, and allocation. At every refinement step, a clearer picture is shaped regarding the system for which the design is powered by the induced specifications, constraints, and operational data all along the refinement process.

In order to improve the likelihood of making a successful design of a complex system, the refinement needs to be supported by methods and tools that ensure a continuous consistency all along the system design, from early design levels where stakeholders have undetailed visions on the future system and its environment, to more advanced design levels with more detailed views.

In this context, the refinement in CReMA methodology is supported by formal means that allow checking the consistency between different refinement levels (refer to Appendix C for the formal definition). Furthermore, a justification framework is used in order to capture the reasoning chain that was followed within the refinement process. It allows justifying the correctness of each refinement step by establishing a set of inference rules that must be satisfied.

3.5 CReMA methodology - Global overview

3.5.1 Overview on CReMA methodology levels

In order to assist stakeholders elaborate the contracts between them, we propose a framework based on multiple modeling levels. Figure 3-7 shows all the methodology levels at the disposal of stakeholders in order to progressively elaborate the contracts and allow them to develop verification models. In this figure, a distinction is drawn between concepts that are used for the description of stakeholders and the concepts that are used to describe the relationships among them at different modeling levels. We also make a clear distinction between the levels that depict visions that are centered on stakeholders, and the levels that rather provide a system view. These two views are strongly inter-related, and the design process continuously iterates between them.

The common thread of our methodology is requirement modeling and refinement. In the “Stakeholder view”, a first sketch of stakeholders’ needs and contracts between them is established. In the “System view”, stakeholders’ needs are progressively refined through the different design steps. Throughout this refinement process which goes along with the design, requirements are subject to various kinds of transformation. They can either be formalized, substituted, or decomposed when passing from one step to another. Moreover, as the maturity of the systems increases, the design choices made by a stakeholder induce new technical and physical requirements (e.g. choosing to use a pump induces additional requirements for the proper operation of the pump). These requirements can be guaranteed by the stakeholder in charge of designing the system and in this case the requirement becomes part of the stakeholder requirement model. Otherwise, when the requirement is to be guaranteed by a system outside the scope of the stakeholder, it is embedded in the so-called “System contract” that is established with an external stakeholder. The “Design justification level” plays a major role to ensure that the refined patterns remain faithful to the upstream levels of abstraction.

From the behavioral modeling perspective, every stakeholder has the freedom to choose the adapted paradigms to perform his analysis and challenge his requirement model. The coupling of stakeholders’ behavioral models can be made using connectors (that can also be called bindings).

The “Stakeholder view” mainly focuses on capturing what we refer to as “social aspects” of stakeholders. These aspects concern the stakeholders’ intrinsic interests, their requirements regarding each other, as well as the interactions between them. The three modeling levels related to the stakeholder view are:

- The intentional level that describes each stakeholders’ high-level goals as well as the dependencies between different stakeholders.

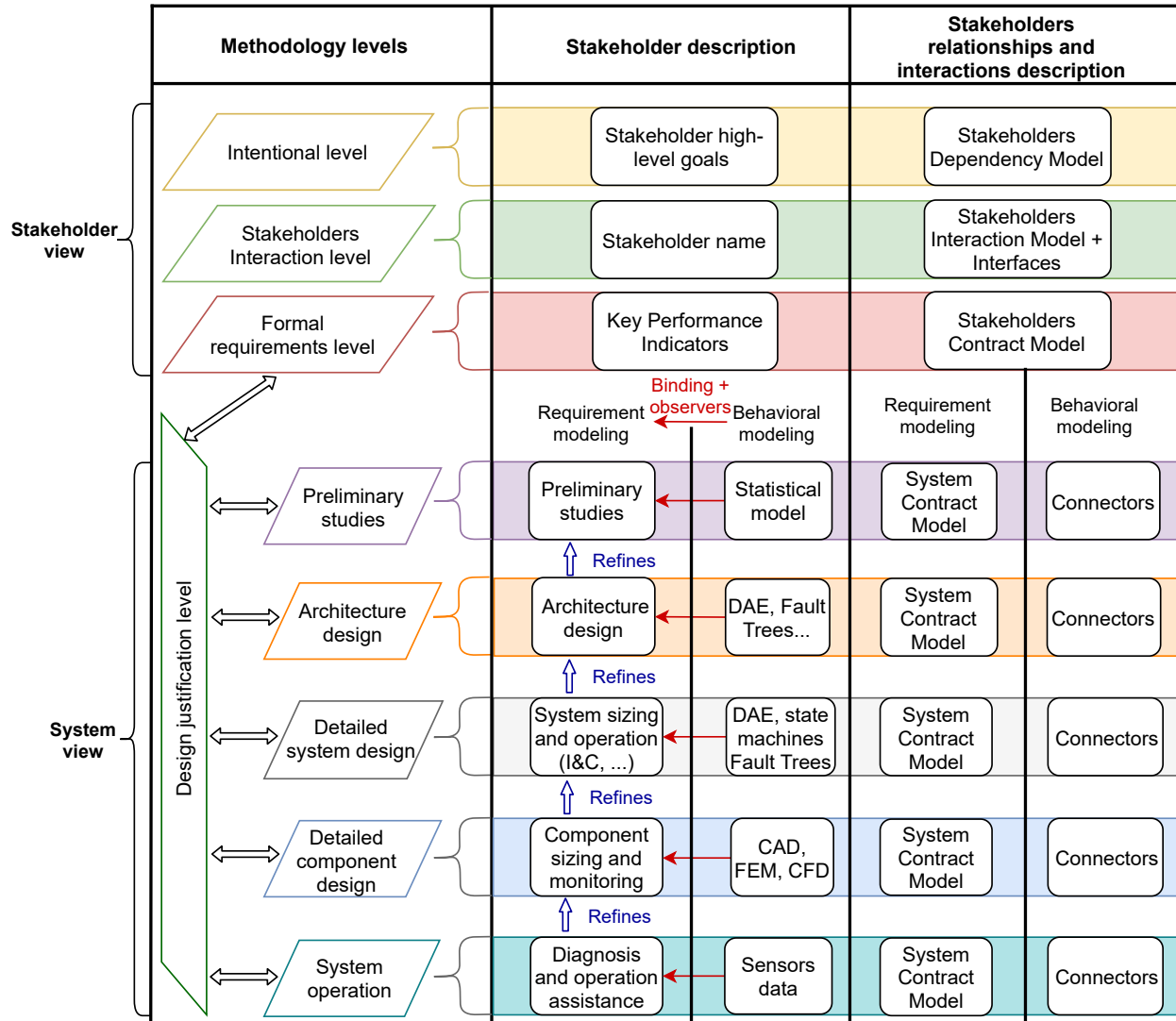


Figure 3-7: Methodology levels

- The stakeholder interaction level that describes the interactions between stakeholders such as financial exchange or data exchange.
- The formal requirements level that formalizes the high-level intents of stakeholders and requirements related to the dependencies between them.

The “System view” is composed of several levels that aim at describing a digital twin of the systems under the responsibility of stakeholders. The digital twin is considered as including dynamic behavioral models of a system and requirement models. These two elements are necessary to ensure that a system meets the goals for which it was developed at every design step. This is even more valuable when dealing with safety-critical systems for which proving

the correct operation of the system is as important as the design itself, as the system cannot be approved for operation if such proof cannot be made.

During different phases of the system lifecycle, the digital twin will undergo several modifications. The behavioral modeling will continuously evolve to perform a more detailed analysis, and the requirement models will be continuously refined as a result of design choices that are made. Figure 3-7 shows five modeling levels that are related to the system view. They are not exhaustive and can widely vary according to disciplines and design practices:

- Preliminary studies: The idea here is to sketch out the landscape by making a rough analysis of the system to be developed. The main goal for stakeholders here is to analyze the feasibility and viability of the project regarding their interests. Knowing that at this level only few information is known about the system, the requirements are challenged using statistical models that generate random test sequences constrained to the assumptions made on the environment of the system.
- Architecture design: This is the first step of the system design. Referring to SE standard ISO15288 [42], the purpose of the architecture definition process is to “*generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views*”. At this design level, and for the sake of identifying the system architecture that answers the stakeholders’ needs, modeling paradigms such as Differential-Algebraic Equations (DAE), Fault Trees (FT), etc. can be used by stakeholders to challenge their requirement models.
- Detailed system design: The main goal of this level is to provide design characteristics of systems and their components while being consistent with the architectural description level [42]. It also aims to identify alternative solutions for system components. Behavioral models at this level incorporate operational aspects of the system. Disciplines such as Instrumentation and Control (I&C) are integrated at this level to study the dynamic aspects of the system.
- Detailed component design: At this level of design, the elementary components of the system are chosen and the interactions between them are defined. In general, after the detailed system design, the designer uses components off-the-shelf, unless specific characteristics are required. This situation is rare and leads to costly development. The components chosen at this level induce new specifications regarding their operational constraints that are integrated into the requirement models. Detailed behavioral models using for instance Computer-Aided Design (CAD), Finite Element Method (FEM), and Computational Fluid Dynamics (CFD) approaches are used in order to perform

V&V on these components. Verification must be made in order to ensure that these additional requirements are compatible with the existing ones.

- System operation: During the system operation phase, an on-line diagnosis can be made. Data coming from the sensors feed the verification models to check that operational requirements are being complied with.

At every design step phase, two concepts are used in order to build verification models:

- Observers allow transforming system physical states (such as mass flow-rates, voltages, etc.) into functional ones (such as a component being turned on or off) which are compatible with the inputs of the requirement models.
- Bindings that connect models of different nature and allow transferring the data computed by the observers in order to perform verification tests using simulation.

The particularity of our work lies in putting the stakeholder’s view at the center of the design process, and therefore, we will synthesize the system view through the introduction of two generic levels : (i) the “Architecture description level” that has the particularity of describing the static engineering knowledge concerning the design solutions, and (ii) the “Behavioral level” that represents the dynamic behavior of these solutions. Both levels are necessary for the V&V of systems. These levels will capture systems architecture and behavioral models at every design step. More broadly speaking, throughout these two levels, we will have a snapshot of the models used by stakeholders, at any specific time.

The methodology levels can thus be reduced to the ones shown in 3-8. This figure emphasizes the underlying techniques of each level, which will be detailed in the following sections.

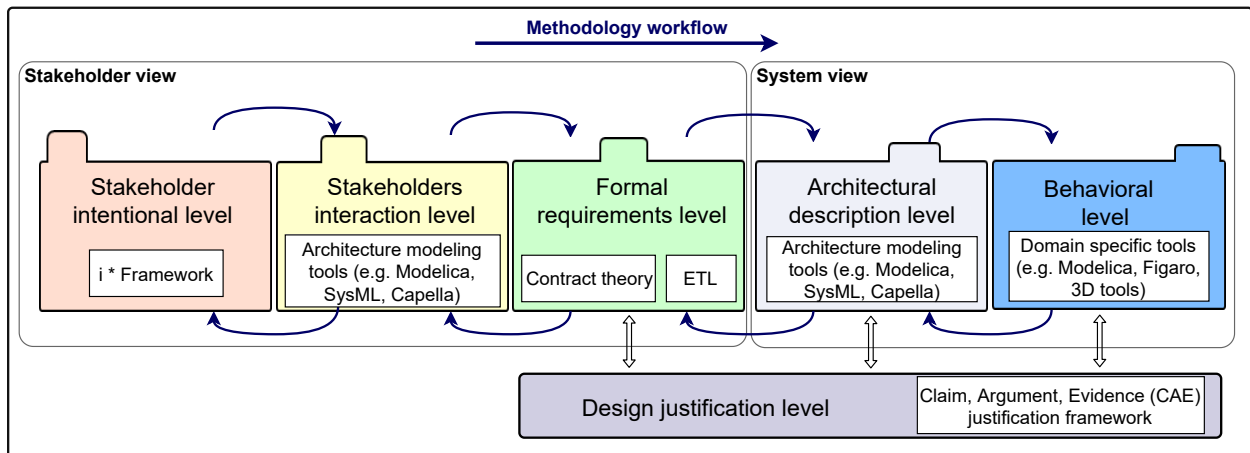


Figure 3-8: CRema methodology levels with underlying techniques

3.5.2 Methodology phases

CReMA methodology iterates over two main phases until a common agreement is found between stakeholders and contracts are signed.

- Contract elaboration phase. Instead of starting from the technical standpoint of the classical system definition, which mainly focuses on what a system shall do and how well it should perform, our starting point is the high-level needs of stakeholders that represent their primary motivations for committing to the overall system. These needs are decomposed through multiple refinement steps in which stakeholders identify the alternative solutions that achieve their goals while keeping track of the rationale behind their decisions. Throughout the decomposition of the high-level intents, relationships and dependencies between stakeholders emerge. Contracts provide a formal framework to contain these relationships and act as interfaces between stakeholders. The contracts are elaborated through several negotiation steps.
- Verification and validation phase. In order to agree on a contract and sign it, a stakeholder needs to make sure that the obligations of other contract stakeholders enable him to meet his high-level needs. Moreover, the stakeholder needs to precisely understand his own obligations and verify his ability to commit to them. Indeed, the contract is a legal commitment to all its signatories.

We emphasize that CReMA methodology is not based on a sequential process that constrains stakeholders to complete one step or level before moving on to the next one. The last approach is not in harmony with our vision of ME-CPS that are constantly evolving and are always subjected to the legacy energy system which means that the design never starts from scratch. As a result, CReMA considers that the different levels are certainly linked to each other to ensure consistency, but the different views can be used independently over time. The stakeholders thus have the freedom to construct the levels of the methodology without having to comply with any chronology that may constrain them throughout their design.

3.6 Methodology metamodel

3.6.1 Definition of a metamodel

A meta-model or information model as called in software engineering is a model of model that is used to provide an insight on all the concepts that are used in the construction of a new modeling paradigm, language, or design methodology. A metamodel is thus an abstraction of a model that is itself an abstraction of real item [91]. The goal of a metamodel is mainly

to structure the relationships between concepts and specify the nature of these relationships, rules, and operations that govern the interactions between them.

A metamodel is a fundamental pillar for conceiving a pertinent SE approach. As a matter of fact, SE as a standard gives far more weight to the definition of engineering concepts (e.g. stakeholder, requirements, architectures, etc.) and the relationships that bind them (e.g. refinement, traceability, allocation, etc.) than to the diagrams and facets that are used to structure a system knowledge and essentially to use them for communication purpose [133]. From one metamodel, multiple facets can be generated depending on what pieces of information the facet aims to communicate and which analysis it intends to make. The metamodel ensures a global consistency between the different facets and guarantees that a modification inside one view is propagated to other ones [133].

In [133], Scott et al. propose a mapping between a SE metamodel (called information model in the paper) and the different representations and facets used in the core of the MBSE approach. Being inspired by this approach, CReMA methodology will be structured in the same way. We are referring to the establishment of a high-level metamodel gathering all the concepts used by the methodology and later decomposing it into smaller metamodels that can be more easily managed by one view or facet. This paradigm guarantees to have the system view or the holistic view of a complex system that is highly sought after by most of MBSE methodologies. At the same time, the decomposition of the global metamodel into smaller ones allows breaking down the complexity of the system view into more manageable concepts that can be handled using understandable diagrams.

3.6.2 CReMA methodology metamodel

In order to bring together the large number of concepts that make up the methodology and define the links between them, a metamodel was developed for CReMA as illustrated in Figure 3-9.

This metamodel plays a major role to allocate methodology concepts to the different levels of CReMA, and ensures that all these concepts are implemented in a consistent way, with no redundancy and no oversight of any methodology key item.

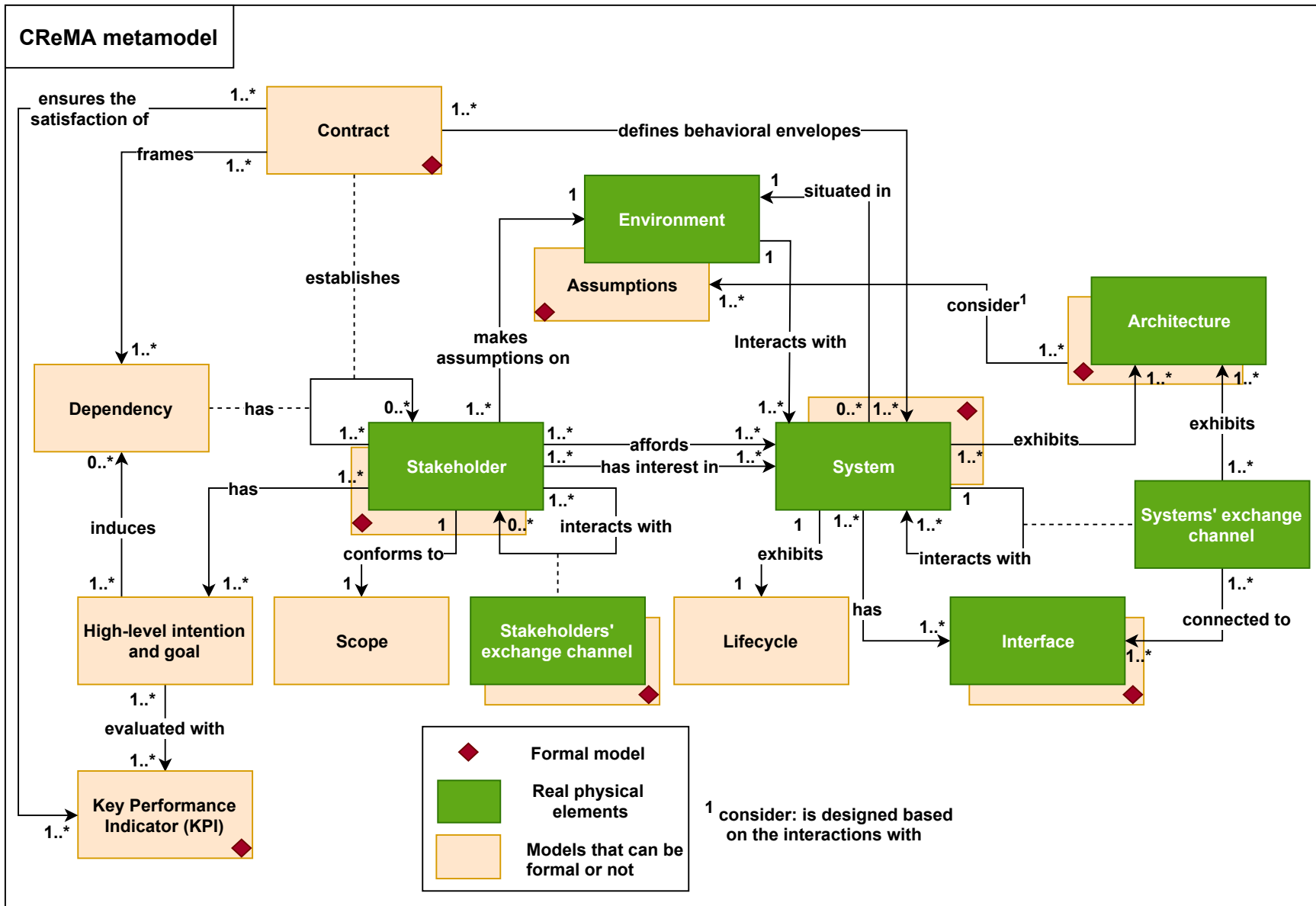


Figure 3-9: Global metamodel of CReMA methodology

A color-coding is used to distinguish between the real objects and the models that are used for their description. The green elements refer to “real” items such as the stakeholder, the system, exchange channels between stakeholders or systems, the interfaces, or also the system architecture (not to be confused with the architecture description). The orange blocks refer to abstractions of the “real” items in green blocks which are captured in models. These models can either be non-formal, meaning that they are based on natural language representations (may be structured in a diagram), or they can be formal meaning that they are based on rigorous paradigms and able to be simulated. Formal elements are marked in the metamodel by a burgundy diamond.

In the next paragraph, the metamodel is explained and the methodology concepts are written using *italic* font.

CReMA methodology metamodel revolves around two central elements, the *stakeholder* and the *system*. In this metamodel, each *stakeholder* is considered as having *high-level intentions and values* that induce *dependencies* regarding other parties. The *high-level intentions and values* are first described using natural language and subsequently characterized using formal *Key Performance Indicators KPIs* that will be used by stakeholders to precisely evaluate their needs and make decisions. The *dependencies* will also be characterized using formal requirements that will be introduced into *Contracts*. Each *stakeholder* has a *scope* that is limited in time and can evolve during the system *lifecycle* as a result of negotiations or modifications in *stakeholder* roles. A *stakeholder* interacts with one or more *stakeholders* through *stakeholders’ exchange channels*. A *stakeholder* makes *assumptions* on its *environment* which are considered to be the boundary conditions based on which the system design is made. The *system* is situated in the *environment* and interacts with it and with other *systems* through *systems’ exchange channels* while passing through the *interfaces*. The *system* and its *interactions* exhibit an *architecture* that is captured using a formal *architecture description* (orange frame). The later is conceived based on the *assumptions* taken on the *environment*. The *architecture description* describes the alternative *architectures* of the *systems* that will satisfy the *contracts* and the *KPIs* of *stakeholders*. The dynamic behavior of *systems* is described using *behavioral models*

In section 3.7 which introduces CReMA methodology level, this metamodel will be considered as the guiding thread. Indeed, the metamodel shown in Figure 3-9 will be decomposed into smaller metamodels that will be assigned to each methodology level. Each sub-model will encompass the concepts that will be used by each specific modeling level.

3.7 Methodology levels

This chapter aims to define the different modeling levels that have been stacked in order to develop CReMA methodology. An overview of these levels is given in section 3.5 and a layer by layer description is given in the sections that follow. Each methodology level was defined using an identity card (inspired from [102]) that includes the following elements :

- **Level name** : the name given to the methodology level.
- **Related view (stakeholder or system)**: refers to one of the two views for which the level contributes using its specific patterns.
- **Level objectives**: shows the reasons for which the level was considered and implemented in CReMA methodology, and how this implementation answers the objectives.
- **Level metamodel (or information model)**: gathers the concepts used in the level and the relationships between them.
- **Positioning in the engineering cycle**: Places the level relatively to the other methodology levels.
- **Diagrams and models supporting the level**: refers to the views that were developed or taken from the state of the art and used to represent a system facet in this level.
- **Syntax of the level**: defines the graphical elements representing the concepts of the methodology level and the relationships between them.
- **Tools and libraries supporting the level**: refers to the tools and libraries that can be used in order to support executable M&S of one or several parts of the system in a specific context.
- **Relationships and interaction with other levels**: captures the mapping of the concepts between different levels and emphasize how they are transformed when refining a level into another.
- **Contribution to the formal verification model**: underlines the concepts that will feed the executable verification model in order to distinguish the elements used for design elicitation and the ones used for verifying the compliance of the design with respect to intents and requirements.

3.7.1 Stakeholder intentional level

- **Level name** : Stakeholder intentional level
- **Related view** : Stakeholder View
- **Level objectives** :

The main objective of the stakeholder intentional level is to introduce a text-based structured framework allowing stakeholders to define their high-level goals and the dependencies between them at early conceptual phases. The high-level goals are the primary reasons for which stakeholders have a concern in the ME-CPS. Goals are the lifeline for design choices that will be made by stakeholders all along the engineering cycle of the system. Stakeholders will seek to maximize these goals and minimize their dues to others since the two aspects generally point in opposite directions. This leads us back to the second objective of the intentional level which is the identification of dependencies between stakeholders. The dependencies are identified as follows: each stakeholder defines his high-level goals, they are subsequently decomposed into more manageable sub-goals and sub-tasks. Then, the stakeholder evaluates its capacity to satisfy these sub-elements with minimal costs, and in case of incapacity, he hands over its endeavour to another stakeholder of the ME-CPS, which induces a dependency towards him. Dependencies have multiple degrees of vulnerability. A stakeholder should assess its dependency towards others and consider the possibility that it may not always be satisfied due to unforeseen circumstances. A reflection of this consideration should be found in the contract that will further be discussed in the formal requirement level of CReMA.

- **Level metamodel:**

Figure 3-10 depicts the two main concepts that are captured in the intentional level. A *stakeholder* is considered as having one or multiple *high-level intentions and goals*. The decomposition of these goals may induce zero, one or multiple *dependencies* between the stakeholder and one or more other *stakeholders*.

- **Positioning in the engineering cycle:**

Modeling stakeholders' intentional level is developed during the early design phases where stakeholders draw a sketch of their needs and their relationships with each other. However, the development of this level does not end at that point, it is indeed built through an iterative process during which stakeholders' goals become more focused and choices are made for the means-ends along with the design process. In addition,

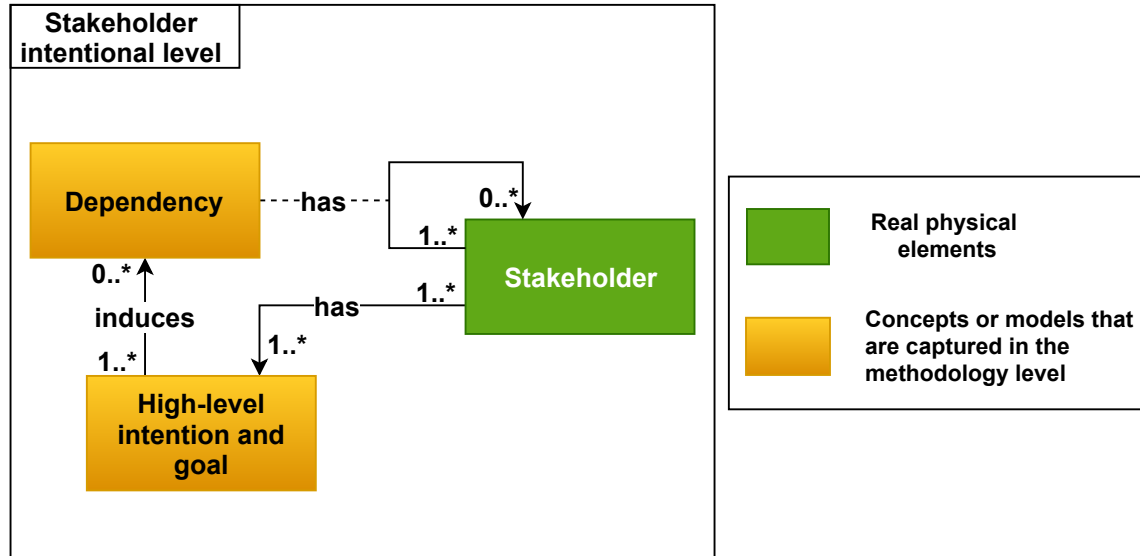


Figure 3-10: Metamodel of the stakeholder intentional level

this level needs to be revisited every time a stakeholder joins or leaves the scope of the designed ME-CPS, which leads to a new stakeholders' arrangement.

- **Diagrams and models supporting the level:**

The two diagrams that are used at the intentional level and shown in Figure 3-11 have been largely inspired from the *i** framework that was introduced in more detail in section 2.3. They are defined as the following:

- The first view called the Strategic Dependency (SD) model which deals with stakeholders as “black boxes” and captures the dependencies between them using natural language.
- The second view concerns the Strategic Rationale (SR) model which deals with stakeholders as “white boxes” and captures the rationales - as its name suggests - behind the dependencies between them. Stakeholders define their high-level goals as well as their roles in this view by using the four objects: goal, task, resource, and soft-goal. These elements are further decomposed using similar types of objects. The aim is to identify on the one side the elementary objects that are more easily achievable and manageable. On the other side, the idea is to capture the alternative means-end for achieving the elementary objects.

We emphasize that the high-level intents and the “rationale” elements justifying the dependencies of a stakeholder are developed by himself and are thereafter integrated into his own SR model. The latter will be the first brick of the stake-

holder’s description and the most abstract one in the sense that it captures his main purposes for taking part in the project. The SR constitutes the first description layer of a stakeholder system and the most abstract one. On the other hand, the dependencies are visible between each stakeholder, as these will be the elements on which the requirements put in the contracts will be based.

Even-though the SR and SD models are part of the first layer of the methodology, it does not mean that they are frozen in conceptual studies. Indeed, they are progressively refined and also modified all along with the development phases. Indeed, the high-level goals of stakeholders as well as their roles can widely vary during negotiations.

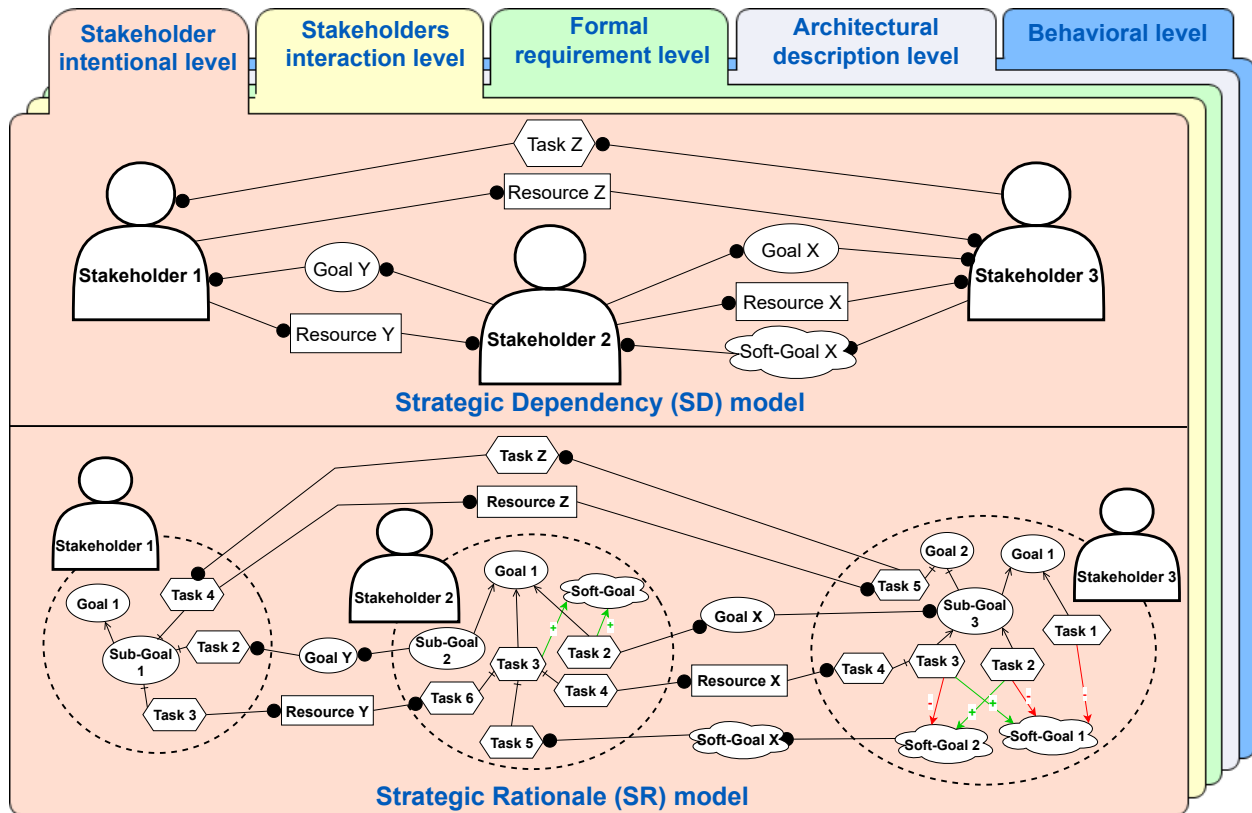


Figure 3-11: Intentional level

- **Syntax of the level:**

The graphical elements that are used to develop the SD and the SR are shown in Table 3.1.

- **Tools and libraries supporting the level:**

Compared to the *i** framework, the intentional level of our methodology is limited to the natural language description of the SD and SR. The formal characterization of


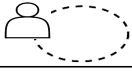


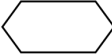

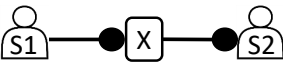
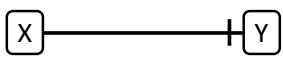
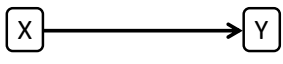



Graphical element	Element description	Example of use
	Stakeholder	Consumers, Producers, Regulators, etc.
	Actor boundary or scope	-
	Goal	Safely operate electrical equipment
	Soft-Goal	Fast processing of a request
	Task	Subscribe to electricity supply contract
	Resource	Electricity, money, data, etc.
	Dependency link between stakeholders	Stakeholder S1 depends on stakeholder S2 to ensure the dependum X
	Decomposition link between items (goals, tasks, soft-goals and resources)	The item X decomposes the item Y
	Means-end link between items	The item X is one of the alternative solutions for achieving the item Y
	Positive contribution to soft-goal	The item X positively contributes to the achievement of the soft-goal Y
	Negative contribution to soft-goal	The item X negatively contributes to the achievement of the soft-goal Y
	Unknown contribution to soft-goal	The contribution of the item X to the achievement of the soft-goal Y is not known at a specific instant.

Table 3.1: Graphical elements used in the intentional level

the viability of dependencies and vulnerability of stakeholders was not considered. It can be the subject of a thorough analysis on its own. The i* framework is supported by a tool named Organization Modeling Environment (OME) [25] that provides the graphical interfaces to develop the SD and SR models.

- **Relationships and interaction with other levels:**

This level will be the foundation for setting up the next three modeling levels of CReMA. In fact, stakeholders' goals and the constraints related to the dependencies will be respectively formalized into KPIs and formal requirements in the "Formal requirements level". Moreover, the resource dependencies identified in the intentional level will be retrieved amongst the interactions between systems that are modeled at the architectural level in section 3.7.5 as well as in the stakeholders' interaction level in section 3.7.2.

- **Contribution to the formal verification model :**

The stakeholder intentional level does not contribute to the formal verification model because it only includes text-based diagrams that are not supported with formal syntax and semantics. This level is mainly used for elicitation purposes of stakeholders' intentions and goals that will later be formalized and refined to bring out executable requirement models.

3.7.2 Stakeholder interaction level

- **Level name :** Stakeholder interaction level

- **Related view :** Stakeholder view

- **Level objectives :**

The stakeholder interaction level, as its name suggests, aims to characterize the interactions among ME-CPS stakeholders. This level has the particularity of capturing the interactions that cannot be observed nor captured in a system view as the latter mainly focuses on the exchanges between physical or cyber components while making a partial abstraction of the stakeholders' level. Examples of interactions at the stakeholder level include financial aspects, administrative aspects, etc. that are of great matter for decision making but are not part of the system views.

- **Level metamodel :**

Figure 3-12 depicts the main pattern that is captured by the stakeholder interaction level which is the formal representation of what is exchanged between *stakeholders*. It is represented using the yellow gradient colour with a burgundy diamond. In this information model, we consider that a *stakeholder* interacts with zero, one or more stakeholders through *stakeholders' exchange channels*.

- **Positioning in the engineering cycle :**

The stakeholder interaction level is developed in early design phases but is a result of an iterative process that is applied over the entire engineering lifetime of the ME-CPS where the set of the involved stakeholders substantially vary depending on the design phase.

This level is intended to be supplemented during detailed design phases by reflecting the interactions that are identified at system levels. Thus, it forms a global view on all the interactions between stakeholders as well as the interactions between the systems under the responsibility of stakeholders.

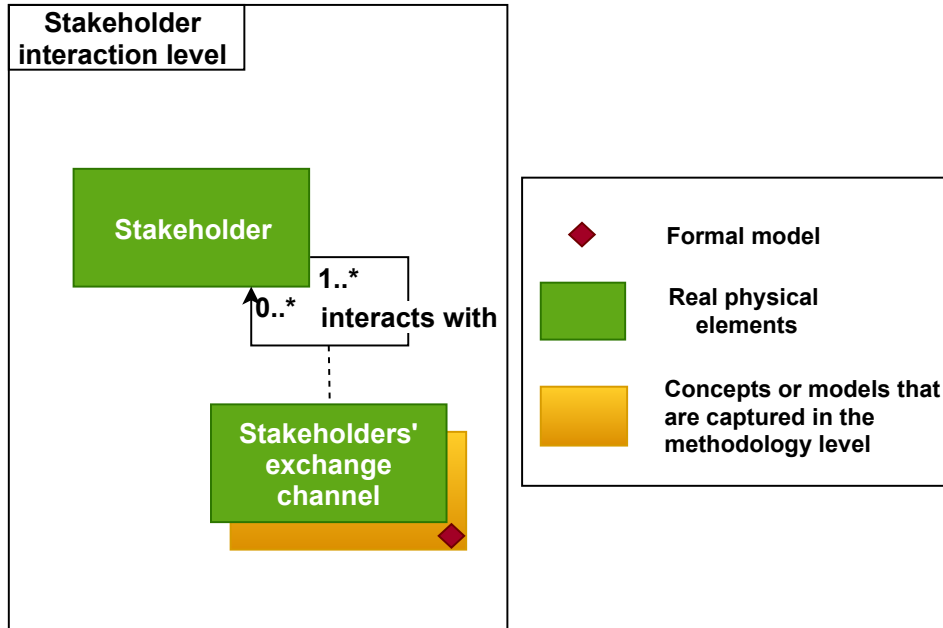


Figure 3-12: Metamodel of the stakeholder interaction level

- **Diagrams and models supporting the level :**

The stakeholder interaction level is supported by the Stakeholder Interaction Model (SIM). The SIM consists of a set of nodes referring to stakeholders and links referring to interactions that can be of various natures as shown in Figure 3-13.

The notion of interaction between two stakeholders or systems can be defined using the following elements:

- What are the exchanged elements: the resources being transferred between stakeholders. (e.g. material, energy, data, etc.)
- How is the exchange done: the protocols and rules that should be complied with during the exchange process. (e.g. Programmable Logic Controller (PLC) exchange protocols)
- Vector of exchange: the means by which the exchange is done. (e.g. physical contact, wires, pipeline, electromagnetic waves, etc.)
- Interface: boundaries between elements in interaction. (e.g. Human-Computer Interaction (HCI), sensor, etc.).

The SIM that models the interactions between stakeholders covers two of these aspects: the elements exchanged between stakeholders and the interfaces between them. On one hand, the exchanged elements are modeled by means of links that transmit flows from

one element to the other. On the other hand, the interfaces are represented using connectors such as inputs, outputs, or other acausal connectors when considering that the two sides of the link are consistent. If this is not the case, intermediate components or “adapters” can be introduced as interfaces that guarantee the consistency between the interacting elements. These adapters allow transforming the data transiting through connectors to fit with others.

The two other elements notably the exchange protocol and the vector of exchange are considered to be covered in other levels that deal with stakeholders in a “white box” paradigm. On the one hand, the exchange protocols to be guaranteed by stakeholders when interacting with each other are taken into account during behavioral modeling. The exchange protocol can also be specified using requirements in the contracts that will be established between stakeholders. On the other hand, the vector of exchange is considered to be part of the responsibility of a specific stakeholder, which will be in charge of its design and modeling. Indeed, compared to small mechatronic systems including elements that interact through simple wires or electromagnetic signals, a vector of exchange in a ME-CPS represents a complex system of its own. These vectors have dedicated stakeholders in charge of guaranteeing the proper transmission of an exchanged resource. It is the case of the electric distribution grid that transmits electricity from producers to consumers and is under the responsibility of an electric distribution operator. Similarly for the telecommunication network that has its specific operator. Thus, every exchange vector will be considered as part of the scope of a specific stakeholder.

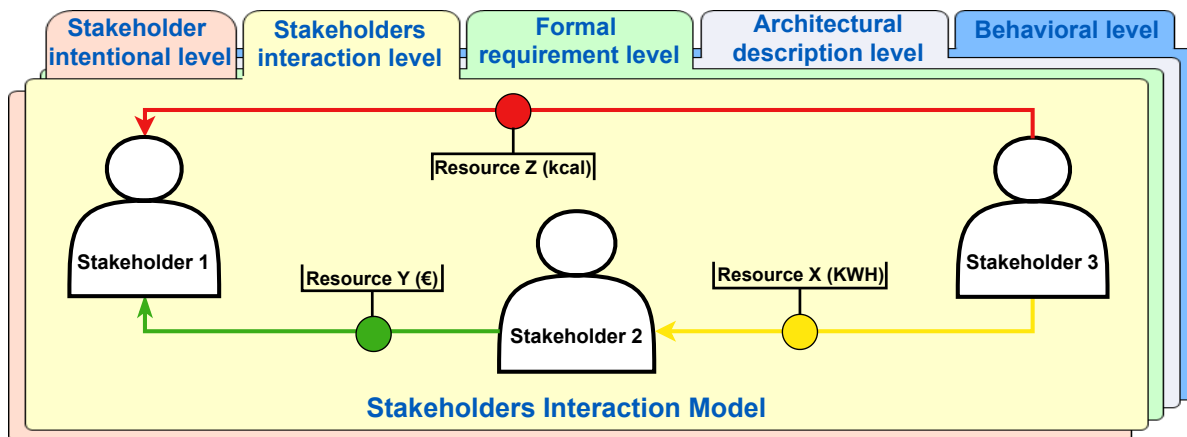


Figure 3-13: Stakeholders Interaction level

- **Syntax of the level :**

The graphical elements used for building the SIM are shown in Table 3.2.


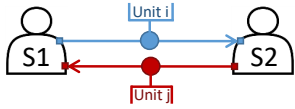
Graphical element	Element description	Example of use
	Stakeholder	Consumers, Producers, Regulators, etc.
	Interactions and interfaces	Stakeholders S1 and S2 interact by exchanging two elements having the units 'i' and 'j'. The exchanged resources pass through the stakeholders interfaces (connectors).

Table 3.2: Graphical elements used by the Stakeholder Interaction Model (SIM)

- **Tools and libraries supporting the level :**

The SIM can be developed using several platforms depending on the tools that will be used during architectural and behavioral modeling, and also their interoperability capacity. The SIM can be considered as a graphical representation reflecting all the interactions between the stakeholders. It can also be used as an executable model including stakeholders KPIs and behavioral models of the systems they are in charge of. The condition for the latter is to have consistent interfaces and a common platform with the capacity of making co-simulations when necessary.

Throughout the testing of CReMA methodology, the SIM was developed in Modelica language [81] having a graphical interface allowing to build a model by connecting blocks. It is indeed not the best-suited tool for this activity, but it allowed structuring the different models introduced by the methodology in the same platform, which was quite convenient at the prototyping phase.

- **Relationships and interaction with other levels :**

The stakeholder interaction level is closely related to the stakeholder intentional level (see section 3.7.1) as the interactions between stakeholders are the result of resource dependencies between them captured in the SD model. However, the SIM ensures a deeper characterization of these interactions.

- **Contribution to the formal verification model :**

The stakeholder interaction level sets the first exchange architecture between stakeholders. In the verification model, if multiple stakeholders are included, the SIM will guarantee the links between their description models while ensuring the consistency between the interfaces.

The SIM is an intermediate representation that takes us one step closer to an executable verification model. It is particularly helpful for defining the interfaces as well as the nature of the flows that will be exchanged between stakeholders' descriptions models in order to make the overall description executable. The stakeholder description models can be introduced here as black boxes with adapted interfaces.

3.7.3 Formal requirement level

- **Level name** : Formal requirement level
- **Related view** : Stakeholder view
- **Level objectives** :

The goal of the formal requirement level is to establish consistent and formal contracts between stakeholders in order to frame the interactions and dependencies between them. This level also aims to formalize stakeholders' needs in order to give them rigorous means to evaluate the achievement of their goals and thus make well-informed decisions. In other words, this level allows formalizing and refining the aspects that are captured in the stakeholder intentional level. Stakeholders' goals and roles are translated into formal properties that become the stakeholders' KPIs. Meanwhile, the dependencies between stakeholders are translated into formal requirements that will be integrated into the contracts between them.

The formal requirement level plays a major role in the coordination between stakeholders. It provides each one of them with means to check the consistency between his goals and the contracts in negotiation with others, starting from early design phases. It also provides means to check the consistency of a contract itself in order to highlight any contradictions or poor specifications amongst the requirements of its parties.

- **Level metamodel** :

Figure 3-14 depicts the two main concepts that are dealt with in the formal requirement level which are the *contract* and the *Key Performance Indicator (KPI)*. On the one side, the notion of *contract* is used in order to frame the *dependencies* between *stakeholders* by formalizing the properties and constraints related to the *dependencies* captured in the intentional level. On the other side, the concept of *KPI* is used in order to formally evaluate the *high level intention and goal* of a *stakeholder*. A *contract* is tightly linked to stakeholders *KPIs* as it has the role of ensuring their satisfaction. Indeed, in order to fulfill a *stakeholder's* needs, the *contracts* established with others must be consistent with the stakeholder's *KPIs* in the sense that they allow him to reach his targets.

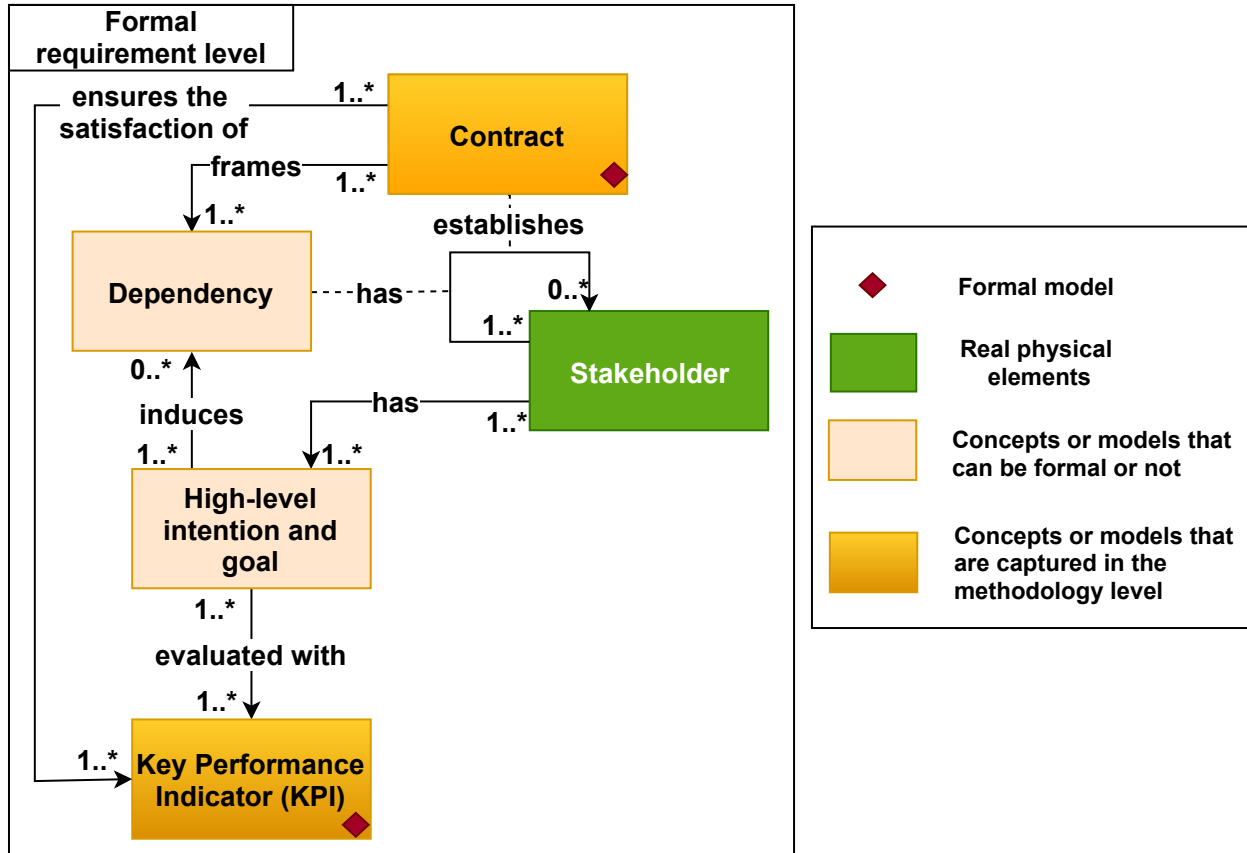


Figure 3-14: Metamodel of the formal requirement level

- **Positioning in the engineering cycle :**

The formal requirement level position itself after the identification of stakeholders goals and dependencies, and just before proceeding to the architectural design of the ME-CPS sub-elements, which are under the responsibility of stakeholders. This level is developed through an iterative process where stakeholders negotiate the terms (formal requirements) of the contracts in order to satisfy needs and fit into what they are able to guarantee to each other. Throughout the system design, the formal requirement level is fed with new constraints that are induced by the design choices that are made during all this phase.

- **Diagrams and models supporting the level :** The formal requirement level is supported by a graphical representation of the contracts between stakeholders as shown in Figure 3-15.

Inspired from [114], the formal requirement level structures the relationships between stakeholders using formal contracts which allows to mitigate the design complexity of

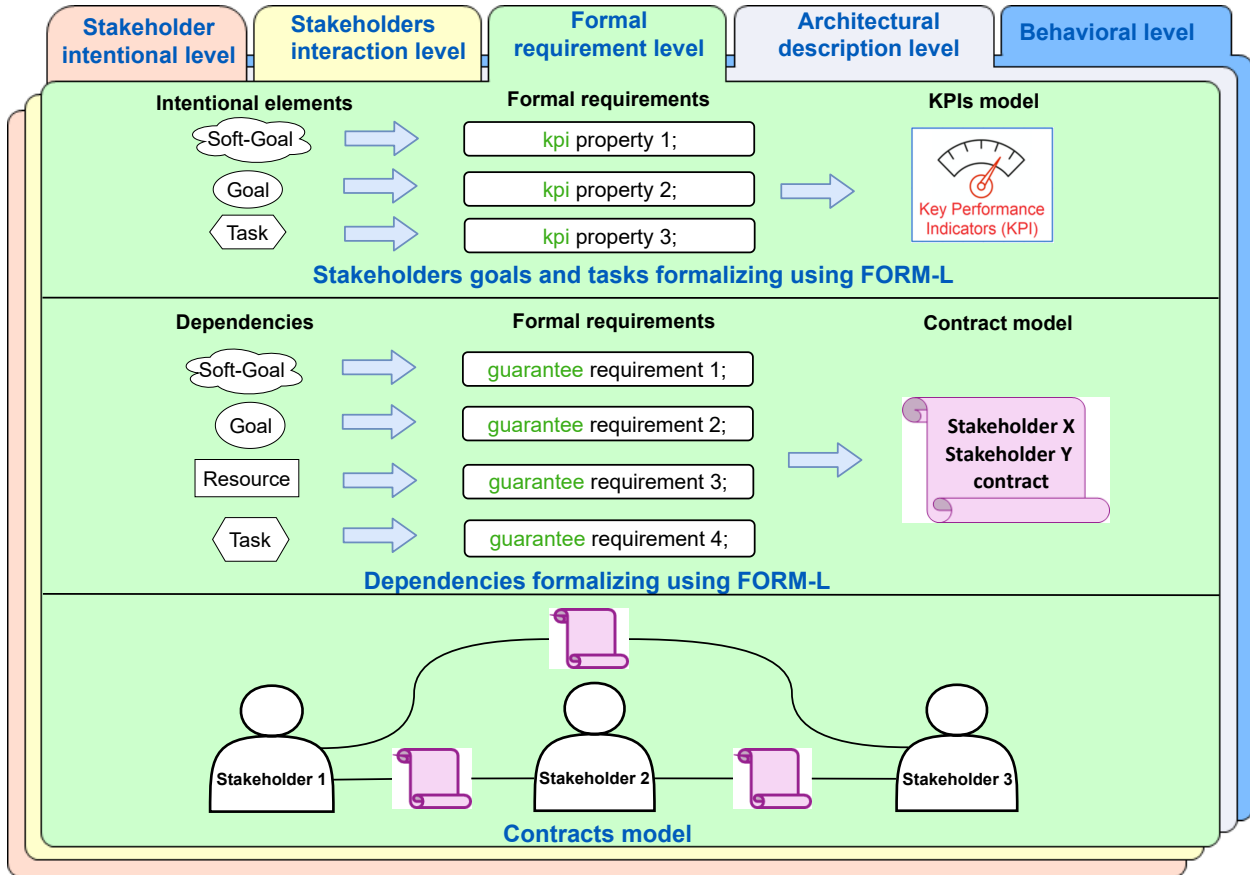


Figure 3-15: Formal requirements level

ME-CPS by decomposing the high-level challenges into more manageable problems. Indeed, establishing contracts between stakeholders enables to rigorously define the knowledge and the information a stakeholder needs to know about the other actors of the system. A contract model introduces envelopes of the behaviors to which each stakeholder must comply.

As stated in section 3.4.4, a contract is made of two main elements: assumptions and guarantees. The guarantees of a stakeholder refer to the requirements that must be ensured to the other parties (a contract may bind more than two parties). The assumptions of a stakeholder refer to what should be guaranteed by the other parties. In a context of a ME-CPS including multiple elements that are managed independently by different parties, contracts allow stakeholders to avoid being dependent on detailed behavioral models of sub-systems that are not under their responsibility, which are therefore very difficult to model and cannot be easily obtained from the responsible stakeholders. Therefore, each stakeholder can separately design the system under his responsibility by considering the contract assumptions as a reliable representation of

the environment of the system under his responsibility. With this vision, each stakeholder can represent the elements outside his scope using the level of detail that is appropriate to what is required by the design of his system. These representations must comply with the assumptions of the contract.

- **Syntax of the level** : The graphical elements used for the contract topology between stakeholders are shown in Table 3.3.


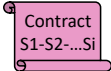
Graphical element	Element description	Example of use
	Stakeholder Si	Consumers, Producers, Regulators, etc.
	Contract between stakeholders S1, S2, ... and Si	Contract between an electricity producer and the electrical distribution operator.

Table 3.3: Graphical elements used by the formal requirement level

- **Tools and libraries supporting the level** : Within the framework of CREMA methodology, requirements are formally captured using FOrmal Requirements Modeling Language (FORM-L)[114] and Extended Temporal Language (ETL) [61].

FORM-L (FOrmal Requirements Modeling Language) [114] was developed in the framework of ITEA2 European project MODRIO [20]. An initiative for the standardization of this language is in progress within the ITEA3 project EMBrACE [21], under the name of CRML (Common Requirement Modeling Language). The relevance of this language lies on its capacity to formally express and organize requirements as sets of constraints on objects physical time and spatial location using a syntax that is intuitive enough for engineers, to the contrary of temporal logics such as LTL or CTL[49], and other (temporal) languages mentioned in [47] that use mathematical notation. The ability to express formal requirements in a language that is close enough to natural language is important in order to ensure flexibility for designers with different backgrounds to express their needs in a unified framework and allows automatic translation into target models for automatic verification [61]. With FORM-L a requirement such as “*When the cooling system is in operation, the system temperature must not exceed 25 degrees Celsius for more than 5 seconds .*” can formally be expressed. Figure 3-16 shows how this requirement is written in FORM-L language.

ETL [61] is introduced as a formal temporal language covering the temporal aspects of FORM-L. Indeed, it was developed in order to capture the generated target models in terms of temporal constraints while referring to temporal and timing properties,

```

requirement nominalTemp is // Requirement name
  during coolingSystem.inOperation // Boolean indicating whether
  after temp > 25*_C within 5*s // the cooling system is operational
  ensure temp <= 25*_C

```

Figure 3-16: Example of a requirement expressed in FORM-L language

as well as physical variables. ETL differs from the existing temporal logic languages such as LTL [49] in the sense that ETL captures real-time constraints and do not require to model the systems using finite-state machines. Therefore, LTL and ETL are dedicated to completely different kinds of analysis. LTL is intended for model checking analysis with logical representations of properties and system behavior. In contrast, ETL is meant for real-time co-simulation of properties with behavioral models in order to verify the system behavior against requirements. The key idea of ETL is the separation between time locators defined as intervals between two events and conditions to be checked. Using this approach enables representing real-time physical constraints which are not the case of existing temporal logic languages.

In ETL, a requirement is a couple $R = \varphi \otimes P$ where φ is the condition and P is the time period. The value of the requirement R is different from the value of the condition φ . The reason is because the requirement is given the delay P to satisfy the condition φ . In other words, it is required that φ must be satisfied at the end of the time period P . The fact that R is satisfied or violated must be reported as early as possible. This means that if the condition φ is violated at some instant within P and that it cannot be satisfied after that, then the requirement is considered to be violated. The same applies if φ is satisfied at some instant within P . Therefore, the value of the requirement can take four values: “undefined”, “undecided”, “true” or “false”. The value of the requirement is undefined before the start of the time period, which means that the requirement is not applicable. The value of the requirement is undecided inside the time period as long as no decision can be made whether the requirement is satisfied or not. The value of the requirement is true or false as soon as the decision can be made whether the requirement is satisfied or not, and no later than the end of the time period. The value of the requirement can be used as the condition of another requirement in order to express requirements on requirements. This is why the condition φ is also a four-valued Boolean. As a simple illustrative example, let us consider the case of a writer that must deliver a book in a time frame comprised between the signature of a contract with his editor and the deadline specified in the contract. Before the signature of the contract, the requirement is not applicable: its

value is undefined. After the signature of the contract and before the deadline, the value of the requirement is undecided as long as the book is not completed. As soon as the book is completed, and before the deadline, the value of the requirement is true (satisfied). If the deadline is reached with the book not being completed, then the value of the requirement is false (violated).

Requirements can be combined with logical operators. Let us consider two requirements $R_1 = \varphi_1 \otimes P_1$ and $R_2 = \varphi_2 \otimes P_2$. By definition, the logical conjunction of R_1 and R_2 is:

$$R_1 \wedge R_2 = val(R_1) \wedge val(R_2)$$

In the same manner, the logical negation of $R = \varphi \otimes P$ is defined by :

$$\neg R = \neg val(R)$$

The values of the requirements follow a four-valued Boolean algebra introduced in Table 3.4 and Table 3.5. This algebra is conceived in order to grant meaningful values to the logical conjunction and negation of requirements. The other logical operators are derived using the Morgan laws and it turns out that doing so gives also meaningful values to the results of applying those logical operators to requirements (logical disjunction “ \vee ” and logical implication “ \Rightarrow ”).

$\varphi_1 \wedge \varphi_2$: Logical conjunction				
$\varphi_1 \backslash \varphi_2$	true	false	undecided	undefined
true	true	false	undecided	true
false	false	false	false	false
undecided	undecided	false	undecided	undecided
undefined	true	false	undecided	undefined

Table 3.4: Truth table of the logical conjunction (from [61])

not φ : Logical negation				
φ	true	false	undecided	undefined
$\neg\varphi$	false	true	undecided	undefined

Table 3.5: Truth table of the logical negation (from [61])

Refer to [61] for more details about ETL syntax and semantic.

ETL was implemented in the multi-domain modeling language Modelica [81] giving birth to the library called ReqSysPro. So far, the translation between FORM-L and ReqSysPro is done manually using the ReqSysPro blocks. A compiler is under development to automate this task in partnership with Inria/Sciworks Technologies and with the EMBrACE consortium.

Figure 3-17 gives an overview of how the requirement mentioned in the previous chapter is built using ReqSysPro blocks.

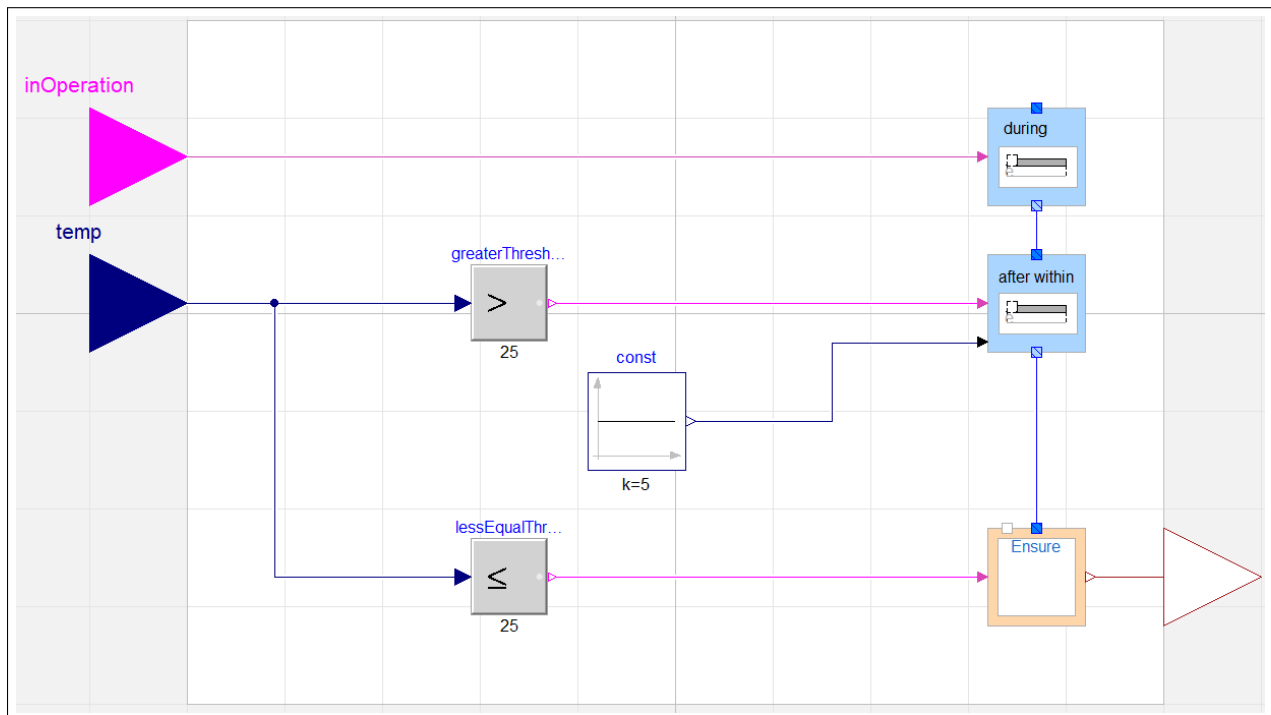


Figure 3-17: Example of a requirement built using ReqSysPro library (corresponds to the requirement in Figure 3-16)

- **Relationships and interaction with other levels :**

The formal requirement level plays a central role in CReMA methodology as the other modeling levels are mainly developed to provide data that will challenge the formal properties that are specified in contracts and in stakeholders' KPIs models.

- Stakeholder intentional level: As mentioned previously, this level formalizes and refines the stakeholders' intentional elements. Figure 3-15 illustrates how the intentional elements are transformed into KPIs and dependencies into requirements that make up a contract.
- Design justification level: the justification level considers the requirement level as a starting point. The requirements specified in the contracts are considered as high-level claims that are refined along with the design. These claims are then supported by an argumentation chain which demonstrates that these claims were fulfilled by the designed system.
- Architectural description level and behavioral level: The formal requirement level interacts with these two levels during the whole design process, essentially for making verification. On the one hand, these levels are gathered in verification models in order to validate each design step by checking the compliance of the architectural models and the behavioral models with respect to the contracts. The bindings and observation models are used in order to ensure the interoperability between them in the verification models. On the other hand, the decisions made throughout the engineering process such as the choice of architecture and its components continuously induce new constraints that take part in the contracts and which will be iteratively challenged with the architecture and behavioral model.
- **Contribution to the formal verification model :** The formal requirements level provides the executable requirements dimension of the verification model, including the contracts and stakeholders KPIs with defined targets.

Throughout the engineering cycle, once a formal requirement model is set, early verification can be made even if no behavioral model is developed yet. The idea is to challenge requirements using random but framed sequence generators that complies with a behavioral envelope that was defined in advance. Sequence generators are considered as abstractions that are substitutes to the behavioral models in the preliminary studies where the design to be compliant with contracts is far from being known. This approach will allow stakeholders to evaluate the compliance between their KPIs and the ongoing contracts' requirements. It can also be

used in order to verify the compliance between the guarantees and assumptions of different stakeholders, and therefore identify inconsistencies and contradictions.

A theoretical description of these verifications is introduced in Appendix C.

3.7.4 Design justification level

- **Level name** : Design justification level
- **Related view** : Transversal level between the Stakeholder view and the System view.
- **Level objectives** :

In contrast with other CReMA levels, the justification framework has the particularity to operate across multiple levels, notably the formal requirement level and other “system” levels, as shown in Figure 3-8. This framework aims at ensuring a design continuity and traceability between the formal contracts and the systems designed to satisfy the requirements they contain as well as stakeholders’ needs.

As a reminder, the justification framework refers to the means of capturing and structuring the reasoning chain that is followed throughout the design process of a complex system. The goal is to ensure that the argumentation and the reasoning chain provide a sufficient confidence level that guarantees that the system meets its high-level missions and requirements.

This is essentially meaningful when establishing contracts with safety authorities that require stringent pieces of evidence to justify the right operation of safety systems. Moreover, this argumentation can also be of major added value during negotiations between stakeholders. A stakeholder can for example demonstrate using this approach that to guarantee a level of system performance, the other stakeholder will have to make compromises in regards to time delay or cost.

The potential of the design justification level lies in the fact that it provides relevant means to implement some SE processes and recommendations [42] that have been so far very complex dealing with. Among them can be cited the traceability that is generally limited to simple links between requirements and implementations. The justification framework allows adding a demonstration dimension based on rigorous rules that guarantee consistent design all along the engineering cycle. This ensures the consistency between the holistic view of a system and its constituting elements that are developed separately.

- **Level metamodel** : not concerned

- **Positioning in the engineering cycle :**

The design justification is a process that needs to be carried out all along with the design in order to guarantee traceability and ensure consistency amongst different design levels. In CReMA methodology, the justification framework is considered to take place starting from the formal requirement level until the detailed design passing through architecture design. At early intentional levels, the traceability is guaranteed using the i^* framework elements and the SD and SR diagrams. In more detailed phases, the design justification level supported by the Claim - Argument - Evidence (CAE) framework was used as a common thread ensuring traceability amongst requirements and detailed design.

- **Diagrams and models supporting the level :**

In order to capture and organize the design argumentation chain followed by stakeholders, the justification level is supported by the CAE framework that was introduced in detail in the previous section 2.6.2.

As a reminder, the CAE framework captures the reasoning chain using a graphical representation including three main elements :

- **Claims**, “*which are assertions put forward for general acceptance. These are typically statements about a property of the system or some subsystem. Claims that are asserted as true without justification become assumptions and claims supporting an argument are called sub-claims.*”
- **Arguments**, “*which link the evidence to the claim. These are the “statements indicating the general ways of arguing being applied in a particular case and implicitly relied on and whose trustworthiness is well established”[136], together with the validation for the scientific and engineering laws used. In an engineering context, arguments should be explicit.*”
- **Evidence**, “*is used as the basis of the justification of the claim. Sources of evidence may include the design, the development process, prior field experience, testing (including statistical testing), source code analysis or formal analysis.*”

In this framework, stakeholders start by making claims that are assumed to be correct (e.g. that the system complies with a specific requirement), and that will be further justified using pieces of evidence that are supported with arguments. A claim has value only through the appropriate evidence supported by arguments that justifies the viability of the claim. The links between claims, arguments and evidence are synthesized in Figure 3-18.

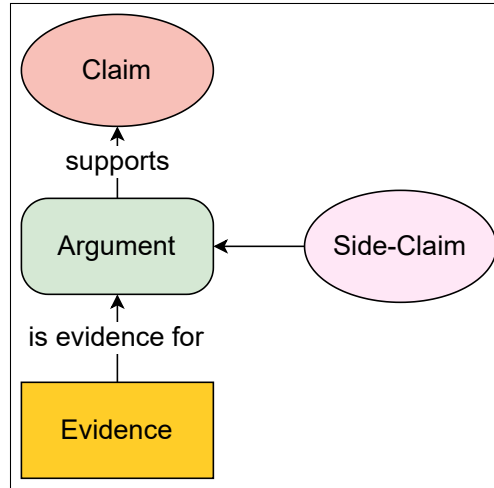


Figure 3-18: Basic graphical elements of CAE (from [116])

- **Syntax of the level :**

The graphical elements used by the CAE framework are shown in Table 3.6


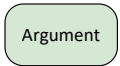

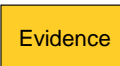
Graphical element	Element description	Example of use
	Claim	The system S_1 is safe
	Argument	Decomposition
	Side-Claim	System S_1 is composed of sub-systems S_{11} and S_{12} $S_1 \Leftrightarrow S_{11} \wedge S_{12}$
	Evidence	Safety analysis results proving that systems S_{11} and S_{12} are safe.

Table 3.6: Graphical elements used by CAE framework

- **Tools and libraries supporting the level :**

The CAE framework is supported by multiple graphical tools such as the Assurance and Safety Case Environment (ASCE) [8], the Assurance Case Automation Toolset (AdvoCATE) [6] developed by NASA Ames Research Center, and a Typed Assurance Case Editor (D-Case) [10].

- **Relationships and interaction with other levels :** In order to ensure continuity between the formal contracts definition phase and advanced design phases, a comparison was done between the argumentation building blocks of the CAE framework and

the engineering architectural and behavioral modeling practices, which has led to identifying close similarities between them. The matching between CAE building blocks and system modeling is as follows (refer to section 2.6.2 for the definition of the CAE building blocks):

- Concretion blocks: these blocks are equivalent to the concept of “replaceable classes and components” that is used during the design in order to replace abstract elements with more detailed components when a more precise representation is needed. In the argumentation process, replacing a class or a component must be justified by demonstrating that the substitute element represents at the very least the properties of the parent element and ensures less abstract and more measurable aspects.
- Decomposition blocks: these blocks match with the standard concept of decomposition used in engineering design and is the most commonly used refinement process as it allows breaking the complexity of a system into more manageable elements. In the argumentation process, the consistency of the decomposition must be demonstrated by checking the equivalence between the conjunction of sub-elements and the parent component.
- Calculation blocks: these blocks are equivalent to “observation models” that compute a system property or function by observing other system properties. The argumentation process is justified by having rigorous formulas linking the different system properties.
- Substitution blocks: similarly to the concretion blocks, the substitution blocks match with the modeling practices of “replaceable classes and components”, but for another purpose. Instead of replacing abstract elements models with more precise ones, elements are substituted with other items that can be considered as equivalent in a specific context. The goal of this substitution can be simplifying a heavy model including a large number of similar instances with a smaller but representative model for a specific analysis. An example would be substituting a large model containing 1000 buildings which computes the specific electricity need of each one of them, with a smaller model that aggregates the global need of all buildings. This substitution is relevant when designing a production plant that aims to provide electricity to these buildings. This analysis does not require knowing the individual electricity needs of each building. Another example would be the substitution of a system’s environment model with a sequence generator that randomly generates behaviors inside a defined envelop. The argumentation process is justified by demonstrating that the substituted element is equivalent to

the replaced one and that the satisfaction of a claim on the latter automatically induces the satisfaction of the same claim on the substituted element.

The added value of the justification framework in the existing design process using modeling and simulation is that it guarantees the traceability between different design iterations, and demonstrates the consistency of the refinement with the parent pattern.

- **Contribution to the formal verification model :**

The design justification level does not directly feed the verification model which allows checking that the designed systems meet their stakeholders' needs and mutual contracts. However, it allows verifying the consistency of each engineering step, which increases the chances of success of the design.

3.7.5 Architecture description level

- **Level name :** Architecture description level

- **Related view :** System view

- **Level objectives :**

The architectural description level is introduced into CReMA methodology in order to capture the static properties of a system or a set of systems, the interactions between them, the interfaces, and the environment with which they interact.

In the context of developing a multi-stakeholders complex system, the architecture description is not only used to give a system view on the design under multiple viewpoints, but it is also used to delimit the scope of responsibility of each stakeholder, which will be guaranteed by the notion of stakeholder's scope.

Similarly to the stakeholder interaction level, the interactions are characterized using the four elements the resource being exchanged, the means of exchange, the protocol of exchange, and the rules of exchange.

- **Level metamodel :**

The architecture description level captures the *architecture* exhibited by one or multiple systems and their interactions with each other. The notion of interaction gather the *exchange channels* and the *interfaces* between systems. The modeling of the *systems' exchange channels* is assumed to be a part of the *system* representation and thus not considered in the architecture description level, however, the *interfaces* between *systems* are captured in this level.

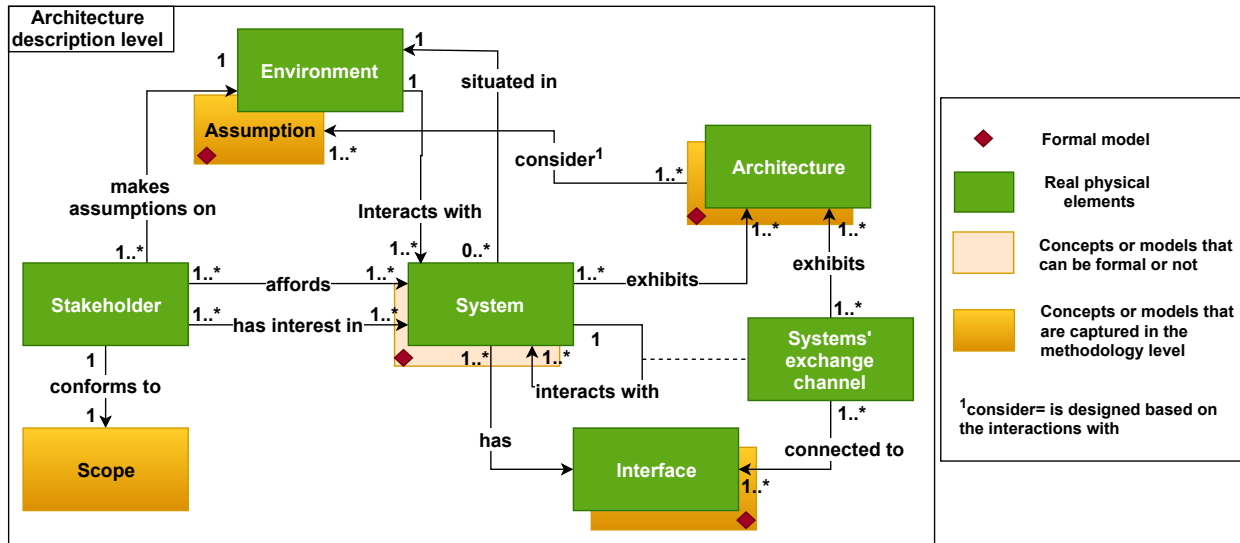


Figure 3-19: Metamodel of the architecture description level

The *architecture description model* is designed taking into account the interactions that the system has with its environment, and thus considers the *assumptions* taken by a stakeholder on his *environment*.

Each *stakeholder* is considered to have a *scope* that is limited in time and can evolve during the system *lifecycle* as a result of negotiations or modifications in *stakeholder* roles.

- Positioning in the engineering cycle :** The architecture description level is a work product that is developed as part of the early design phases. It is conducted after defining stakeholders' preliminary requirements in order to draw the first sketches answering their needs. The architecture description level can also be conducted during early feasibility studies which precede the requirements definition when dealing with ME-CPS that are strongly constrained by legacy systems. Once the architectural description models are developed, dynamic behavioral models are developed in the next methodology level. These models assist stakeholders to choose among the alternative architectures the one that fits best with their needs.
- Diagrams and models supporting the level :**

The architecture description level is supported by the architectural description model that represents system elements using blocks and links them using connections and interfaces.

Compared to classical architecture descriptions, the notion of stakeholder's scope was

added in this view in order to accommodate our multi-stakeholder view. It allows on one side to define the scope of action of each stakeholder and assign him the elements which he is responsible of as shown in Figure 3-20, and on the other side, it allows identifying the physical and cyber interactions between the stakeholders' scopes that complete the stakeholder interaction view. The architectural elements under the responsibility of each stakeholder will be thereafter integrated as a new block in the stakeholder description model.

We emphasize that the architecture description as well as the stakeholders' scopes can significantly vary over the different stages of the systems life cycle.

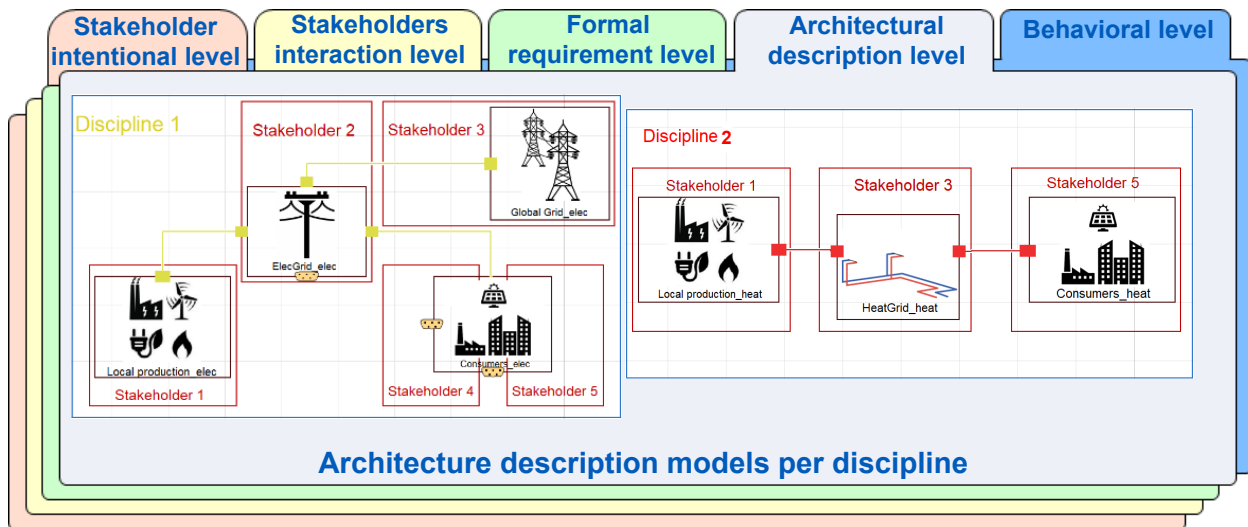


Figure 3-20: Architectural description level

- **Syntax of the level :**

The graphical elements used to build the architectural description model are shown in Table 3.7


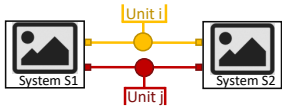

Graphical element	Element description	Example of use
	Architecture description	Power production plant, building, electric grid, control system, etc.
	Interactions and interfaces	Systems S1 and S2 interact by exchanging two elements having the units 'i' and 'j'. The exchanged resources pass through the square shaped interfaces (connectors).
	Stakeholder perimeter	A perimeter including the electric grid and the control system.

Table 3.7: Graphical elements used for building architecture description model

- **Tools and libraries supporting the level :**

The architecture description model can be developed using different possible platforms depending on the concerns of its designers, depending on the viewpoint from which the system will be represented, and depending on whether or not the architectural level is part of a formal and executable modeling process. It can indeed be limited to graphical representations reflecting the interactions between systems under the responsibility of stakeholders as it is the case when using tools such as SysML [119] or Capella [128]. However, it is highly recommended in CReMA methodology to integrate the architectural description as part of an executable model in order to have the means for making formal but rough analysis at early design phases. Amongst the tools that enable performing these analyses is the multi-methods simulation environment AnyLogic [7]. With this tool, a dynamic architecture description of the system can be represented including the static engineering data, which allows observing and evaluating the flows between architecture description components.

Aside from SysML and Anylogic, the architecture description modeling can be also done using the behavioral modeling tools that are used by designers. The architecture description would ensure a double mission by representing the system architecture and also structuring the behavioral modeling, the connections, and the interfaces between elements. Throughout the validation of CReMA methodology on the PowerGrid case study, the architectural modeling was done in the modeling language [81] supported with graphical blocks representation, which is indeed not the best suited for this task, yet it was convenient structuring the different models introduced by the methodology in the same platform.

- **Relationships and interaction with other levels :** The architecture description level is mainly related to the following levels:

- The formal requirement level: they are connected when developing the verification model where requirements are connected to the architectural description model and the behavioral model.
- The stakeholder interaction level: they both make a focus on the interactions but between different elements. On the one hand, the stakeholder interaction level emphasizes the interactions between stakeholders, and on the other hand, the architectural description level emphasizes the interactions at a system level. In CReMA methodology, we consider that the architectural description level feeds the stakeholder interaction level in order to guarantee a holistic view of all the interactions between stakeholders. That way, stakeholders will be able to conduct negotiations knowing all the ties that bind them to each other. However, this is not a bijective process. In fact, among the interactions identified at the stakeholder interaction level are those that are intrinsic to the exchanges between the stakeholders and do not make sense in a system vision of the ME-CPS. Examples of these interactions are financial and administration exchanges.
- The behavioral level: this level represents the dynamic aspects of the architecture description elements. The structure set by the architecture description level can be retrieved in the behavioral modeling.

- **Contribution to the formal verification model :**

The architecture description level is an important contributor to the construction of the verification model because it provides system design data that is specific to the architectural model, and that can be required to assess particular requirements. An example would be the NPSH characteristic curve of a pump that defines the minimum required inlet pressure that can only be found as part of the pump architectural description model. This data will enable evaluating a property such as a pump cavitation, which is not necessarily evaluated in the behavioral model.

3.7.6 Behavioral level

- **Level name :** Behavioral level
- **Related view :** System view
- **Level objectives :** The goal of the behavioral level is to give stakeholders the possibility to model the dynamic behavior of the systems under their responsibility in an independent way. This level is built upon the idea that stakeholders are part of different entities and have different knowledge on different disciplines that remain confidential, however, they can choose to share it in some cases for global optimization.

Therefore, no particular tool or paradigm is recommended at this level. The idea is to give engineers the freedom to use their customary domain-specific tools to model their systems behaviors.

The behavioral level plays a major role in designing dependable ME-CPS as it allows us to precisely understand their behavior under different scenarios. In CReMA methodology, stakeholders may use different kinds of modeling paradigm which fit with the required level of detail of the design phases in order to choose over multiple potential solutions the one (or the ones) that answers best their objectives and their obligations.

• **Level metamodel :**

The behavioral level depicts the dynamic aspects of the *System* of interest as well as the interactions between its elements. The notion of interaction gathers the *systems' exchange channels* and the *interfaces* between systems. The behavioral model exhibits the same *interfaces* that were earlier defined at the architecture description level. The modeling of the *systems' exchange channel* is assumed to be part of the *system* description model, however, the *interfaces* between *systems* are modeled separately.

The behavioral modeling of a *system* takes into account the same *assumptions* on the environment that were considered at the architectural description level. These *assumptions* can be refined at the behavioral level to fit with the required level of detail of the design phase.

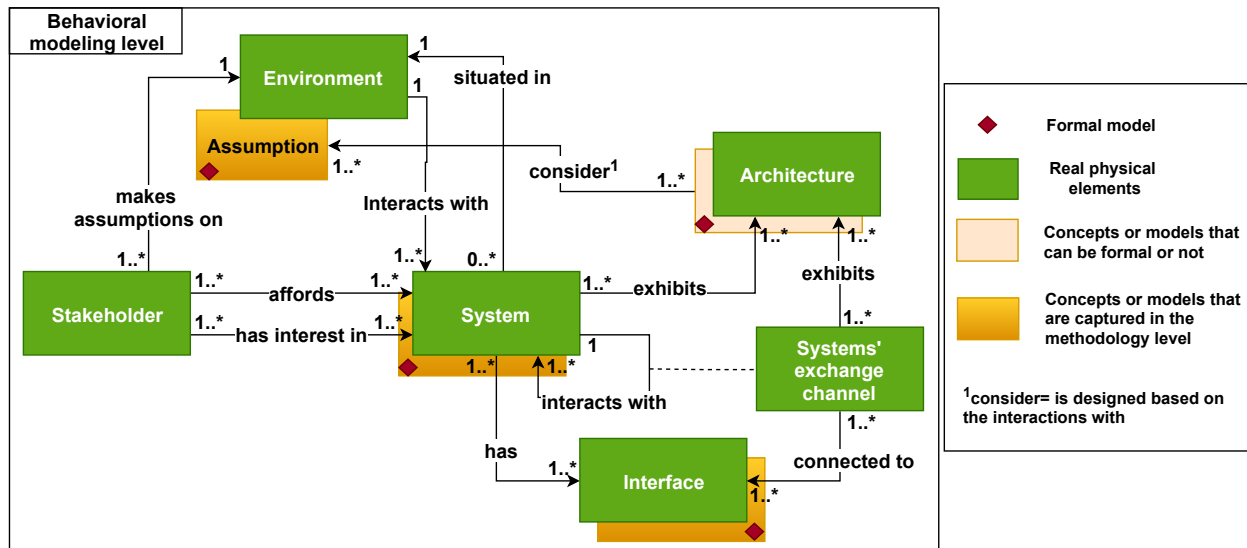


Figure 3-21: Metamodel of the behavioral level

• **Positioning in the engineering cycle :**

The development of the behavioral level is done within the advanced design steps, after identifying one or multiple system architectures that are likely to answer the contracts requirements and stakeholders KPIs. Similarly to the other CReMA methodology levels, the behavioral modeling is an iterative process that goes through multiple modifications and refinement steps throughout the various design phases that were mentioned in the global overview in section 3.5.

- **Diagrams and models supporting the level :**

CReMA methodology does not display any diagram for the behavioral modeling level as it considers that stakeholders will develop their behavioral models for systems under their responsibility in a separate manner and using different domain-specific tools as shown in Figure 3-22.

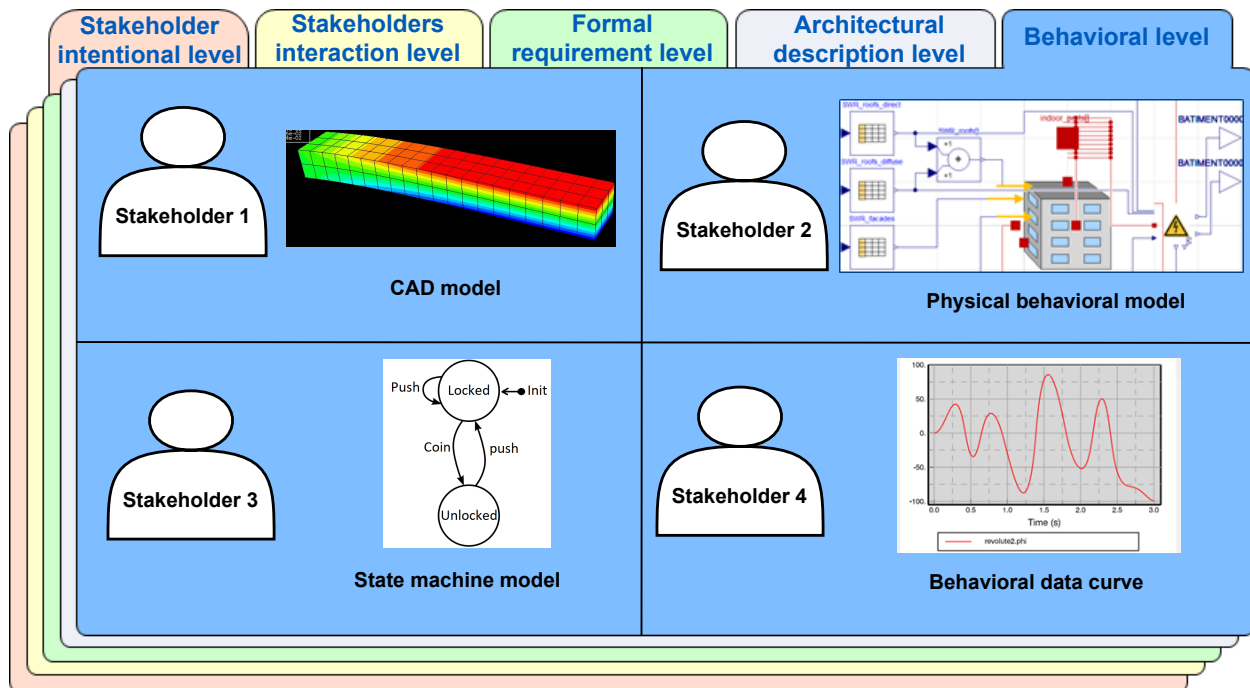


Figure 3-22: Behavioral level

- **Syntax of the level :** not concerned

- **Tools and libraries supporting the level :**

As mentioned earlier, no specific modeling tool or library is recommended at this level because each discipline has its own domain-specific tools and languages.

- **Relationships and interaction with other levels :**

The behavioral model allows to model the dynamic aspects of system components that were depicted at the architectural description level. Therefore, the behavioral level contains similar components as the architectural description level but embedded with models representing their dynamic behaviors. The behavioral model may also exhibit the same structure between its components as the architectural description level. However, these two rules do not apply in general as one or multiple architectural description components may not be represented in the behavioral model.

The behavioral level interacts with the formal requirement level and the architecture description level when developing a verification model. The interoperability between the models developed at each level is ensured by means of bindings and observation models.

- **Contribution to the formal verification model :**

The behavioral modeling level plays a major role in the verification model. It provides a representation of the dynamic variation of the system over the simulation time and allows challenging requirements and evaluating stakeholders KPIs.

CReMA methodology makes it possible to connect the behavioral models to the requirements models without having to modify the behavioral models for that purpose (provided that the behavioral models can answer the questions of the requirements models such as e.g., is this pump started?). This is a considerable improvement with respect to current practices where checks are made by manually inspecting whether the simulation results comply with the requirements, with risks of errors and omissions in the verification. Test automation allows, in particular, to perform FMEAs at every step of the design process, and not only at the end of the design process when errors are difficult to detect, in particular in the absence of a rigorous elicitation procedure, and costly to repair.

3.7.7 Modular construction of the verification model

The idea behind the verification model is not to create a single global testing model encompassing all stakeholders' descriptions, architectures, and behavioral models because such objective is difficult, or even impossible to reach at a reasonable cost. The main goal is to rather have a modular modeling framework that allows making partial verification according to different purposes as shown in Figure 3-23. The contracts play a major role in having the flexibility to build "on-demand" verification models. Indeed, the contracts allow each stakeholder to define assumptions that represent behavioral envelopes of external interactions with the systems under the responsibility of others. Thus, each stakeholder can

undertake his own verification on his system separately under the reasonable assumption that the behaviors of the other systems remain within the scope of the contracts.

The framework gives the possibility to structure the verification models in two ways :

- **Single-stakeholder verification model**, as shown in the blue frame of Figure 3-23. The verification model includes the KPIs, the contracts, and the test sequences that comply with the assumptions on the other systems (i.e. the guarantees from the other stakeholders as defined by the contracts). Hence, the test sequences stand for the behavioral models of the other stakeholders which do not need to be provided. They can be expressed as simplified deterministic or stochastic trajectories within the envelopes defined by the assumptions. The idea is to be able to generate such test sequences automatically from the formal expressions of the assumptions. Such an endeavor is under way within the EMBrACE project[21]. Therefore, for the single-stakeholder verification model, only the contracts with the other stakeholder are needed.
- **Multi-stakeholder verification model** as shown in the red frame (between two stakeholders) and the orange frame (between more than two stakeholders) of Figure 3-23. The objective is to collaborate more tightly with other stakeholders when necessary, for instance when there are control loops across stakeholders. A tighter integration of the models from the different stakeholders involved including KPIs, contracts, architectural descriptions, and behavioral models is done to perform co-simulation. Considering that the multi-stakeholder verification model represents the tight integration of several stakeholders, the multi-stakeholder model is essentially the same as the single-stakeholder model with respect to the stakeholders not involved in the integration which can view the integrated group as a single stakeholder. In particular, the environment of the integrated group can be represented as test sequences describing the behavior of the other stakeholders, as for the single-stakeholder verification model.

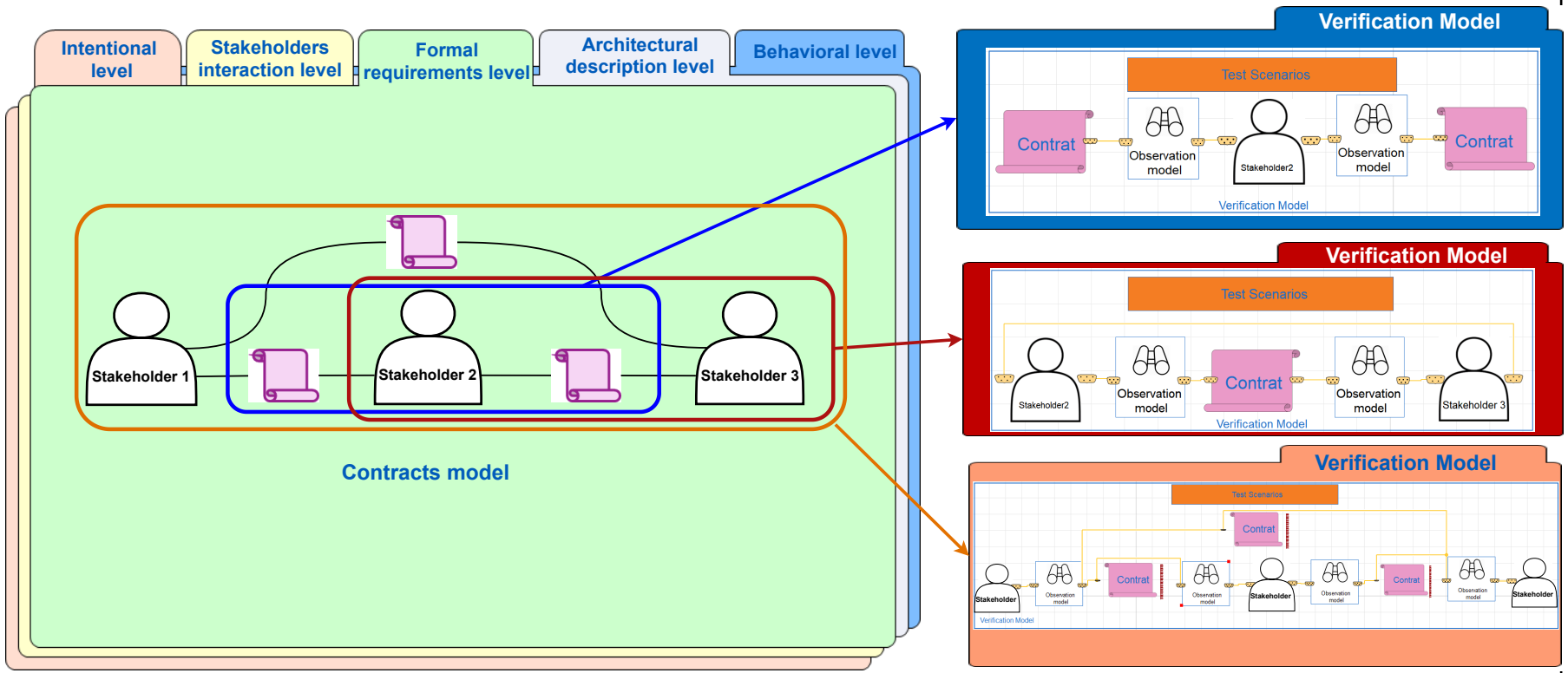


Figure 3-23: Modular verification model

3.8 Summary

CReMA methodology is a multi-level approach that puts the stakeholders at the center of the design process by making the assumption that the system will be correctly designed and operated if all stakeholders reach a common agreement in the form of formal contracts that can be simulated for verification and validation before signature.

The novelty of CReMA methodology is its ability to break away from the classical view that is centered on a single analyst who considers the external stakeholders of a complex system as sources of constraints that need to be satisfied, without being able to integrate them into an overall design framework (when it is appropriate) in order to optimize the system in a holistic way by seeking common agreements and not from a single point of view.

CReMA methodology introduces a co-design approach built upon five modeling levels separated into two main views: the Stakeholder view and the System view. An additional level called the design justification level spreads over the two views and serves as a bridge that ensures the traceability between the stakeholder view and the system view of CReMA methodology. A meta-model of CReMA methodology was introduced in order to set an overview of all the concepts that make up the methodology and define the links between them. It was used as “pivot” diagram from which were allocated the concepts to the different levels of CReMA while making sure that all these concepts were implemented in a consistent way, with no redundancy and no oversight of any methodology key item.

In contrast to the state-of-the-art methodologies, CReMA has allowed bridging the gap between the early conceptual studies where stakeholders define their high-level objectives and conduct preliminary studies using abstract representations of their systems in order to assess its viability, and the more detailed design phases where stakeholders supported by engineering teams use formal domain-specific tools for behavioral modeling. Moreover, CReMA methodology is based on rigorous techniques that support the formal modeling of the system “digital twin” that is composed in our framework of (1) requirements models that represent the stakeholders’ interests, (2) formal contracts that represent the mutual obligations of the stakeholders, (3) formal assumptions that represent the uncertain behavior of the system environment, and (4) behavioral models that represent the possible solutions.

Through the use of bindings and observation models, CReMA methodology is of great help to disciplinary engineers since they are no longer required to adapt their practices in order to build “consenting” behavioral models, which will be used to challenge formal requirements. The bindings and observation operators are non-intrusive modeling concepts that are used to extract pieces of information from behavioral and architecture description models. These pieces of information are then transformed into variables and functional states that are compatible with the inputs of requirement models, allowing executable V&V.

Thanks to the implementation of the contract design for formalizing the relationships

between stakeholders, CReMA methodology gives these later the possibility to conduct their own analysis and design the system that will fulfill their obligations and guarantee their goals while having the appropriate hypothesis on their external environment (including the other stakeholders). This is ensured by the means of the assumptions specified in the contract which describes an envelope of all possible behavioral trajectories that the other stakeholders (with the sub-systems inside their scope) are allowed to take at a certain period of time.

Finally, a rigorous formalism is introduced in Appendix C characterizing the key operations that can be carried out by stakeholders throughout the negotiation process and during the validation of the agreements between each other. In this appendix, a new theoretical contract logic is presented. This formalism defines the main concepts of CReMA methodology such as contracts, KPIs, environment, and implementation, etc. as well as the key operations that can be applied to these objects throughout the design process such as contract refinement, KPIs satisfaction, or contracts signature. This formalism is considered to be independent of any tool, language, or modeling approach. It was inspired by the algebra presented by Inria in [53] and adapted to fit our challenges.

CReMA methodology was mainly prototyped using Modelica (except the intentional level developed using the i^* framework and the design justification level using CAE framework) because the only available implementation of FORM-L requirements is made possible through the ReqSysPro Modelica library [61]. The other toolchain elements pushing the scope beyond Modelica are under development within the EMBRrACE project [21].

Chapter 4

Validation of the methodology - the PowerGrid case study

This chapter validates CReMA methodology by making an application on the case study named PowerGrid which is an industrial research project at EDF-Lab. The PowerGrid study aims to propose a renovation of a local multi-energy district of “Vélizy” which is a city located in the Paris suburban area in order to make it “greener” and “smarter”, greener because it should involve more renewable energy, smarter because it should have a more efficient energy management system. These objectives are translated into the KPIs and contracts negotiated among the different stakeholders. The PowerGrid renovation represents a typical example of ME-CPS that include a scope of 719 buildings, a district heating network, a distribution electrical grid, some local production and back-up units, a telecommunication infrastructure and supply services.

In this project, a role-playing was established: A group of 20 engineers from different EDF departments and having different disciplinary backgrounds and expertise have played the roles of the different PowerGrid stakeholders, each one of them having intrinsic goals to be satisfied, and establishes contract with the other interacting stakeholders. This project is therefore representative of the coordination challenges that encounter large ME-CPS including multiple stakeholders.

In the following, we start by introducing the PowerGrid demonstrator, its objectives, the involved stakeholders, and the scenarios chosen to be analyzed. Subsequently, we describe how CReMA methodology was applied to address two major phases of the PowerGrid development. Simulation results of the verification models and their interpretations are presented at the end of each phase. Finally, the global conclusion summarizes the achievements made by the methodology application and gives some perspectives for further refinement steps allowing a better coordination between stakeholders.

4.1 PowerGrid story board

4.1.1 Overview

The PowerGrid is a demonstrator that is part of a research project at EDF Lab. This use case was inspired by the Efficacity institute [13] and concerns the local multi-energy system of a district located in Vélizy which is a city located in the Paris suburban area. It represents a typical example of ME-CPS including a scope of :

- 719 buildings from different types such as private homes, small businesses, and larger industrial installations.
- A district heating network.
- A distribution electrical grid.
- Local production and back-up units.
- Supply services.

The PowerGrid study aims to propose a renovation of the Vélizy district that is “greener” and “smarter”, greener because it should involve more renewable energy, smarter because it should have a more efficient energy management system. These objectives are translated into the KPIs and contracts negotiated among the different stakeholders.

In the framework of the PowerGrid project, the scope of 719 buildings was mainly considered because among the demonstrator objectives was to set up heavy models that would allow pushing the Dymola environment (the behavioral modeling tool used in the detailed design) to its limits. In order to have models of reasonable size that would allow us to analyze a large number of scenarios and test smart energy management strategies, we have chosen in the context of this thesis to limit ourselves to a scope of 20 buildings which are representative of the different kinds of the consumers in the PowerGrid district consumers. The production units were also scaled up for this matter.

The role of the thesis in the project was to first develop a methodology to coordinate stakeholders by the means of formal contracts combined with modeling and simulation. Once the methodology was mature enough, it was applied to the PowerGrid demonstrator. The engineers involved in the project have mainly provided the behavioral models representing the PowerGrid sub-systems under their responsibility using Dymola tool [12] that is based on Modelica language [81], and the formal requirements written in FORM-L language. In the framework of this thesis, the following tasks were established in collaboration with the project engineers:

- The capture of the high-level goals of the PowerGrid stakeholders and the dependencies between them at the intentional level using the i* framework formalism.
- The identification of stakeholders' interactions.
- The translation of FORM-L requirements using the ReqSysPro library into executable Modelica blocks.
- The definition of the architecture description of the PowerGrid.
- The capture of the design justification framework.
- The construction of the verification models by setting the bindings and the observation models between its four main components: requirements, architecture description, behavioral model, and test scenarios model.
- The analysis of the verification simulation results.

We emphasize the design justification level had been integrated into CReMA methodology after the design of the PowerGrid elements was done and the behavioral models were developed. Therefore, this level will be limited to some examples showing how it can be used to capture the design argumentation chain.

4.1.2 PowerGrid development phases

The renovation of the PowerGrid district is considered to be progressively carried on throughout a process divided into three phases:

- **Phase 0: Urban energy planning** that aims at setting the architecture of the future smart grid to meet high-level requirements such as “having a more autonomous energy system with lower carbon emissions” and macro-constraints related to investment costs, available local resources, etc.
- **Phase 1: Dynamic sizing** that aims at optimally designing and managing the district sub-components such as the electric and heat networks, the power plants, the buildings, etc. while taking into consideration stakeholders requirements and the operational and technical constraints (network stability, intermittent renewable energy, etc.).
- **Phase 2: PowerGrid operation**, which consists of implementing smart operation strategies to improve energy efficiency. The idea is using real-time control strategies on flexibility mechanisms (load shedding, electric vehicles smart charging, batteries, etc.) in order to maximize system performances by adapting the consumption to the fluctuating production of energy.

CRema methodology will be applied to phase 0 and phase 1 of the PowerGrid as phase 2 is still in process by the time this thesis is being written.

4.2 Phase 0: PowerGrid urban energy planning

4.2.1 Stakeholders, objectives and tested scenarios

In this first phase of urban energy planning, the high-level objectives that are the reasons for being of the PowerGrid are specified by the “Sponsors” who initiated the project. They have the responsibility for setting up the necessary framework and overall organization enabling the development of the PowerGrid. The unique sponsor that has been considered in the PowerGrid program is the “Town council”.

In the sequel, we consider that the town council is assisted by a service provider or design office enabling him to make technical analysis and evaluate his goals. Thus, referring to this stakeholder is used in a broader sense that includes him and the assisting entity.

The urban energy planning can be assimilated to preliminary studies conducted by the town council who initiated the project. This phase aims to first specify the town council’s goals in a more precise way and sketch out the landscape by making rough analyses to evaluate the feasibility and viability of the project regarding his interests. Besides, this phase aims to identify the key mechanisms on which the *town council’s* can act on to optimize his goals at minimum costs.

In order to identify the best-suited architecture of the PowerGrid district, the *town council’s* KPIs were evaluated for twelve scenarios that represent different levels of investments that can be made. These investments include efforts in terms of renewable energies and also buildings renovation for more efficient thermal insulation. For the sake of conciseness, only three scenarios are considered in this case study. They can be presented as follows:

- **No measures:** represents the landmark of the analysis as it refers to the initial state of the PowerGrid district before renovation. In this scenario, no effort is made towards the implementation of renewable energies in the area (i.e. 0% of buildings are equipped with PV panels).
- **Average:** represents an intermediate scenario with a “greener” effort made in regards to the renewable energies. (i.e. 50% of buildings are equipped with PV panels).
- **High PV:** represents the case with the highest possible investments in regards to the renewable energies. (i.e. 100% of buildings are equipped with PV panels).

4.2.2 Stakeholder intentional level

The stakeholder intentional level is composed of two models, the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. However, in this preliminary phase, only one stakeholder is considered, and thus no dependencies can be drawn. The SR will be the only diagram drawn in this section.

In the SR model shown in Figure 4-1, we consider that the *town council* had in the past some goals to achieve and tasks to perform denoted respectively X and Y. For specific reasons that can be political, societal, or even financial, a new ambition has emerged for having a greener and smarter district. This new goal will have a significant influence on all the future decisions taken by the town council regarding the district renovation.

In order to have a green and smart district, the town council has identified four sub-goals to make this happen: (i) Having renewable energy, (ii) Having a low carbon footprint, (iii) Being competitive, (iv) and Being self-sufficient.

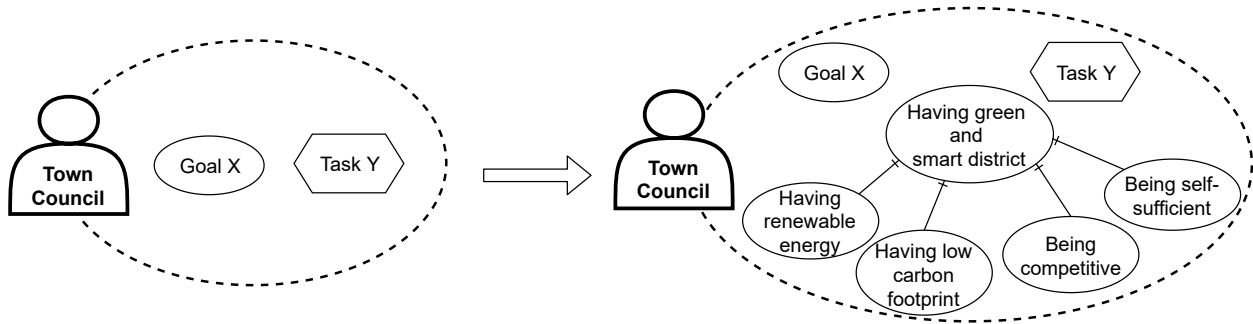


Figure 4-1: Observing goal change in the strategic rationale model of the town council

4.2.3 Stakeholder interaction level

This stakeholder interaction level is not concerned by this preliminary phase of the PowerGrid as the unique stakeholder involved up to now is the *town council*. Therefore, there are no interactions to be captured.

4.2.4 Formal requirement level

In order to embody his vision in the PowerGrid district, the *town council* has conceived the idea of the emergence of a new local electricity and heat supplier that will be named “*NeighbourPower*”.

The ambitions of the *town council* were reflected using the following KPIs (taken from [60]):

- kpi1 (ratio of renewables): In any given year, the energy provided to the clients of *NeighbourPower* (the local heat and electricity supplier associated with PowerGrid) should have at least 30% of renewables (taking account of self-consumption and imports from the national grid).
- kpi2 (CO2 emissions): Over a year, the CO2 emissions to produce the energy consumed by *NeighbourPower's* clients should be less than 60 g per kWh (accounting also for electricity from the national grid).
- kpi3 (competitiveness): For at least 90% of consumers, the yearly bill with *NeighbourPower's* best offer should be lower than the bill with competitor's best offer.
- kpi4 (self-sufficiency): Over a year, at least 60% of the energy managed by *NeighbourPower* or self-consumed by its clients should be produced locally.

It is reminded that KPIs targets that are set by the *town council* at this phase are only estimations that can be more or less close of what is actually feasible. Therefore, these targets may be revisited by the stakeholder once a clearer overview of the PowerGrid future performances is available.

These KPIs were then written using FORM-L language. Figure 4-2 taken from [60] shows the FORM-L model of kpi1. These KPIs were formalized and integrated into Anylogic

```

requirement kpi1 is when endYear
  ensure renewableEnergy >= totalEnergy*0.3;

// Renewable energy produced in a year by clients, nP and from natGrid import
renewableEnergy is yearlyIntegral(
  sum(for all c in clients: c.autoPower) + // Clients pv auto-consumption
  nP.renewableElec + // Renewable local electricity
  nP.renewableHeat + // Renewable heat
  nP.ngPower * natGrid.renewablesRatio); // Renewable imported electricity

// Total energy used in a year by clients, nP and natGrid imports
totalEnergy is renewableEnergy + yearlyIntegral(
  nP.nonRenewableElec +
  nP.nonRenewableHeat +
  nP.ngPower * (1-natGrid.renewablesRatio));

```

Figure 4-2: Modeling of kpi1 in FORM-L (from [60])

toolbox that will be used for representing the behavioral model of the PowerGrid at the urban energy planning phase. Since we only have a unique stakeholder at this phase of the PowerGrid, no contract is established yet.

4.2.5 Design justification level

The urban energy planning phase of the PowerGrid does not aim to make design choices and set the energy architecture of the district but rather aims to sketch out the landscape and roughly identify the potential solutions that fit with the *town council's* goals. Therefore, the design justification framework is not concerned yet in this phase of the PowerGrid.

4.2.6 Architecture description level

A considerable part of the complexity of the PowerGrid is due to its multiple components that interact at different dimensions. In order to tackle this complexity and have a clearer vision on the PowerGrid sub-systems, three architecture descriptions were developed which represent the PowerGrid from three disciplinary points of view: (i) Electricity, (ii) Heat, and (iii) Economics.

Figure 4-3 depicts the architecture of the PowerGrid from the electrical point of view. Four systems are identified exchanging electricity (i) the local producers including renewable energies and fossil energies, (ii) the consumers that represent the local buildings and industries, but can also include PV panels when the consumers decide so, (iii) the distribution grid that absorbs the electricity from producers and feed the buildings, and (iv) the high voltage or national grid called *Global Grid* that also feeds the distribution grid with electricity.

To name an example of data captured by the architectural description elements, we can cite the nominal electricity production capacity of the producers, and the ratio of the district consumers that subscribe to *neighbourPower* supply services.

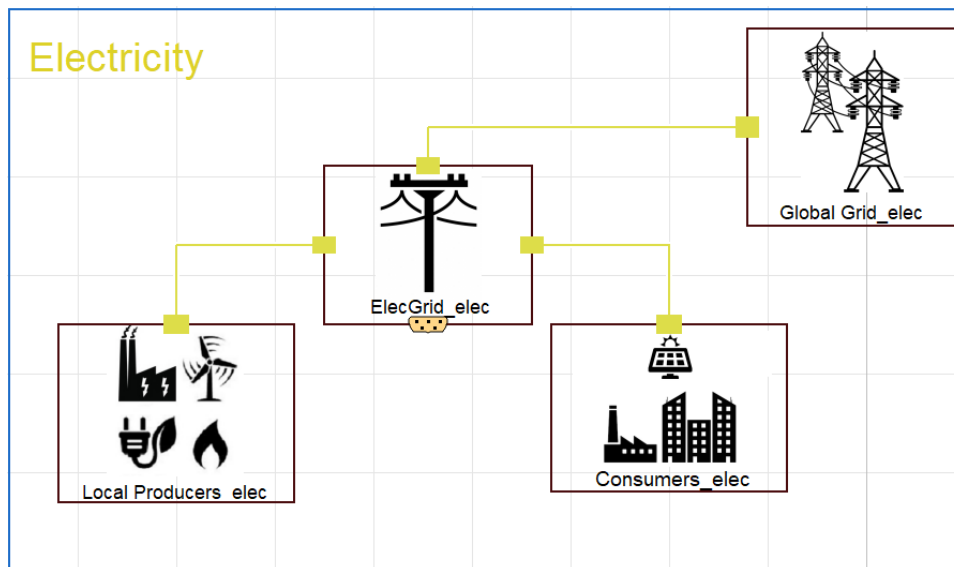


Figure 4-3: PowerGrid electric architecture

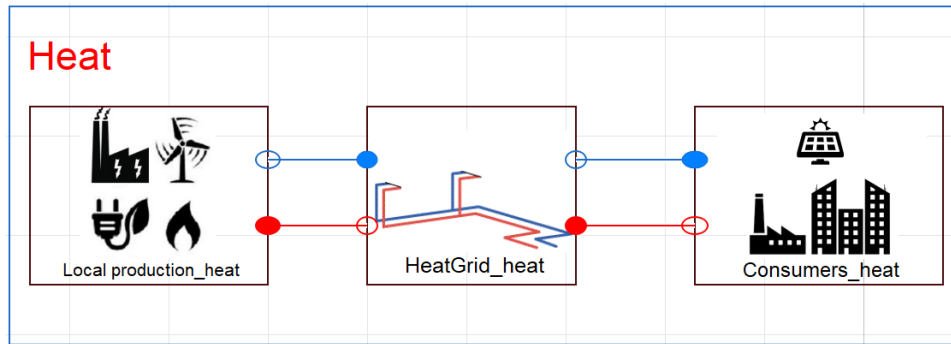


Figure 4-4: PowerGrid heat architecture

Figure 4-4 depicts the architecture of the PowerGrid from the heat point of view. Three elements are identified exchanging heat, (i) the local heat producers that produce heat using fossil energies or biomass that is burned to boil water (ii) the heat grid that transfers the hot water to (iii) consumers. Once consumers have absorbed the heat, the temperate water is sent back to the heat grid that transfers it to the producers.

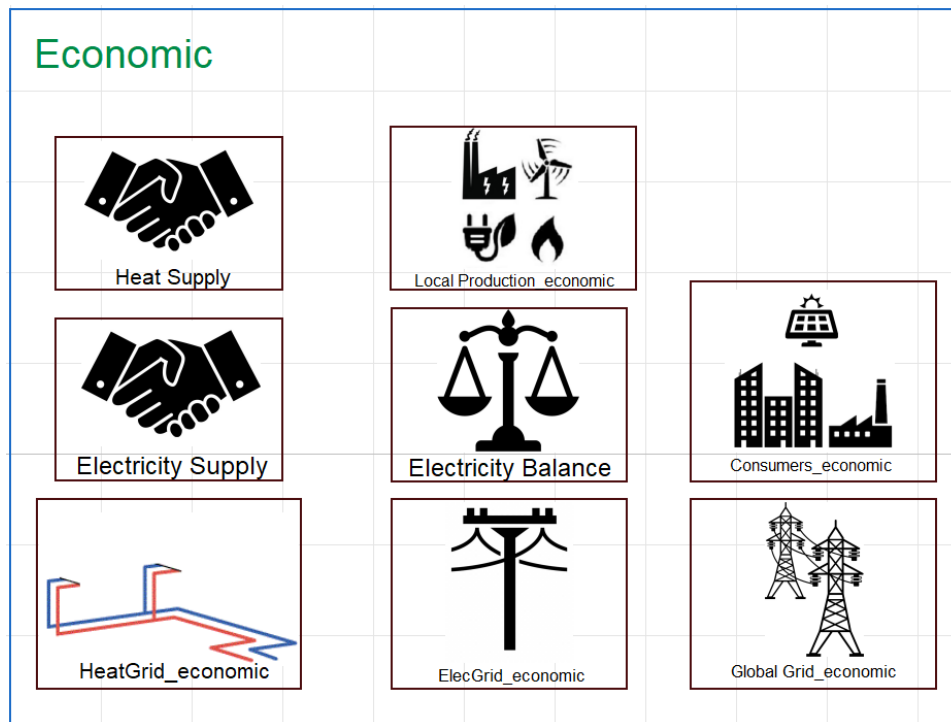


Figure 4-5: PowerGrid economic architecture

The economic point of view shown in Figure 4-5 is different from the two others above. Indeed, this architecture description does not aim at capturing the financial interactions be-

tween its components as there is no financial interaction between systems (we consider that financial interactions are only between stakeholders), however, this description aims to capture the static economical data related to each PowerGrid sub-system. This architecture will be the foundation of economic analyses to assess the viability of potential investments that will be made in the PowerGrid by *sponsors*. In Figure 4-5 can be found the economic components related to the PowerGrid sub-systems mentioned earlier such as the local production units, the consumers, the electric distribution grid, and the national grid (also called transmission grid). In addition to them, services such as the electricity and heat supply, and the electricity balance were included into this architecture description. They have a significant weight on the economical viability of the PowerGrid renovation. Furthermore, adding these services in the architecture description will allow assigning them to the future stakeholders that will join the project in more advanced design phases.

4.2.7 Behavioral level

In the urban planning phase of the PowerGrid, the behavioral modeling was performed by EIFER (European Institute for Energy Research) using AnyLogic [7] toolbox and the "EnergyLogic" library dedicated to energy systems, which was developed by this same entity. This library provides means to make rapid prototyping for the assessment of multiple energy concepts. It uses an agent-based modeling approach to represent the interactions between complex systems' components. In the framework of the PowerGrid, the agents represent the different energy components of the district, notably the production units, the distribution grids (heat and electricity), the consumers, etc. that interact by exchanging production and demand data on heat and electricity [60].

An overview of the PowerGrid model using the EnergyLogic library is shown in Figure 4-6.

It mainly includes three elements:

1. Demand computation which is based on empirical consumption profiles that are specific to each building of the PowerGrid. These profiles differentiate between the heating demand that can either be obtained from the heat network or by electric heating depending on whether or not the building is connected to the district heat network, the domestic hot water that can also be obtained by either way and the electricity needed for specific uses for each building.
2. Energy supply computation that can be separated into two elements:
 - Heat supply: ensured by means of a gas-fired Combined Heat and Power (CHP) and four back-up gas-fired boilers.

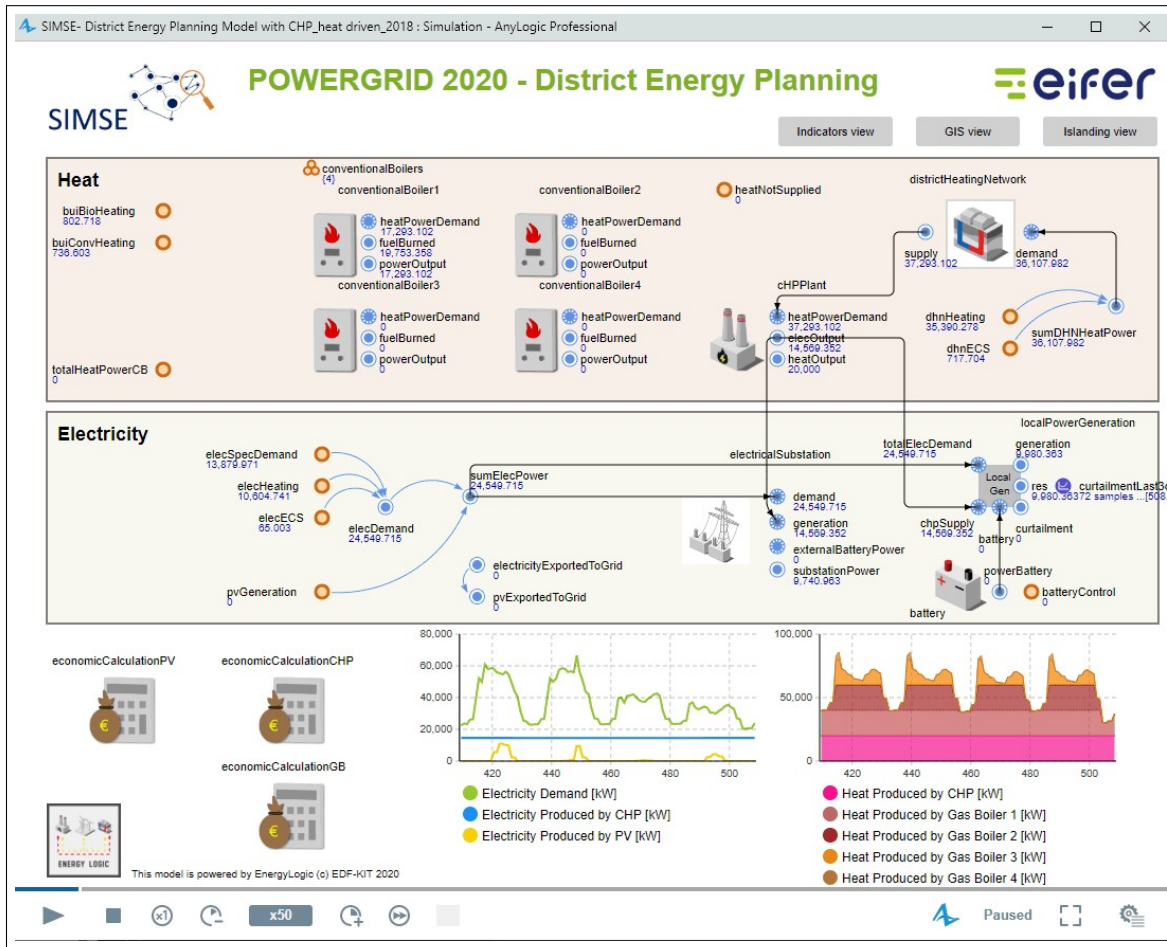


Figure 4-6: Main interface of the EnergyLogic PowerGrid Demonstrator Tool (from [74])

- Electricity supply: ensured by means of (i) the local PV panels that equip a set of pre-defined buildings and which have the priority in the electric network, (ii) the CHP that produces electricity (regulated to give priority to the heat supply), and (iii) the national grid that provides all the residual electricity needs.

3. Economical computation which considers:

- Fixed and variable investments regarding the energy supply units (CAPEX) for CAPital EXpenditure and the operational costs (OPEX) for OPERational EXpenditure.
- The revenue from selling electricity and heat.
- The costs of buying the residual electricity from the national grid.

The modeling components in EnergyLogic are represented as ideal systems that provide the energy demand required by producers without taking into consideration the physical

phenomena of the energy production and distribution. Nevertheless, the grid losses in terms of heat and electricity were taken into account using static factors.

4.2.8 Verification model

The simulations that were made by EIFER at the urban energy planning phase intend to evaluate the *town council's* KPIs and compare them to their targets. A user-friendly panel was developed to give a clear perspective of these KPIs.

At this PowerGrid phase, no requirements verification was made. In this thesis, and as mentioned earlier, we will only consider a restricted perimeter of 20 buildings from the PowerGrid district. The *town council* KPIs concerning the ratio of renewables and self-sufficiency were assessed for that perimeter under the three scenarios 0%, 50%, and 100% of the installation of PV panels for a set of selected buildings. Figure 4-7 shows the KPI panels for *kpi1* and *kpi4*, and the three scenarios.

The two other KPIs concerning the *neighbourPower* competitiveness and CO2 emissions are only representative when making a global analysis on the PowerGrid including the entire zone with 719 buildings. Indeed, the investments in production units and distribution grids can only be cost-effective at a district level, similarly to the CO2 emissions that are mainly induced by the gas-fired CHP and back-up boilers, which is representative only at a district level. Therefore, these two KPIs were evaluated for the 719 buildings of the PowerGrid. These KPIs will not be addressed in the rest of our analysis. However, to show an example of their evaluation, Figure 4-8 depicts the assessment results for the scenario of 50% of buildings equipped with solar panel.

4.2.9 Results

The rough analysis carried out in the first phase of the PowerGrid essentially allowed the town council to identify the main levers to achieve its objectives, notably the impact of the PV panels penetration and buildings' renovation (not detailed in this thesis). The first results concerning the KPIs assessment does obviously not reach the targets set by the *town council*. The objective of the next phases will be to refine this analysis by integrating the stakeholders responsible for the different elements of the PowerGrid in order to have a finer estimation and a sharper resolution in regards to the energy consumption, production as well as the grid losses due to energy distribution.

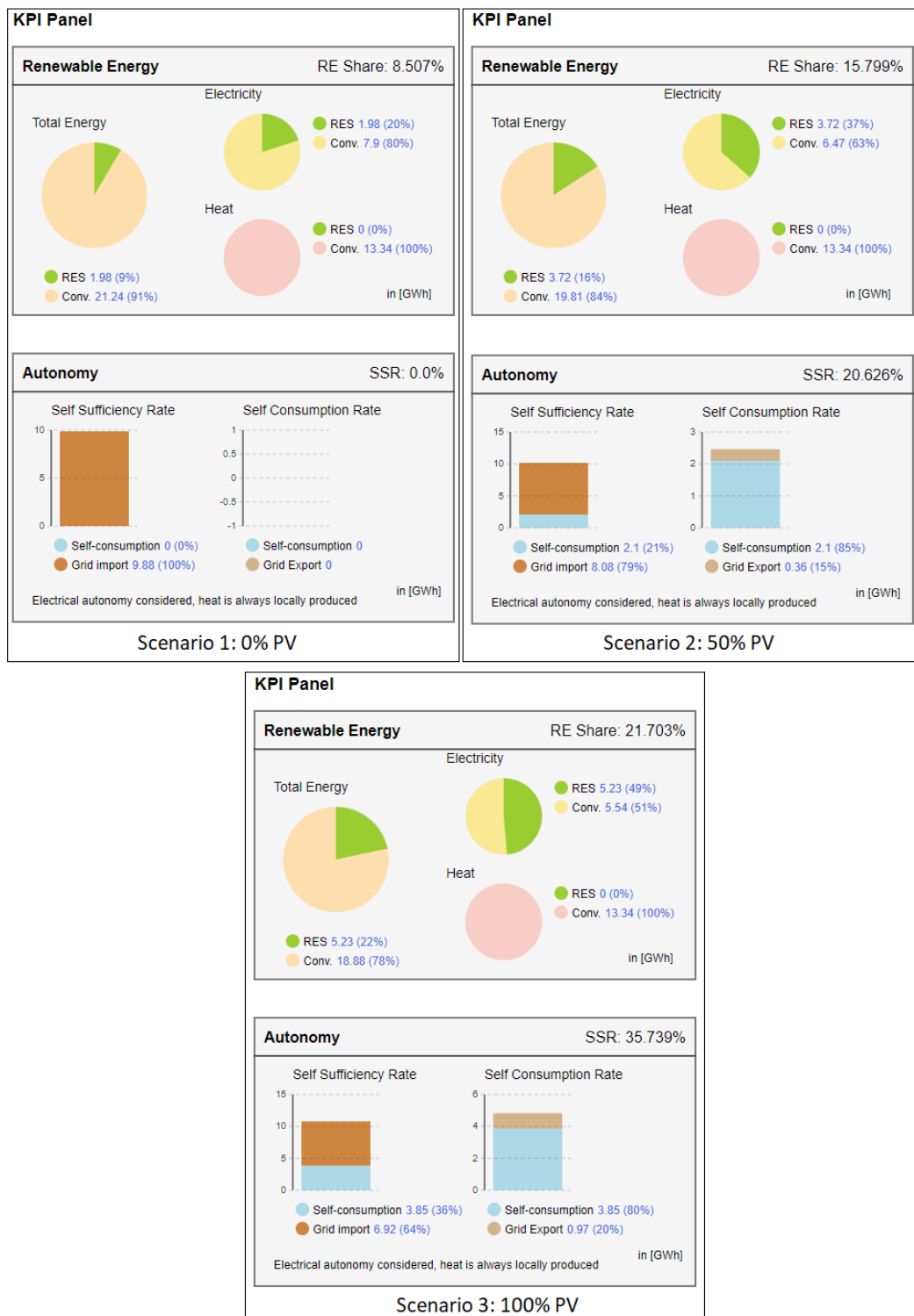


Figure 4-7: Evaluation of the town council's kpi1 and kpi4 (made by EIFER)

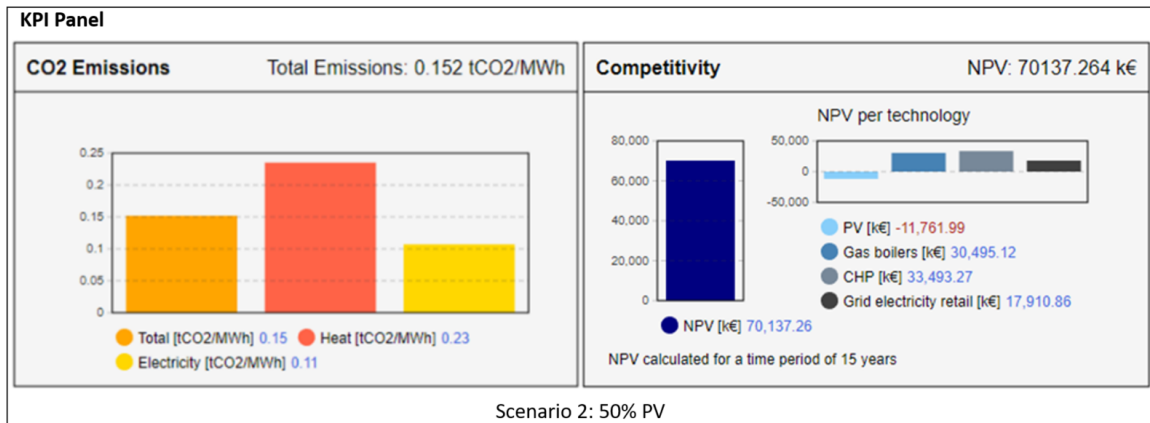


Figure 4-8: Evaluation of the town council’s kpi2 and kpi3 (made by EIFER)

4.2.10 Summary

In this preliminary phase of the PowerGrid, the notions of stakeholder interaction, dependencies, and contracts between stakeholders are missing as only one stakeholder is considered, which did not allow to show the full potential of CReMA methodology. Nevertheless, the relevance of applying our methodology at the urban energy planning phase remains firstly in the capture of the *town council’s* high-level goals that were decomposed into sub-goals that are more likely to be specified using formal properties, secondly, the identification and characterizing of the legacy energy system that is already operating in the PowerGrid zone, and which will be the starting point of the renovations. This was done in the architecture description level which was an opportunity to ask and explore multiple questions concerning the operation of the local energy system. The establishment of this level was also an opportunity to seek data concerning the PowerGrid existing components such as the local production units, the consumption structures, the heat, and electric grids, including their physical and economical aspects.

The town council KPIs are embodied in the next chapter throughout the emergence of the new supplier “NeighbourPower” that will have more roles than the classical suppliers, and will therefore propose to clients new contracts and services to make the PowerGrid district greener and consumes local energy while being competitive. As the first evaluation results of the town council KPIs were relatively far from their targets, the ratio of renewables and the self-sufficiency rate were downsized to 25% and 50% respectively instead for 30% and 60%.

4.3 Phase 1: PowerGrid dynamic sizing

4.3.1 Stakeholders, objectives and tested scenarios

The goal of the PowerGrid dynamic sizing phase is to design the different parts of the district while including the different stakeholders that will be in charge of their design and operation. The idea is to follow CReMA methodology and elaborate contracts between stakeholders to give each one of them the possibility to design its scope independently. Negotiations will take place throughout the design process in order to revisit the contracts' requirements to fit on the one hand with stakeholders' needs and on the other hand with what they can guarantee under specific assumptions.

In this phase, we have expanded the scope of stakeholders involved in the PowerGrid to include the following ones:

- Consumers (Buildings from different types).
- Electric Distribution System Operator (E-DSO): in charge of the electric distribution grid that transfers low and medium voltage electricity.
- Heat Distribution System Operator (H-DSO): in charge of the heat distribution grid that transfers heat through hot water.
- Local Producers (electricity and heat): in charge of the local production of electricity and heat in the PowerGrid. Historically, a CHP and gas-fired boilers already exist in the PowerGrid district and feed nearly half of its buildings in heat.
- Transmission System Operator (TSO): in charge of the electric national transmission grid that transfers high voltage electricity. The TSO is also considered in the PowerGrid as an interface with the national grid and non-local electricity producers.
- Suppliers: in charge of commercializing the electricity and heat.
- Balance Manager: in charge of ensuring the electrical balance management service between the demand and offer.

As mentioned earlier, the town council has translated his high-level goals by envisioning the local supplier called NeighbourPower for which multiple roles will be attributed for a more efficient management of the new PowerGrid Energy system. The NeighbourPower is thus considered to have simultaneously the following roles:

- Electricity supplier
- Heat supplier (the unique in the PowerGrid district)

- H-DSO
- Balance manager.

4.3.2 Stakeholder intentional level - Strategic rationale models

In this chapter, we consider that each stakeholder defines his high-level goals and also the tasks that are assigned to him as part of his historic missions in the PowerGrid district. This does not mean that these tasks will remain still. In fact, they represent the starting point that takes into account the legacy system and stakeholders' roles, which can widely change during negotiations. Therefore, we start by defining a Strategic Rationale (SR) model for each PowerGrid stakeholder :

4.3.2.1 Consumers

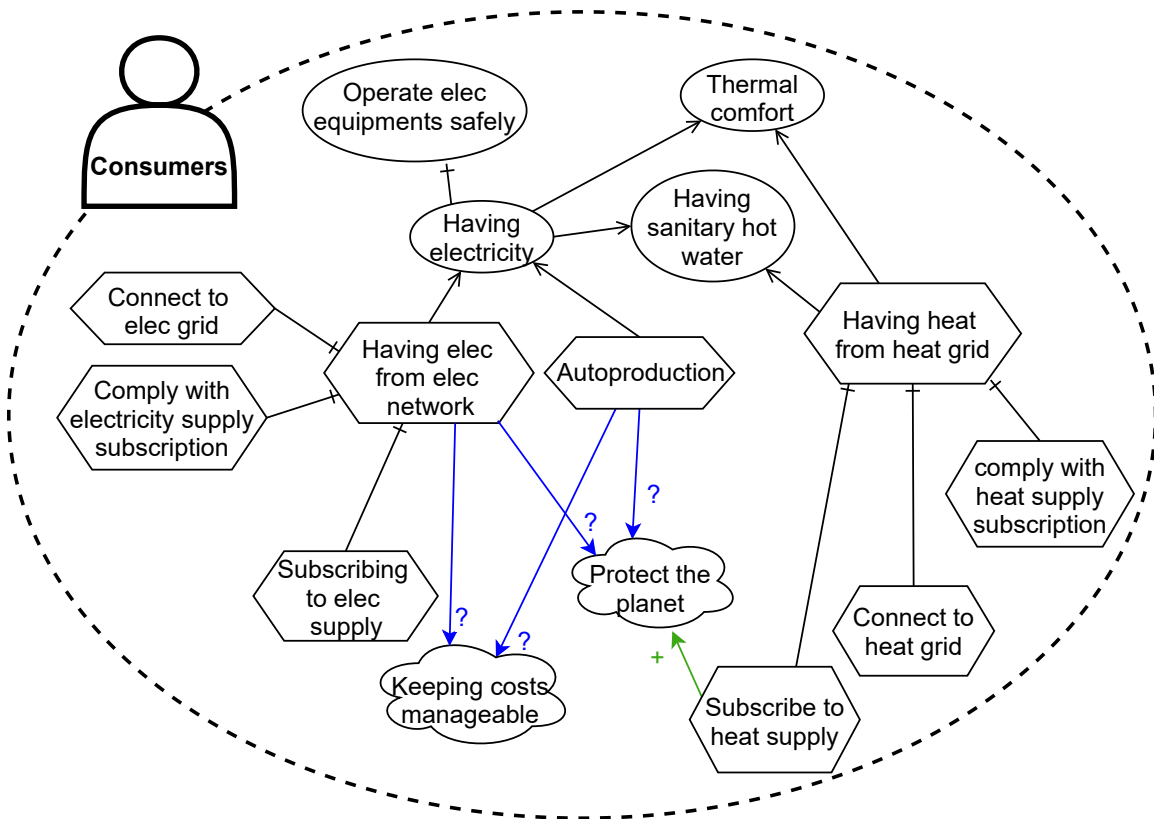


Figure 4-9: Consumers Strategic Rationale Model

Figure 4-9 introduces the SR model of consumers. The first main objective of a consumer regarding the PowerGrid is having electricity with safety and quality. The second main objective of a consumer is having thermal comfort at home or at the office depending on

the type of buildings. In addition to that, the consumers have two soft-goals (cloud-shaped) that are not fully specified at this phase of the system lifecycle due to the lack of knowledge concerning the targets that can be reached at a reasonable effort. The first soft-goal is keeping the costs manageable and the second one concerns the protection of the planet.

The consumers' goals and soft-goals are then decomposed into sub-goals and sub-tasks in order to identify their means-ends. Consumers' thermal comfort can be satisfied by either having electricity or having heat from the heat grid. Having electricity can also be satisfied by either doing self-production or by subscribing to an electric supply contract. These two possibilities have obviously different impacts on the consumers' soft-goals and will both have an important weight for deciding which alternative to be chosen. Detailed analysis can be done in order to evaluate which possibility answers the intentions of the consumers. In Figure 4-9, the blue arrows refer to the unknown impact of a task on the soft-goals. Indeed, only a sharp analysis can give a clear vision on whether or not self-production has a better impact on the planet compared to subscribing to a supplier contract. Similarly for the impact on "Keeping the costs manageable" soft-goal.

4.3.2.2 Producers

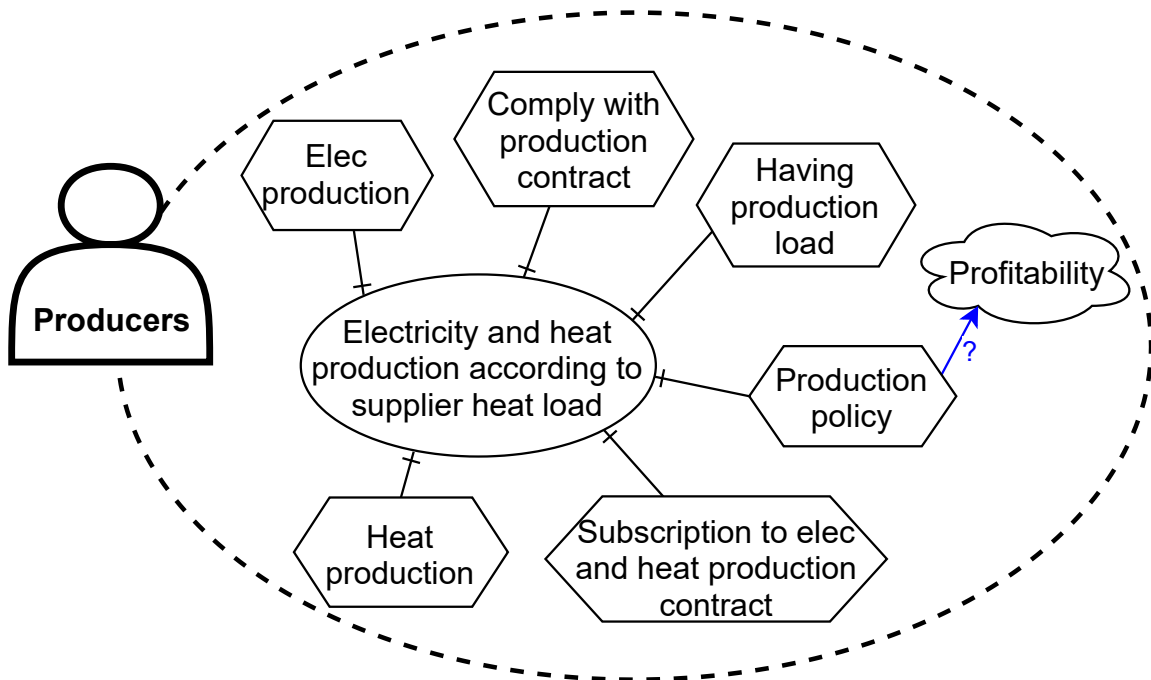


Figure 4-10: Producers Strategic Rationale Model

Two kinds of producers are considered in the PowerGrid : (i) National producers who only produce electricity and inject it in the national grid, (ii) Local producers who produce

electricity and heat according to the energy demand sent by the supplier while being profitable. Here in the PowerGrid demonstrator, the local energy production is driven by the heat demand because the local producers operate a gas-fired CHP that is itself run by the required heat load and produces electricity as a secondary “bonus” energy that increases the overall efficiency and profitability.

To achieve their main intentions and inject the produced electricity and heat respectively into the electrical grid and the district heating network, the producers need to subscribe to production contracts and comply with their terms.

Furthermore, the producers have a production policy that specifies their internal production and billing strategies. Different policies can be explored and tested in order to choose the most profitable one.

4.3.2.3 Electric Distribution System Operator (E-DSO)

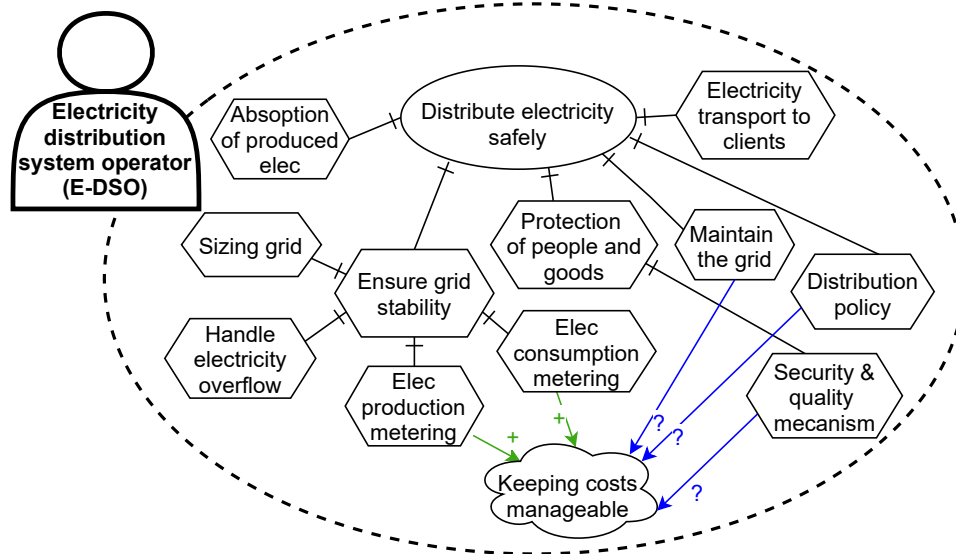


Figure 4-11: E-DSO Strategic Rationale Model

Figure 4-11 shows the intentional level of the E-DSO. His main goal is closely related to his historic missions in the district, which is to distribute electricity safely to the local consumers. To this end, the E-DSO should perform the following tasks:

- Absorb the electricity produced by the local producers and the one imported from the national grid, then transport it to the consumers.
- Maintain the grid and protect people and their goods by ensuring security and quality mechanisms.

- Ensure the grid stability by (i) correctly sizing the grid, (ii) handling the electricity overflows, (iii) and making accurate metering of the electricity production and consumption.
- Define the distribution policy that specifies the billing strategy for suppliers (suppliers are in charge of the energy production and consumption subscriptions that are set with producers and consumers)

These tasks have different impacts on the second main priority of the E-DSO which is the soft-goal “Keeping the costs manageable”. The green arrows in Figure 4-11 refer to the positive impact of the accurate metering on the correct billing of the distribution services to the suppliers. In contrast, the blue arrows refer to a poor understanding of what is the real impact of the tasks on the financial aspects at a specific time T. Further analysis needs to be done to define the right distribution policy, maintenance strategy, and which security systems to invest in to have the optimal satisfaction of the E-DSO priorities.

4.3.2.4 NeighbourPower as the Heat Distribution System Operator (H-DSO)

Figure 4-12 shows the intentional level of the H-DSO. His main goal is to provide heat distribution to the PowerGrid consumers while keeping his costs manageable, which is his second main priority. To this end, the heat distribution system should perform the following tasks:

- Ensure an efficient heat interface with producers by absorbing the produced heat and returning water at a temperature below a specific value in order to guarantee an efficient thermodynamic cycle for heat production.
- Transport heat to clients.
- Maintain the heat grid and ensure hygiene conditions specified by regulations.
- Metering of the heat production and consumption

The distributor has distribution conditions under which he bills the supplier for the services ensured to consumers.

4.3.2.5 NeighbourPower as Electricity and heat supplier

Figure 4-13 shows the intentional level of NeighbourPower as a supplier. His main goal is to ensure the electricity and heat supply service for the PowerGrid while being profitable.

For the sake of achieving his profitability soft-goal, the NeighbourPower needs to have a considerable market share. To do so, the supplier must be competitive and have a good

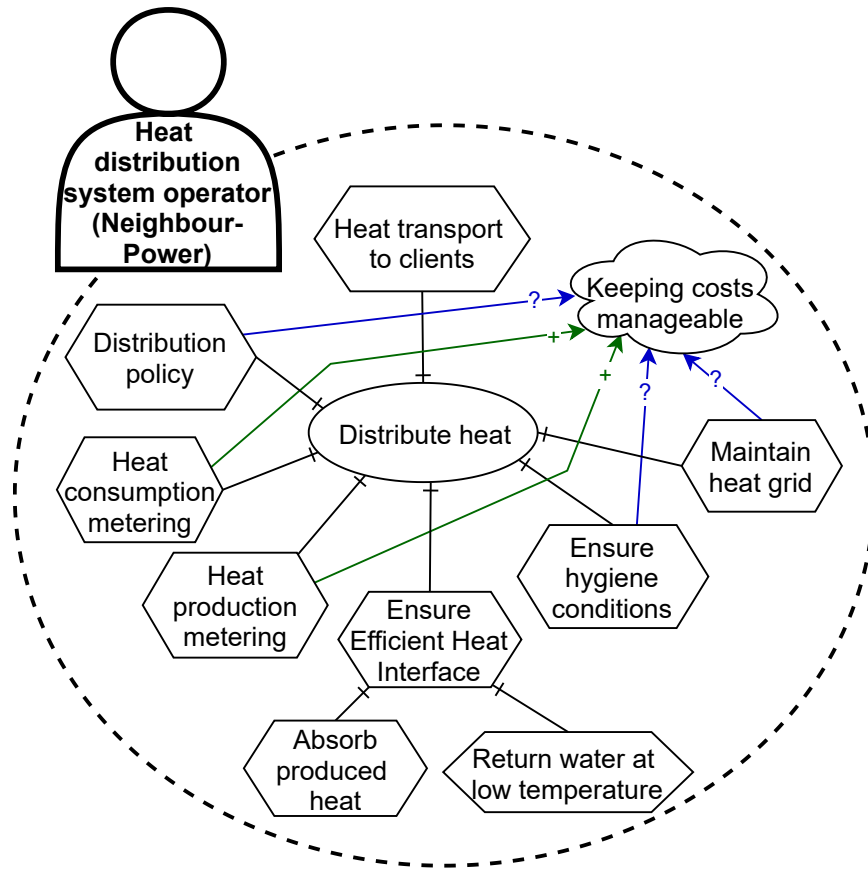


Figure 4-12: Strategic Rationale Model of NeighbourPower as the H-DSO

supplier image to maximize the number of clients. Two options can be considered by the supplier, he can either subscribe to a green production contract (with renewable energies) or a non-green electric production (non-renewables). The first renewable option has a positive impact on the image of the supplier but a negative impact on the competitiveness as the green electricity costs more. This is global observation for this district in particular, which is urban - therefore with limited green energy production possibilities - and "old" - hence with an existing legacy which would be costly to completely renovate. The second non-renewable option has the opposite impact on both the competitiveness and the supplier image soft-goals.

A part of the supply mission is to predict the electricity and consumption in order to send to the producers the production load to perform. The predictions are based on the electricity and heat consumption history as well as an analysis concerning the consumption behavior changes. The history of consumption is also used in order to bill consumers referring to his sell policy and to pay the distributor on the basis of the electricity transported.

We emphasize that competitor suppliers have a similar strategic rational model but are limited to the electricity supply. NeighbourPower is considered to be the only heat supplier in the PowerGrid zone. In the next chapters, and for the sake of conciseness, none of the

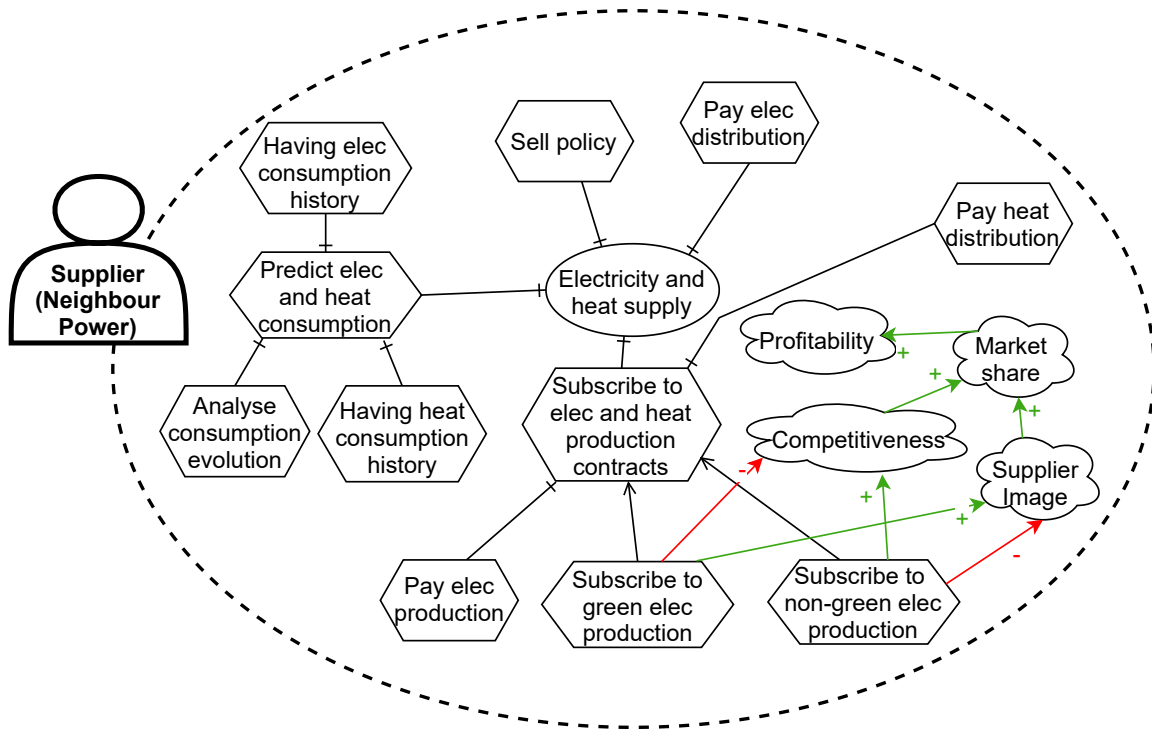


Figure 4-13: Strategic Rationale Model of NeighbourPower as a supplier

competing electricity suppliers will be considered.

4.3.2.6 NeighbourPower as Balance manager

Figure 4-14 shows the intentional level of NeighbourPower as a balance manager. His main goal here is to ensure the balance management mission for the PowerGrid's electric network while being profitable.

The balance management of the electric network is here considered off-line and purely financial: no physics is involved. Based on the difference between the metering of what each supplier has provided (via the metered production of its producers) and the metering of what its clients have consumed, the balance manager determines the monthly compensation to be received by the E-DSO (since the E-DSO may have to ensure adequate supply to each consumer through calls to the TSO). In addition, the balance manager determines the monthly fine or reward to be paid by or to each supplier, with conditions specified as part of the balance management policy. The latter also includes specifications concerning the fees for this balance management service.

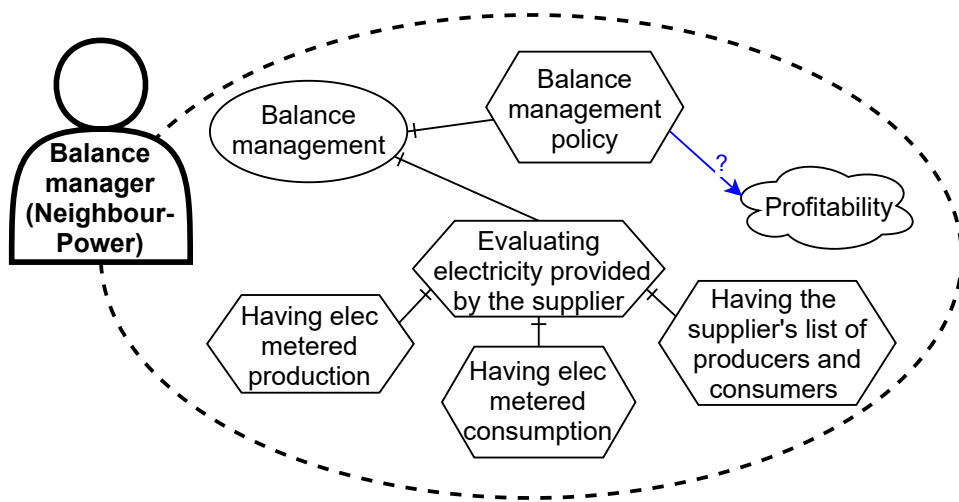


Figure 4-14: Strategic Rationale Model of NeighbourPower as a balance manager

4.3.2.7 Transmission System Operator (TSO)

Figure 4-15 shows the intentional level of the TSO. His main mission is to provide electricity transmission for the high voltage electricity while keeping his costs manageable. This mission is decomposed into ensuring the high voltage grid stability and to assist the distribution grid for its stability. The TSO has a transmission policy for the service offered to the DSO and also for the payment of the overproduction of the PowerGrid.

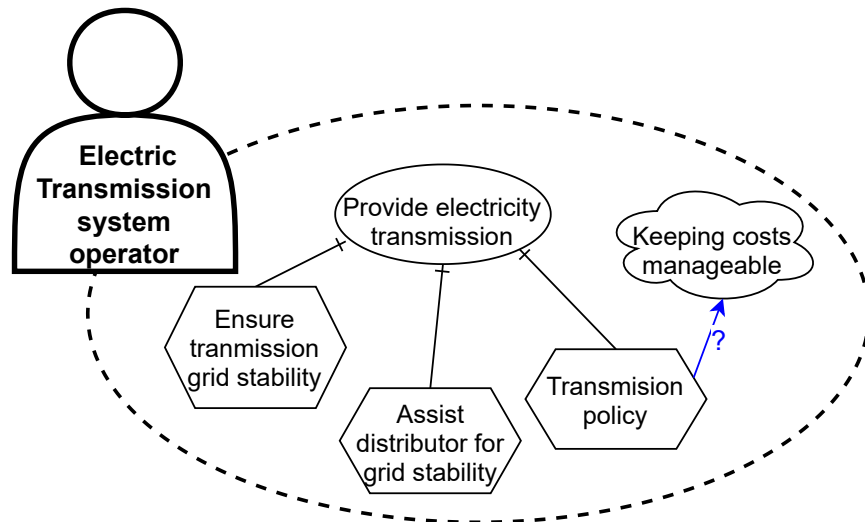


Figure 4-15: TSO Strategic Rationale Model

4.3.3 Stakeholder intentional level - Strategic dependency models

The second step of the intentional level consists of representing the dependencies and relationships between every two stakeholders using the dependency model.

4.3.3.1 Dependency: Consumers – H-DSO (here NeighbourPower)

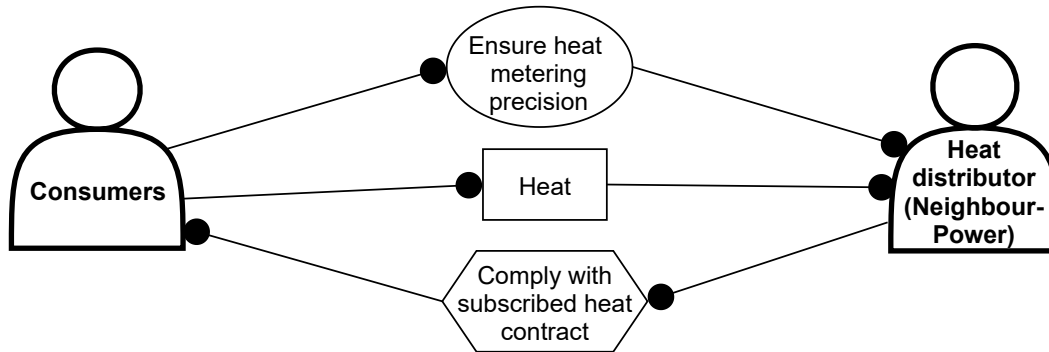


Figure 4-16: Consumers - H-DSO dependency model

Figure 4-16 depicts the SD model between the consumers and the H-DSO (here NeighbourPower). Firstly, the consumers depend on the heat distributor to have the heat transported to their place. Secondly, the distributor depends on the consumers to comply with the subscribed heat contract. Indeed, the correct operation of the heat grid requires that consumers return the heated water to the grid with specific characteristics that are specified in the contract. Finally, the consumers depend on the distributor for the precision of the metering of their consumption.

4.3.3.2 Dependency: Consumers – Supplier (here NeighbourPower)

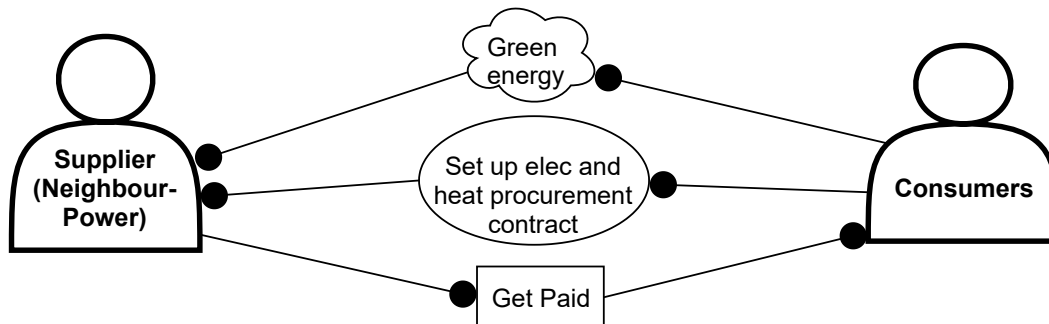


Figure 4-17: Consumers – Supplier dependency model

Figure 4-17 depicts the SD model between the consumers and the supplier (here NeighbourPower) for electricity and heat. Firstly, the consumers depend on the supplier to either

have their heat or electricity procurement contract or both of them. Secondly, the consumers depend on the supplier to have the green energy soft-goal with no specific targets to be obtained at this level of the development cycle. Finally, the supplier depends on the consumers to pay their dues in regards to the supply service as well as electricity production and distribution.

4.3.3.3 Dependency: Producers – H-DSO (here NeighbourPower)

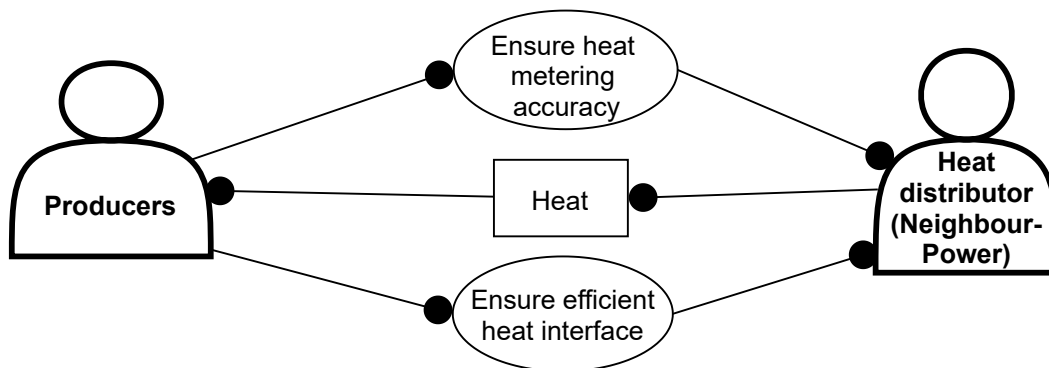


Figure 4-18: Producers - H-DSO dependency model

Figure 4-18 depicts the SD model between the producers and the H-DSO. On the one hand, the heat distributor depends on local producers to produce the heat that will be injected into the heat network while complying with the production contract. On the other hand, the producer depends on the H-DSO to ensure an efficient heat interface by absorbing the produced heat and returning water at a temperature below a specific value in order to guarantee an efficient thermodynamic cycle for heat production. The producers also depend on the H-DSO to have accurate metering of the heat produced and fed-in the grid.

4.3.3.4 Dependency: (heat and electricity) Producers – (heat and electricity) Supplier (here NeighbourPower)

Figure 4-19 depicts the SD model between producers and the supplier (here NeighbourPower). Firstly, the producers depend on the supplier to set up their electricity and heat production contracts. Secondly, they rely on him to have the production load data in terms of heat or electricity according to the production unit. On the other hand, the supplier relies on local producers to ensure the production of the heat and electricity, and also to have green energy. Finally, the producers depend on the supplier to get paid.

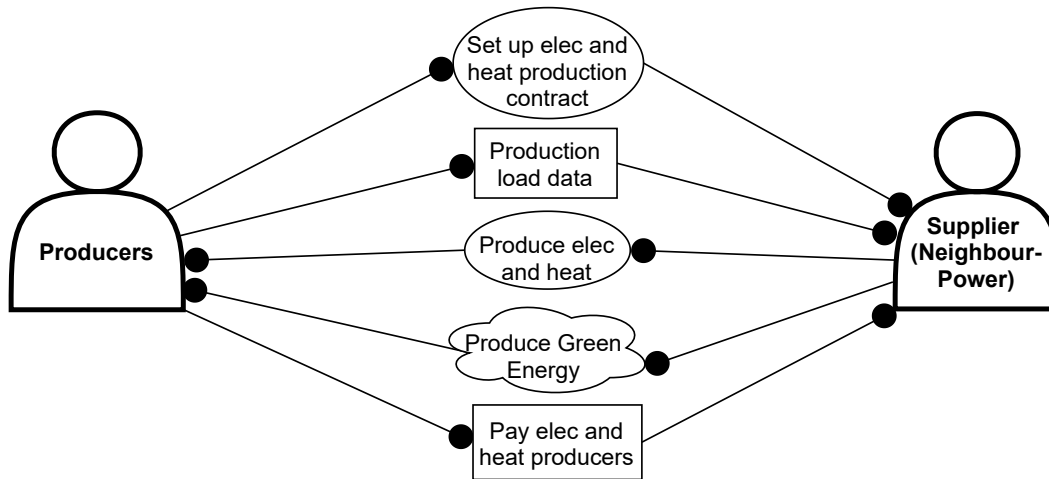


Figure 4-19: Producers - Supplier dependency model

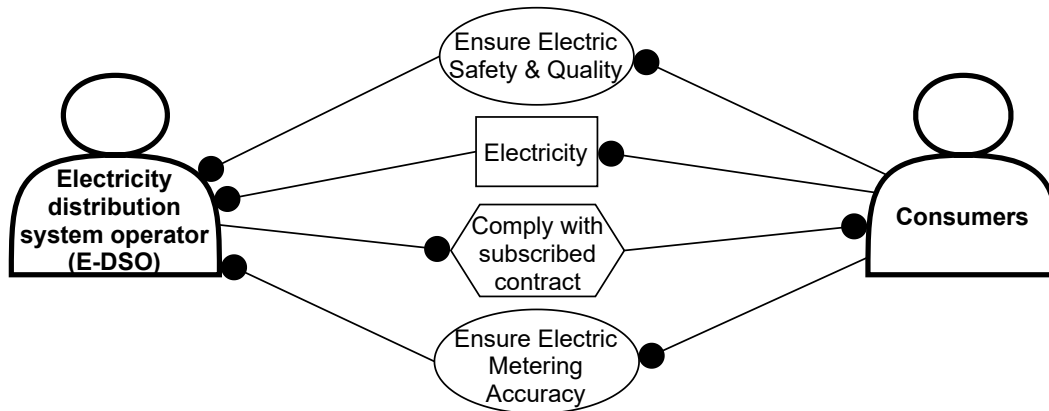


Figure 4-20: E-DSO - Consumers dependency model

4.3.3.5 Dependency: E-DSO – Consumers

Figure 4-20 depicts the SD model between the E-DSO and the consumers. On the one side, the consumers depend on the distributor to have the electricity transported to their place with safety and quality requirements. The consumers also depend on the distributor for the precision of the metering of their consumption. On the other side, the distributor depends on the consumers to comply with the subscribed contract with the supplier.

4.3.3.6 Dependency: Producers – E-DSO

Figure 4-21 depicts the SD model between the producers and the E-DSO. On the one side, the electric distributor depends on local producers to produce electricity and to comply with the production contract terms. On the other side, the producers depend on the distributor to ensure the electric network balance. The producers also depend on the distributor to have

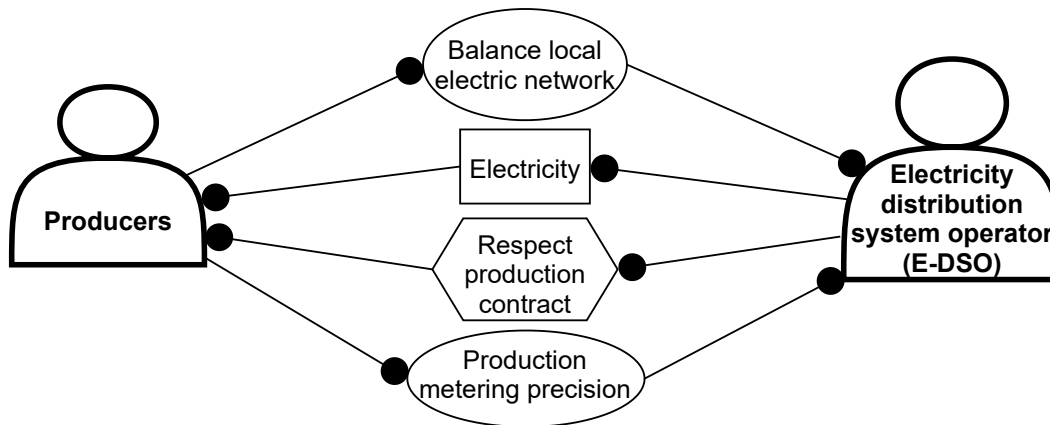


Figure 4-21: Producers - E-DSO dependency model

precise metering of the electricity produced and fed into the grid.

4.3.3.7 Dependency: E-DSO – Supplier (here NeighbourPower)

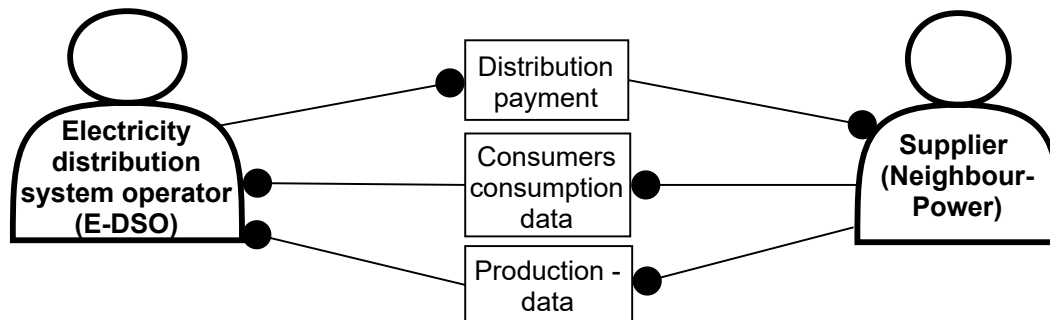


Figure 4-22: E-DSO - Supplier dependency model

Figure 4-22 depicts the SD model between the E-DSO and the supplier (here NeighbourPower). On the one hand, the supplier depends on the E-DSO to send him the data about the electricity consumption of clients as well as the production data. On the other hand, the E-DSO depends on the supplier for the payment of the electricity distribution service.

4.3.3.8 Dependency: TSO – E-DSO

Figure 4-23 depicts the SD model between the TSO and the E-DSO. The DSO uses the TSO network as an unlimited and instantaneous source of electric power when local suppliers do not match the consumption of their clients and when local producers cannot compensate for the imbalance. Thus, the DSO depends on the TSO for the electricity back-up task in return for payment.

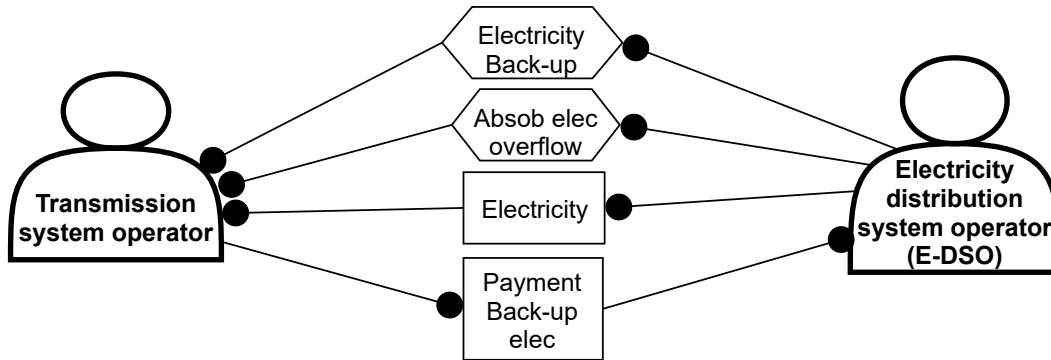


Figure 4-23: TSO - E-DSO dependency model

The DSO also depends on the TSO to absorb the overflow of the electricity produced locally and not consumed by PowerGrid consumers. The payment of the overflow injected in the global grid goes through an intermediary, which is the balance manager. Assisting the DSO for the stability of its grid is a legal obligation as part of the TSO role.

The dependency “Electricity” describes the physical resource exchanged between the two stakeholders. This resource can be transferred in either direction depending on the state of the distribution grid as introduced earlier. However, in terms of dependency, it is the DSO who depends on the TSO electricity to regulate its state and guarantee the stability of the local grid.

4.3.3.9 Dependency: Balance manager (here NeighbourPower) – E-DSO

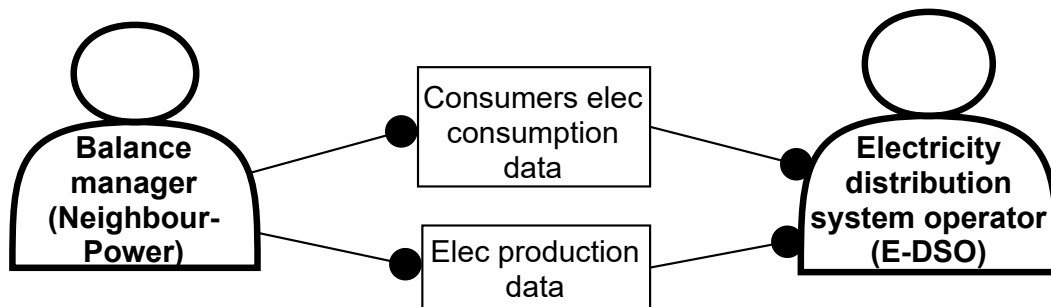


Figure 4-24: Balance manager - E-DSO dependency model

Figure 4-24 depicts the SD model between the balance manager and the E-DSO. In order to ensure the balance billing, the balance manager depends on the DSO to provide him with the consumption power metering data, as well as the power production metering of all local producers. Providing this data is a legal obligation as part of the DSO missions.

4.3.3.10 Dependency: Balance manager (here NeighbourPower) – TSO

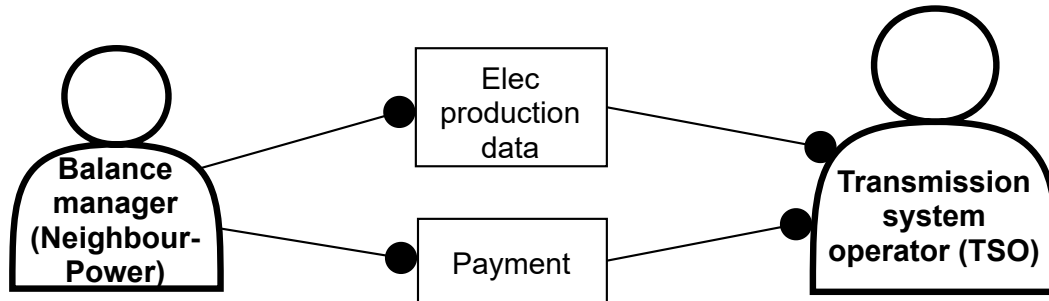


Figure 4-25: Balance manager - TSO dependency model

Figure 4-25 depicts the SD model between the balance manager and the TSO. The balance manager depends on the TSO to provide him with all active power metering for non-local production as a legal obligation. Based on that, the balance billing will be made taking account of power transfers between the local and the global networks. If the bill is positive, the TSO must send a payment, meaning that on average, the electricity flow went from the local grid to the global one. Contrariwise, if the bill is negative, the TSO receives a reward.

4.3.3.11 Dependency: Balance manager (here NeighbourPower) – Supplier (here NeighbourPower)

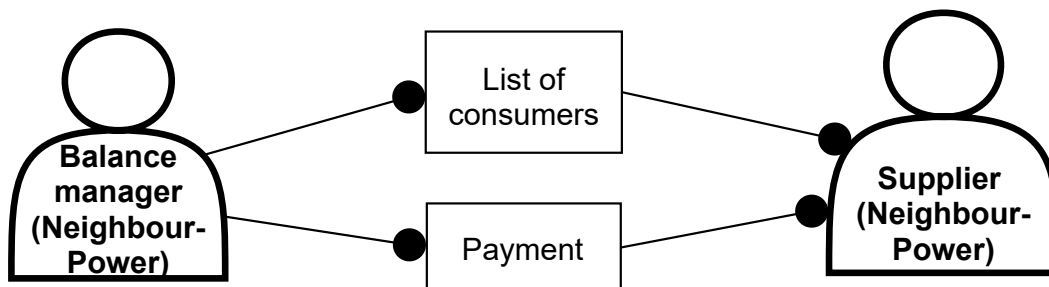


Figure 4-26: Balance manager - supplier dependency model

Figure 4-26 depicts the SD model between the balance manager and the supplier. Here both roles are attributed to NeighbourPower. The balance manager depends on the supplier to send him the list of his consumers as part of the energy market rules. The balance manager also depends on the supplier to pay the penalties if the electricity consumption of its clients does not match with his production contracts. In contrast, the payment can be in the other way if the supplier has compensated for the lack of electricity for a competing supplier.

4.3.3.12 Dependency: Supplier (here NeighbourPower) - H-DSO (here NeighbourPower)

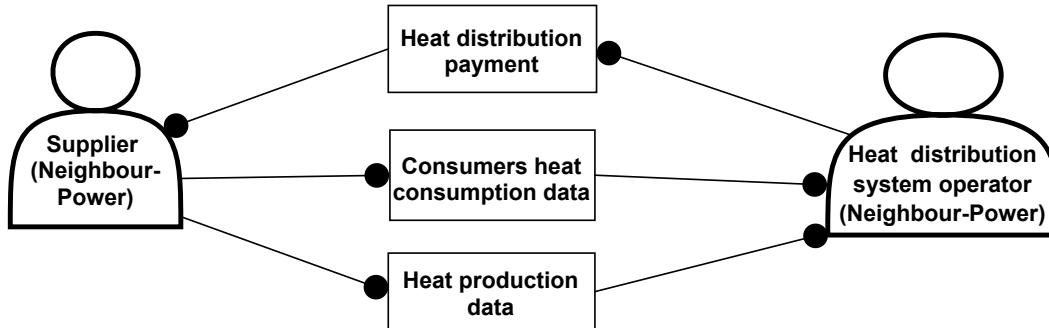


Figure 4-27: Supplier - H-DSO dependency model

Figure 4-27 depicts the SD model between the supplier and the H-DSO. Here both roles are attributed to NeighbourPower. On the one side, the supplier depends on the H-DSO to send him the data about the heat consumption of clients as well as the heat production data. On the other hand, the H-DSO depends on the supplier for the payment of the distribution service.

4.3.3.13 Global strategic dependency model

Figure 4-28 shows the global strategic dependency model of the PowerGrid gathering all the stakeholders involved in phase 1 of the PowerGrid renovation: its dynamic sizing.

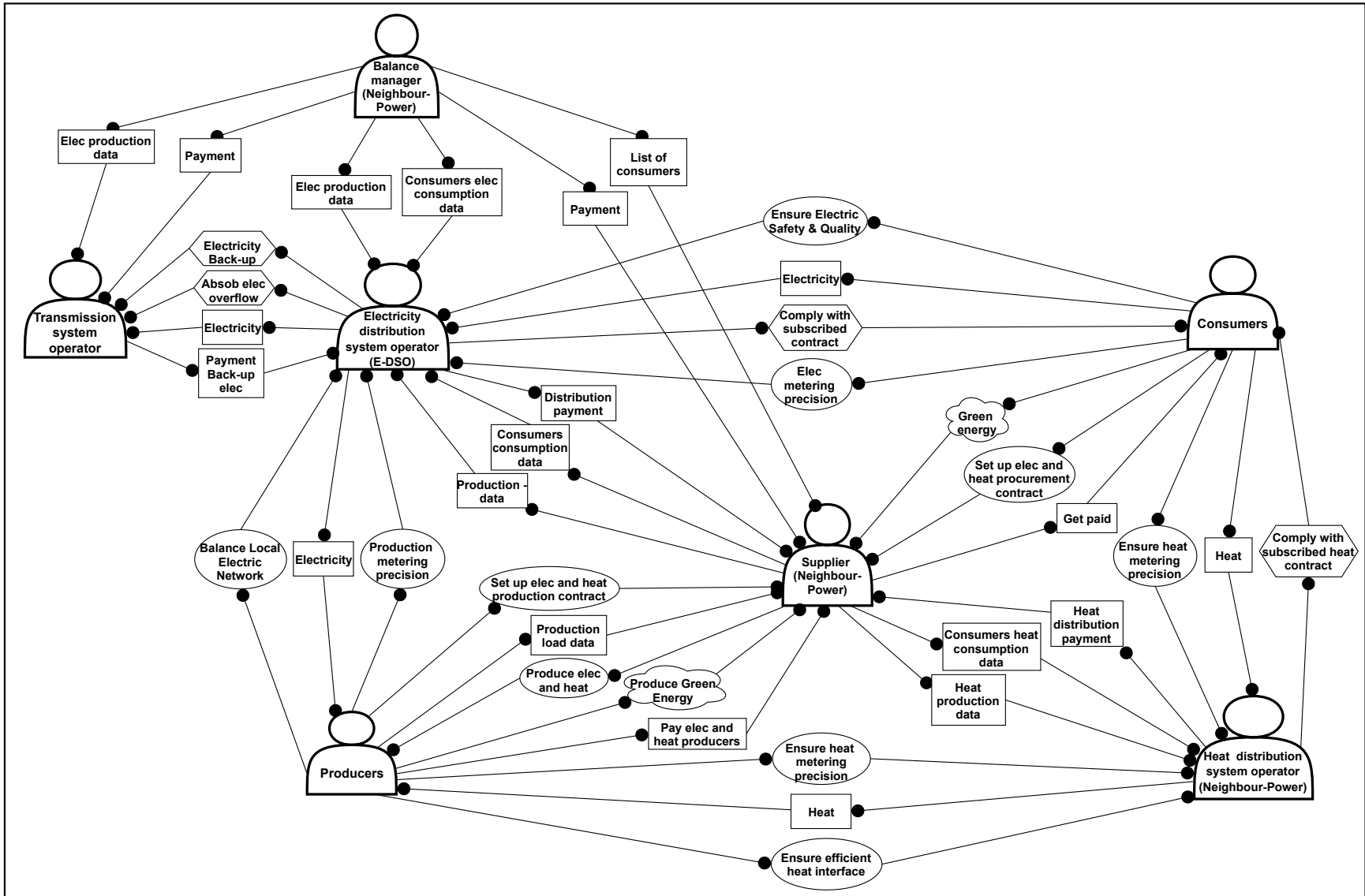


Figure 4-28: Global SD model

4.3.4 Stakeholder interaction level

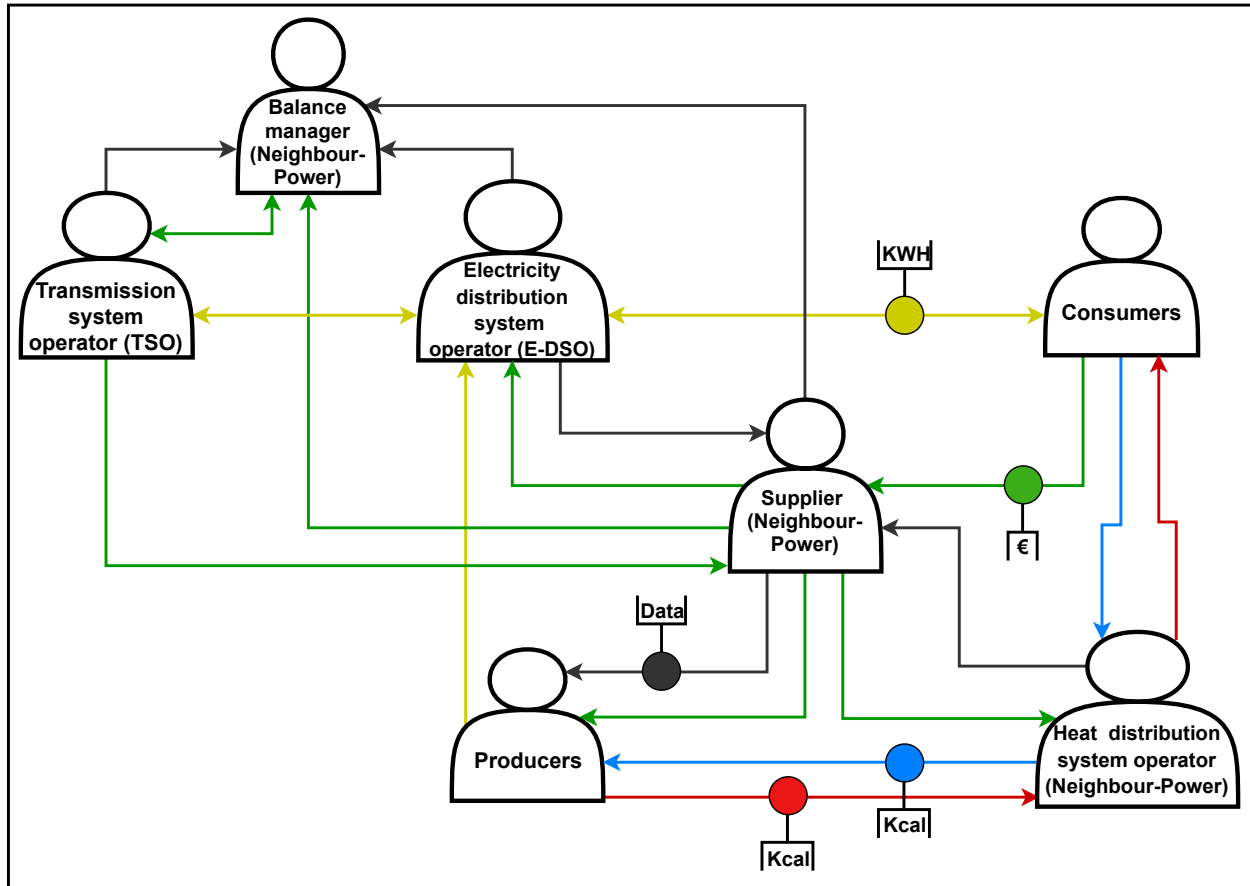


Figure 4-29: Stakeholders global interaction model

Figure 4-27 depicts the interactions between the PowerGrid stakeholders that are involved in phase 1. If we match this figure with the global SD model in Figure 4-28, we will notice that the interactions between the stakeholders reflect the resource dependencies. To give some examples of interactions, we present the following ones:

- Financial interaction between the supplier and consumers
 - What are the exchanged elements: Money.
 - How is the exchange performed: a bank automatic transactions is made at the end of every month corresponding to the electricity consumption and the subscription fees.
 - Vector of exchange: the banking system.
 - Interface: Human-computer interface or consumer to supplier representative.

- Data interaction: between the supplier and consumers.
 - What are the exchanged elements: the electric and heat monthly electronic bill.
 - How is the exchange performed: At the end of each month, a bill is automatically edited based on the consumer subscription and consumption, and then sent to him.
 - Vector of exchange: the internet.
 - Interface: electronic mail or costumer area in the supplier’s web platform.
- Electrical interaction between clients and the electric network.
 - What are the exchanged elements: electricity.
 - How is the exchange performed: No exchange protocol is specified. The consumers use electricity at random instants of time and the electric network adapts accordingly.
 - Vector of exchange: electric cables.
 - Interface: Electricity connection point including a cut-off device.
- Heat interaction between producers and heat network.
 - What are the exchanged elements: Heated water and tempered water.
 - How is the exchange performed: producers inject heated water into the heat grid based on the heat load coming from the supplier.
 - Vector of exchange: pipelines connecting the production units to the PowerGrid heat grid.
 - Interface: heat exchangers (heat substations).

4.3.5 Formal requirement level

Once the stakeholders’ intrinsic objectives were defined and the dependencies and interactions between them arise, formal properties (constraints) are defined in order to first rigorously characterize the stakeholders needs using formal KPIs, and secondly to express the relationships between the stakeholders by formally defining their mutual obligations. All of these properties are structured using formal assumptions/guarantees contracts.

We stress that here, the formal properties are first captured using FORM-L language, then, they are translated into the target domain-specific language Modelica in order to make formal verification.

Throughout the PowerGrid project, an important focus was made on identifying the requirements between stakeholders to coordinate the different engineering teams implied. Thus little emphasis was placed on formalizing the specific KPIs of each stakeholder. The main goal was challenging the behavioral models developed by each stakeholder in order to validate the sizing and design under his responsibility. Therefore, we will limit ourselves in the sequel to the analysis of the two stakeholders “Consumers” and the “E-DSO”, and the contract between them. This scope also serves the conciseness of this document.

4.3.5.1 Formal KPIs

Consumers The nature of this stakeholder automatically implies a diversity within its own constituting individuals which will have interests that can widely differ from one to another. This diversity is also subject to the nature of the buildings of consumers as some of them are connected to the heat grid and others who do not. Therefore, the consumers’ interests were characterized in such a way that each individual or group of individuals can specify their own objectives. Each KPI was allocated with one or more parameters that will be assigned with different values depending on each individual consumer’s objectives.

The four KPIs that were identified for consumers are as follows:

- KPI_1 “Thermal comfort”: characterized using a consumer setpoint temperature k_t at the customer’s premises and a boolean k_s which indicates whether or not the thermal shedding is allowed. The setpoint temperature is between 17 and 24°C. The properties related to this KPI are as follows:
 - **ktCompliance**: When k_s is false (shedding not allowed) the temperature should always be higher or equal than the client setpoint k_t . When k_s is true (shedding allowed), the effective *setpoint* (set by an external signal) must never drop below 15°C and for a period that is below 5% of the time over a year.
 - **temperatureRamp**: When the temperature is below k_t it should not drop with a speed above 1°C/h.
 - **kpiComfort**: Ensure that ktCompliance and temperatureRamp are never violated.
- KPI_2 “Keeping costs manageable”: characterized using a minimum and maximum budgets $minBudget$, $maxBudget$ and constant k_b that refers to the percentage of the budget up to which the stakeholder is ready to go if the energy is greener. The property related to this KPI is as follows:
 - **Budget**: The annual electricity and heat bills must not exceed $minBudget + (maxBudget - minBudget) * (1 - kb)$ $minBudget$ and $maxBudget$ will be es-

timated once the behavioral model of consumers will be set. A first simulation will be made considering the minimum comfort profile (setpoint temperature $kt = 17^{\circ}C$) while subscribing to the cheapest energy offer will allow determining *minBudget*. A second simulation will be made considering the maximum comfort profile (setpoint temperature $kt = 23^{\circ}C$) while considering the subscription costs of a green and local energy contract, which is generally more expensive than classical offers.

- KPI_3 “Green energy”: characterized using a boolean constant k_r that states whether or not renewables are required. If k_r is true, the consumer automatically accepts to pay more if necessary. The property related to this KPI is as follows:
 - **Green:** Over a year, the total energy (electricity and heat) provided to the consumer must at least have 25% of renewable energies.
- KPI_4 “Local production”: characterized using a boolean constant k_l that states whether or not a local production of energy is required. If k_l is true, the consumer automatically accepts to pay more if necessary. The formal related to this KPI is as follows:
 - **Local:** Over a year, the total energy (electricity and heat) provided to the consumer must at least be produced locally with a rate of 50%.

The parameters characterizing the four KPIs are correlated (for instance, we cannot consume more energy to guarantee a thermal comfort at any time without impacting the cost of the energy bill). The idea is to illustrate how the CReMA methodology can help in obtaining acceptable compromises taking into account each stakeholder constraints.

In the sequel, and given that we limit our illustration example to the contract between the consumers and the E-DSO, we will only consider the thermal comfort KPI which is directly tied to the behavior of the electric grid.

Figure 4-30 shows an overview of KPI_1 written in FORM-L.

The three properties of KPI_1 were later translated manually into Modelica language using the ReqSysPro library as shown in Figure 4-31.

The representation of KPI_2 , KPI_3 and KPI_4 can be seen in Appendix D.

E-DSO The high-level intentions of the E-DSO were formalized using two main KPIs. The first KPI concerns the safe distribution of electricity goal. It is characterized with the property of ensuring an appropriate voltage at every point of the electric grid that will be called “Node”, including all the network elements such as the transformers, the distribution lines, and connection points with clients. The second KPI concerns the financial aspects. It


```

Consumer begin
  partial model Comfort begin
    Temperature temperature is external; // Room temperature,
                                         // from physical model
    Temperature kt begin // Requested temperature setpoint
      value is specific;
      ensure value in [17, 24]*°C;
    end kt;

    Temperature setpoint; // Effective temperature setpoint
                          // provided to physical model
    constant Boolean ks is specific; // Whether shedding is allowed

    // Yearly duration of effective setpoint lower than kt
    Duration notKt is yearlyDuration (setpoint < kt);

    objective ktCompliance i
      // If shedding is not allowed, setpoint must not drop below kt
      during not ks ensure notKt = 0*s
      // If shedding is allowed, setpoint must not drop below 15°C,
      // and be below kt not more than 5% of the time over a year
      otherwise ensure setpoint >= 15*°C and notKt <= year*5*perCent;

    objective temperatureRamp is
      during temperature < kt
      ensure temperature.derivative <= 1*°K/h;

    objective kpiComfort is
      ensure no (ktCompliance.eViolation or temperatureRamp.eViolation);
    end Comfort;
  end Consumer;

```

Figure 4-30: Consumers KPI_1 written in FORM-L

is characterized using the property of having a minimum revenue for the right operation of the distribution system. These KPIs are defined as follows:

- KPI_1 “Distribute electricity safely”: the appropriate voltage at any electricity node is characterized the following properties:
 - “**absoluteLimit**”: The instantaneous electric voltages between the ground and each phase are always in the $230V - 10\% + 5\%$ range.
 - “**absoluteLimit3**” : The instantaneous electric voltages between phases are always in the $400V - 10\% + 5\%$ range.
 - “**averageLimit**”: The electric voltages between the ground and each phase, averaged over 10 minutes, should be in the $230V - 15\% + 10\%$ range, more than 95% of every calendar week.

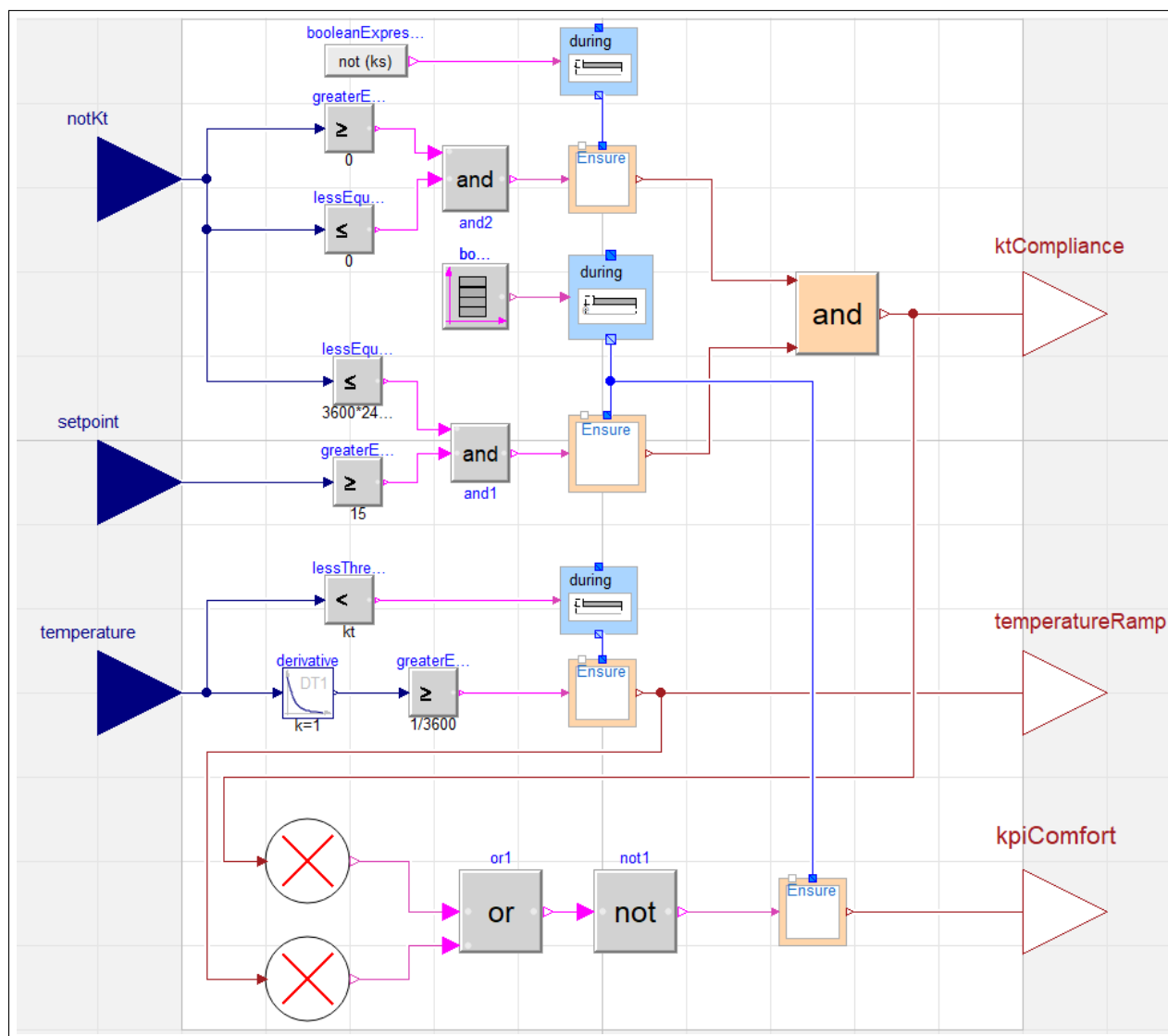


Figure 4-31: KPI_1 represented using ReqSysPro blocks in Modelica

- “**averageLimit3**”: The electric voltages between the ground and each phase, averaged over 10 minutes, should be in the $400V - 15\% + 10\%$ range, more than 95% of every calendar week.
- KPI_2 “Keeping costs manageable”:
 - “**Budget**” Over one year, the total revenues must be 1% higher than the total costs of the annual operation of the distribution grid.

In the sequel, only KPI_1 of the E-DSO is simulated due to a lack of financial data on the distribution operation.

4.3.5.2 Formal contracts

Figure 4-32 shows the global contract model that gathers all the contracts that were established between the PowerGrid stakeholders.

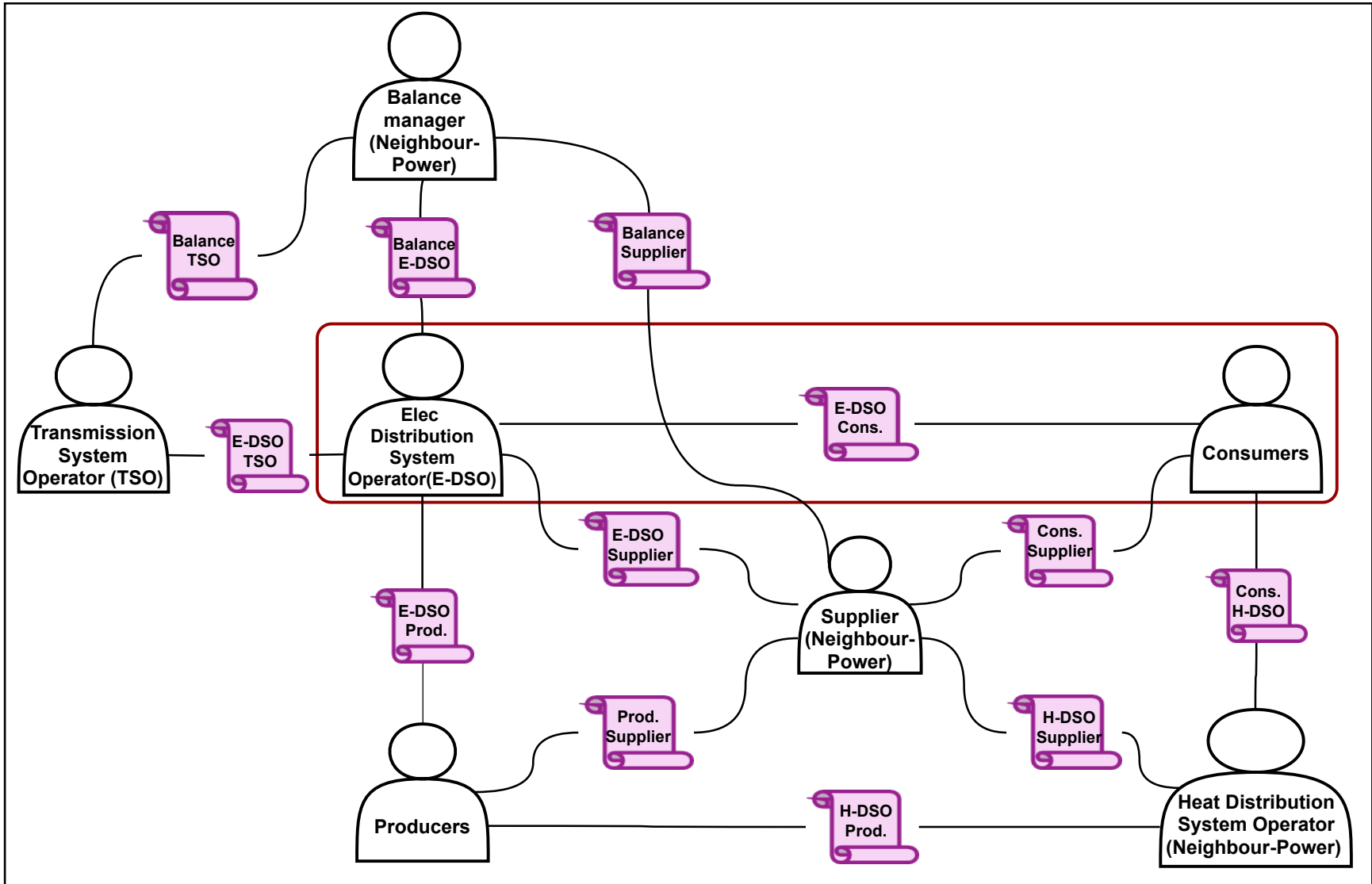


Figure 4-32: Global contract model

As mentioned earlier, we will focus here on the contract “Cons & E-DSO” between consumers and the E-DSO, which is stressed by the red frame in Figure 4-32.

The starting point for defining the contract’s requirements between them is the dependency model that was earlier presented and which is duplicated in Figure 4-33. Let’s take

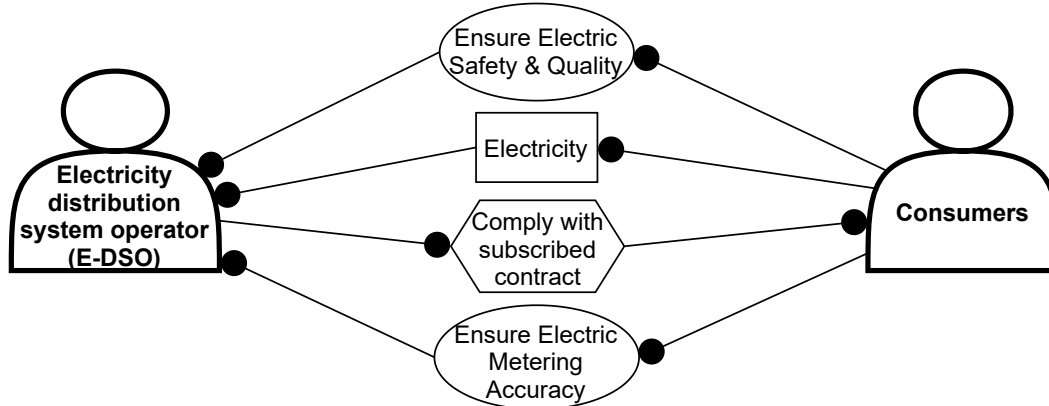


Figure 4-33: E-DSO - Consumers dependency model

the distributor’s point of view and establish his sub-contract including the assumptions and guarantees regarding consumers.

The dependency “Comply with subscribed contract” of the E-DSO regarding the consumers have led to the following requirements, which are taken as assumptions by the E-DSO:

- Assumption 1 “**complyMaxP**”: The active power consumption P should always be below or equal to $maxP$.
- Assumption 2 “**complyMaxQ**” : The reactive power consumption Q should always be below or equal to $maxQ$.
- Assumption 3 “**complyPQConstraint**” : The angle of dephasing φ between P and Q should always be in the range of $+/- 2$ degrees. The values of the thresholds $maxP$ and $maxQ$ were specified according to the historical consumption of buildings.

From an obligation point of view, the E-DSO shall guarantee the dependencies “Electricity” as a resource, “Ensure electric safety & quality”, and “Ensure electric metering accuracy” to consumers. Firstly, it is the duty of the E-DSO to make sure that when the access point ensuring the physical interface between a consumer and an electricity network is in operation, the electricity voltage at that point is appropriate. We stress here that the contract specifies the safety property exclusively for the connection points with consumers in contrast with the E-DSO KPIs that include a wider scope including all the electric grid nodes. Secondly,

the E-DSO needs to ensure the accuracy of the power metering every 10 minutes, and the energy metering every 24 hours.

These dependencies have led to the following requirements, which are considered as guarantees by the E-DSO:

- Guarantee 1 “**absoluteLimit**”: The instantaneous electric voltages between the ground and each phase are always in the $230V - 10\% + 5\%$ range.
- Guarantee 2 “**absoluteLimit3**” : The instantaneous electric voltages between phases are always in the $400V - 10\% + 5\%$ range.
- Guarantee 3 “**averageLimit**”: The electric voltages between the ground and each phase, averaged over 10 minutes, should be in the $230V - 15\% + 10\%$ range, more than 95% of every calendar week.
- Guarantee 4 “**averageLimit3**”: The electric voltages between the ground and each phase, averaged over 10 minutes, should be in the $400V - 15\% + 10\%$ range, more than 95% of every calendar week.
- Guarantee 5 “**meteredP.accuracy**”: The measurement of the active done every 10 minutes should have an accuracy of $+/- 0.2\%$.
- Guarantee 6 “**meteredQ.accuracy**”: The measurement of the reactive done every 10 minutes should have an accuracy of $+/- 0.2\%$.
- Guarantee 7 “**meteredEnergy.accuracy**”: The measurement of the energy done every 24h should have an accuracy of $+/- 0.2\%$.

Switching to the opposite standpoint, the consumers will replicate these same requirements on their sub-contract while changing the position of the assumptions into guarantees and vice-versa. We stress that this switching process does not serve as a general rule. The consumers could have defined different requirements or could have set different target values for the constraints of their sub-contract. In that case, the compatibility between the sub-contracts must be verified based on the formal definition presented in Appendix C, section ??.

In that respect, the consumers sub-contract is defined as follows:

- Assumption 1 “**absoluteLimit**” : The instantaneous electric voltages between the ground and each phase are always in the $230V - 10\% + 5\%$ range.
- Assumption 2 “**absoluteLimit3**” : The instantaneous electric voltages between phases are always in the $400V - 10\% + 5\%$ range.

- Assumption 3 “**averageLimit**” : The electric voltages between the ground and each phase, averaged over 10 minutes, should be in the $230V - 15% + 10%$ range, more than 95% of every calendar week.
- Assumption 4 “**averageLimit3**” : The electric voltages between the ground and each phase, averaged over 10 minutes, should be in the $400V - 15% + 10%$ range, more than 95% of every calendar week.
- Assumption 5 “**meteredP.accuracy**”: The measurement of the active done every 10 minutes should have an accuracy of $+/- 0.2%$.
- Assumption 6 “**meteredQ.accuracy**”: The measurement of the reactive done every 10 minutes should have an accuracy of $+/- 0.2%$.
- Assumption 7 “**meteredEnergy.accuracy**”: The measurement of the energy done every $24h$ should have an accuracy of $+/- 0.2%$.
- Guarantee 1 “**complyMaxP**”: The active power consumption P should always be below or equal to $maxP$.
- Guarantee 2 “**complyMaxQ**” : The reactive power consumption Q should always be below or equal to $maxQ$.
- Guarantee 3 “**complyPQConstraint**” : The angle of dephasing φ between P and Q should always be in the range of $+/- 2$ degrees. The values of the thresholds $maxP$ and $maxQ$ were specified according to the historical consumption of buildings.

In the sequel, the requirements concerning the metering accuracy will not be considered because the metering mechanisms were not modeled in the behavioral modeling level.

Figure 4-34 shows an overview of the assumptions and guarantees included in the E-DSO sub-contract, and written in FORM-L language.

The contracts were then translated manually into Modelica language using the ReqSysPro library. To show an example, Figure 4-35 illustrates how the requirement “absoluteLimit” is represented using ReqSysPro. The other requirements representations can be found in Appendix D.

4.3.6 Design justification level

As mentioned earlier, the design justification level was not applied by the involved engineers during the design phases of the PowerGrid as focus were put on the definition of the interfaces and the coordination between the different engineering teams implied. Nevertheless, we will illustrate this level of CReMA methodology on three examples from different kinds.

```

for all x in consumers // Consumers refer to the buildings
  assumption x.complyMaxP is
    ensure abs(sum(x.p)) in [0*_W, abs(maxP)];
  assumption x.complyMaxQ is
    ensure abs(sum(x.q)) in [0*_W, abs(maxQ)];
  assumption x.complyPqConstraint is
    ensure arcTan(x.p*x.q^-1) in [-2,2]*degree;
end x;

for all x in nodes // Nodes refer to the electric access points
  guarantee x.absoluteLimit is
    during x.service // Whether the access point is operational
      for all y in x.phases // Every phase of the access point
        ensure y.voltage in 230*[0.9, 1.05]*_V;
      guarantee x.absoluteLimit3 is
        during (x.service
          and x.elecType <> lt1) // lt1: low tension single phase
          for all y in x.interPhases
            ensure y.voltage in
              if x.elecType = mt3
                then refVoltage*[0.90, 1.10]
              else 400*[0.85, 1.10]*_V;
          guarantee x.averageLimit
            every 7*day while (c.service
              and x.elecType <> mt3 // mt3: medium tension
              and time > t0) //three phases
              ensure x.nbSatisfaction / (6*24*7) >= 0.95;
          guarantee x.averageLimit3
            every 7*day while (c.service
              and x.elecType <> mt3
              and time > t0)
              ensure x.nbSatisfaction3 / (6*24*7) >= 0.95;
        end x;
      end x;
  end x;
end x;

```

Figure 4-34: E-DSO subcontract written in FORM-L language

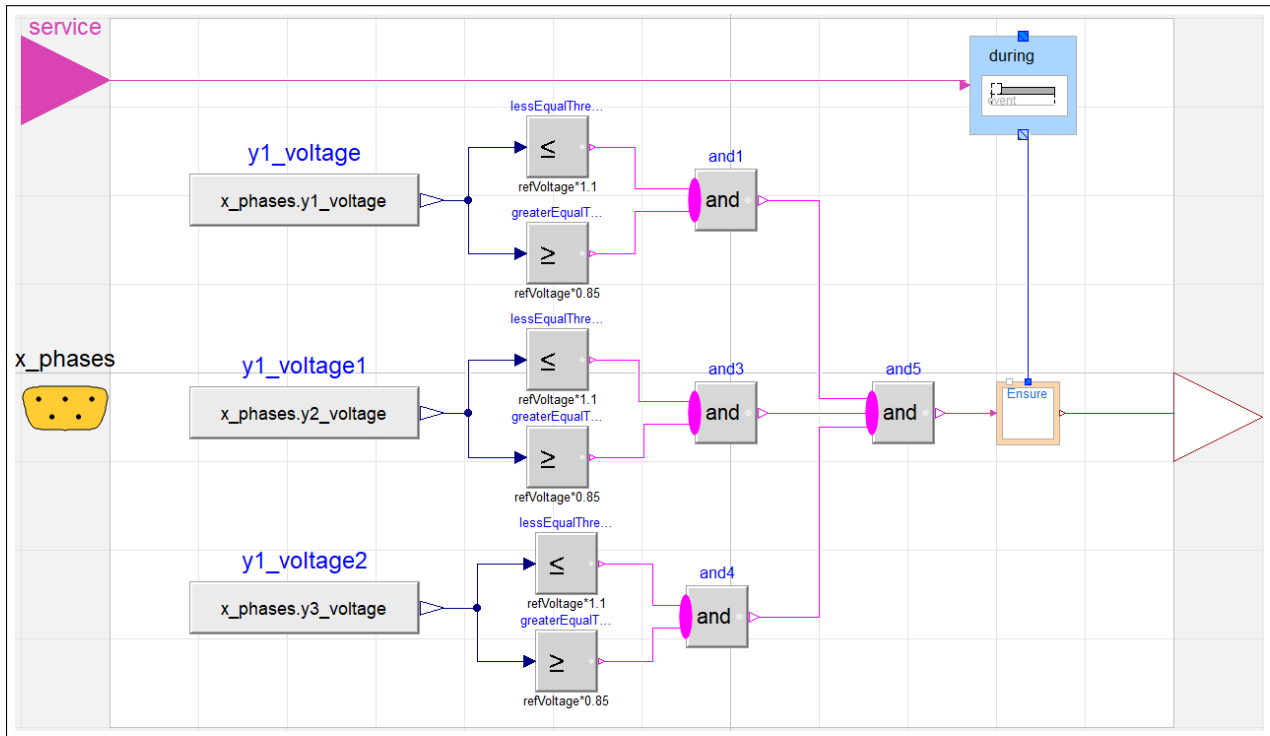


Figure 4-35: absoluteLimit requirement translated into Modelica using using ReqSysPro

4.3.6.1 Example 1: Prove the safe distribution of electricity to consumers

The graphical illustration of the justification chain of this property is shown in Figure 4-36.

- **Claim C_1 :** The electricity is safely distributed to consumers under the three considered scenarios of 0%, 50%, and 100% PV.
- **Argument A_1 :** Claim C_1 is formalized using the claim C_2 .
- **Side-Claim S_1 :** The official documents of the distribution grid operator ENEDIS [4] and the regulatory texts mentioned in it specify the requirements that should be met at the delivery points to consumers to ensure a safe electricity distribution.
- **Claim C_2 :** The satisfaction of the requirements : absoluteLimit, averageLimit, absoluteLimit3 and averageLimit3 for all the electricity delivery points to consumers under the three scenarios 0%, 50%, and 100% PV.
- **Argument A_2 :** Claim C_2 is decomposed into three claims C_3 , C_4 , and C_5 , one for each scenario of PV shares 0%, 50%, and 100% respectively.

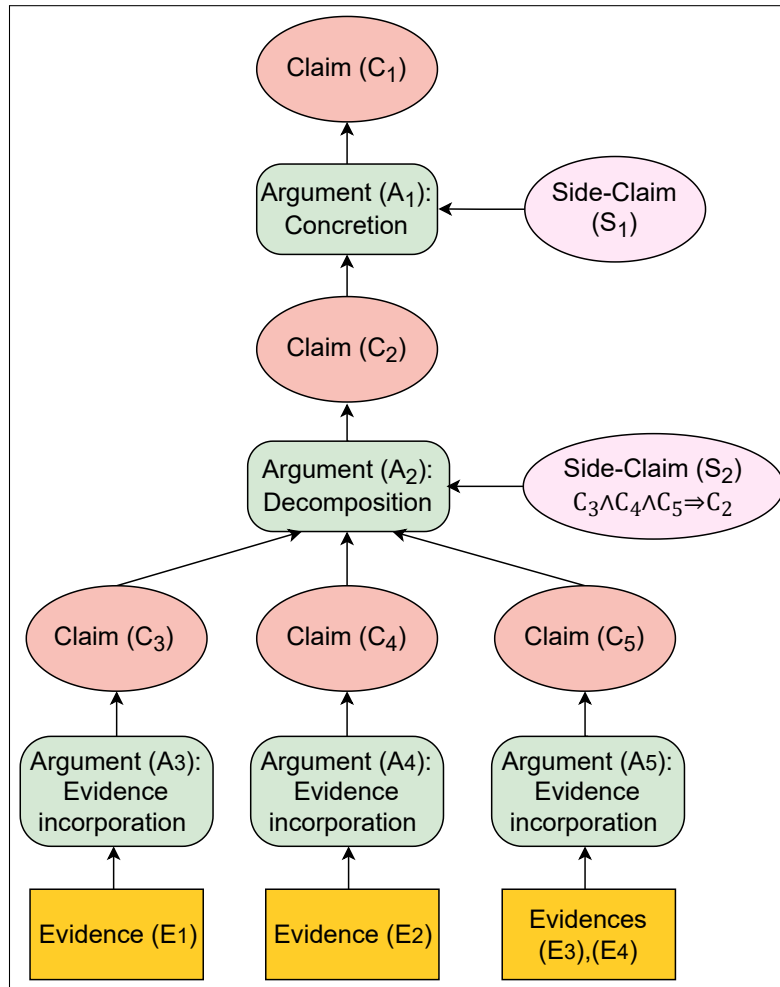


Figure 4-36: CAE structure of the justification of the safe distribution of electricity to consumers

- **Side-Claim** S_2 : There are no dependencies or relationships between the three claims C_3 , C_4 , and C_5 , and therefore the separate satisfaction of these claims implies the satisfaction of the refined claim C_2 .
- **Claim** C_3 : the satisfaction of the requirements : absoluteLimit, averageLimit, absoluteLimit3 and averageLimit3 for all the electricity delivery points to consumers under the scenario 0% PV.
- **Argument** A_3 : The simulation results of evidence E_1 have shown that all the requirements are satisfied for the scenario of 0% PV and therefore the electricity distribution is safe in this case.
- **Evidence** E_1 : The simulation results of 0% PV scenario show that the voltage constraints have been satisfied for all the connections points with buildings throughout

the whole year.

- **Claim C_4** : the satisfaction of the requirements : absoluteLimit, averageLimit, absoluteLimit3 and averageLimit3 for all the electricity delivery points to consumers under the scenario 50% PV.
- **Argument A_3** : The simulation results of the piece of evidence E_2 have allowed to identify 2 requirements violations for the scenario of 50% PV. Zooming into these results shows that there were minor voltage thresholds overrun. The experts of the electric distribution networks have stated that in the real distribution grid, such minor thresholds overruns are automatically compensated with regulation mechanisms that were not modeled in the behavioral models.
- **Evidence E_2** : The simulation results of 50% PV scenario show that the voltage constraints have been satisfied for 18 out of 20 connection points with buildings throughout the whole year. The constraints violations are due to a minor threshold overrun of the maximum allowed voltage.
- **Claim C_5** : the satisfaction of the requirements : absoluteLimit, averageLimit, absoluteLimit3 and averageLimit3 for all the electricity delivery points to consumers under the scenario 100% PV.
- **Argument A_4** : The simulation results of the piece of evidence E_3 have allowed identifying 10 requirements violations for the scenario of 100% PV with major thresholds overruns that cannot be compensated with the distribution grid regulation mechanisms. Therefore, the engineers have chosen to test the implementation of storage batteries to flatten the load curves while taking into consideration the regulation mechanisms in behavioral modeling. The piece of evidence E_4 has proven that this implementation allowed meeting all the requirements for safe electricity distribution.
- **Evidences :**
 - E_3 : The simulation results of 100% PV scenario show that the voltage constraints have been satisfied for 10 out of 20 connection points with buildings throughout the whole year. The constraints violations are due to major threshold overruns of the maximum allowed voltage.
 - E_4 : The simulation results of 100% PV scenario while adding storage batteries and regulation mechanisms to the network model show that the voltage constraints have been satisfied for all the connections points with buildings throughout the whole year.

We emphasize that the behavioral model of the distribution grid including storage batteries and regulation mechanisms is still under development at the time of writing the current manuscript. Therefore, the results mentioned for the piece of evidence E_4 are considered as objectives of the studies that are conducted. They were yet assumed to be correct in this section to illustrate the application of the design justification framework.

4.3.6.2 Example 2: Prove the reliability of the hypothesis taken on the outside temperature of buildings during the design phase

The graphical illustration of the justification chain of this property is shown in Figure 4-37.

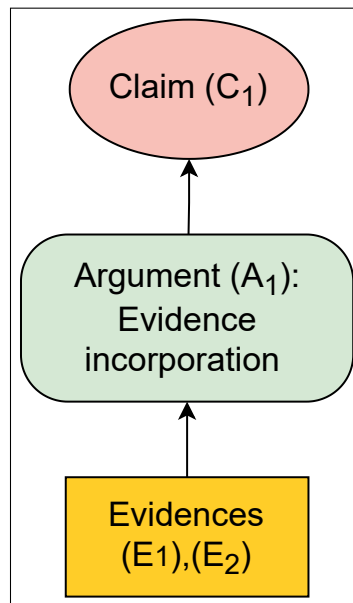


Figure 4-37: CAE structure of the justification of the reliability of a design hypothesis

- **Claim** C_1 : The outside temperature in the PowerGrid district follows the profile named: “Temp_out_30y” that was taken from the public data of Météo France, the official service of meteorology and climatology in France.
- **Arguments** A_1 : Based on the pieces of evidence E_1 and E_2 , the experts in the area of buildings modeling have considered the use of the “Temp_out_30y” profile to be relevant for the outside temperature of the PowerGrid district.
- **Evidences**:
 - E_1 : The profile “Temp_out_30y” takes into consideration the “average” climate over the last 30 years which is very representative for the coming years. In these

30 years, only punctual and rare events have deviated from the profile predictions in the PowerGrid zone.

- E_2 : The profile “Temp_out_30y” is used in the French thermal regulation [3] to calculate the annual consumption of a building. This regulation is the reference for the thermal aspects of the new constructions in France.

4.3.6.3 Example 3: Guarantee good time performances for the E-DSO behavioral model to test prospective scenarios

The graphical illustration of the justification chain of this property is shown in Figure 4-38.

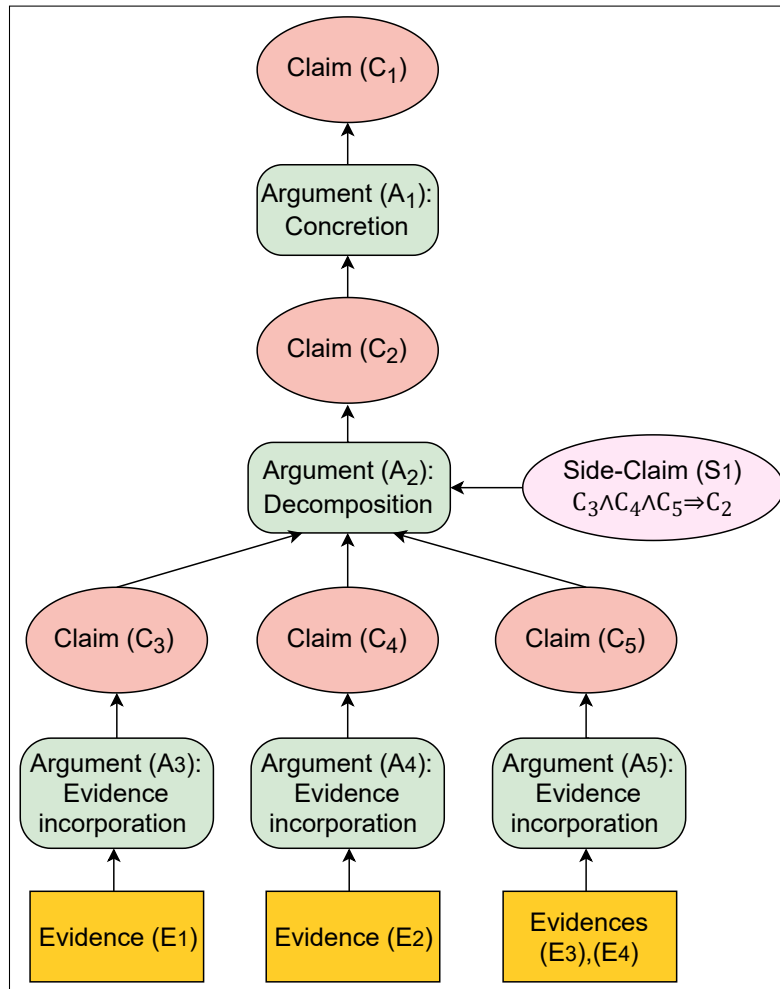


Figure 4-38: CAE structure of the justification of the reasonable simulation time of the E-DSO model

- **Claim C₁**: Developing a behavioral model of the electric grid allowing to have a reasonable simulation time (less than 1h) to evaluate the properties concerning the E-DSO

KPIs and his sub-contract under multiple prospective scenarios (i.e. different penetration rates of solar panels and different monitoring strategies to ensure the electric grid stability).

- **Arguments** A_1 : The load flow in the distribution network of the PowerGrid district was initially represented using the library EPSL. This library was pertinent for the detailed design phase because it represents in detail and allows the evaluation of the power transiting in each of the three phases of the network. However, this library was not convenient for testing multiple scenarios because it induces long time-consuming simulations (around 10 hours for one scenario). Therefore, a new library called PowerSysPro dedicated to the load flow analysis was developed, simplifying the overall representation of the distribution grid and thus decreasing a lot the time computation needed.. PowerSysPro takes the hypothesis that the network is balanced, meaning that the three phases have the same current and voltage. Therefore, a single-phase is represented instead of three [134]. The claim C_1 is thus concertized using the claim C_2 .
- **Claim** C_2 : Switching from the library EPSL to the library PowerSysPro for the electric grid modeling allows having a convenient simulation time.
- **Arguments** A_2 : In order to prove the accuracy of the transition between the library EPSL and the library PowerSysPro, claim C_2 is decomposed into three claims C_3 , C_4 , and C_5 .
- **Side-Claim** S_1 : The modeling experts of the distribution grid have established three criteria for the accurate transition between the two libraries: the calculation accuracy, the size reduction, and the reduction of the simulation time.
- **Claim** C_3 : The calculations made by PowerSysPro are accurate compared to EPSL.
- **Argument** A_3 : The comparison between the simulation results of the PowerGrid load flow model developed in EPSL and the one developed in PowerSysPro have shown that the error induced by the later does not exceed 0.18% referring to the piece of evidence E_1 . This error is considered as an acceptable deviation according to experts for the kind of analysis that needs to be done for the evaluation of the E-DSO KPIs and sub-contract.
- **Evidence** E_1 : Table 4.1 depicts the errors between EPSL and PowerSysPro results at initialization time.

Table 4.1: Relative errors comparing EPSL and PowerSysPro results at initialization time (from [134])

	Source	MV-LV Transformer	1-phase Load	3-phase Load
Efficient Current	-0,028%	0,129%	-0,006%	-0,021%
Line to line Voltage	0,004%	-0,179%	0,015%	0,020%
Apparent Power	-0,023%	-0,050%	0,009%	0,000%

- **Claim C_4 :** The size of models made using PowerSysPro have lower size compared to EPSL
- **Argument A_4 :** Referring to the piece of evidence E_2 , the size of the PowerGrid model in terms of unknowns/equations is nearly 24 times.
- **Evidence E_2 :** Table 4.2 comparing the number of unknowns/equations between a first PowerGrid model developed in EPSL and a second developed in PowerSysPro.

Table 4.2: Comparing the number of unknowns/equations between EPSL and PowerSysPro on the PowerGrid demonstrator (from [134])

	EPSL	PowerSysPro
Number of unknowns/equations	413 475	17 355

- **Claim C_5 :** The simulation time of models developed using PowerSysPro are lower than the ones using EPSL, and are convenient for multi-scenarios analysis.
- **Arguments**
 - A_5 : The comparison between the EPSL model and the PowerSysPro model in terms of initialization time as stated by the piece of evidence E_3 have shown a considerable decrease in the initialization time. It is 25 five times faster for PowerSysPro.
 - A_6 : Referring to the piece of evidence E_4 , the simulation time of the PowerGrid load flow model connected with the buildings model is done in a very convenient simulation time.
- **Evidence:**
 - E_3 : The initialization time is 25 times lower for PowerSysPro compared to EPSL.
 - E_4 : The simulation of the distribution grid model under PowerSysPro library connected to buildings model is done in 6 minutes.

4.3.7 Architecture description level

The architecture description level of the PowerGrid design phase is similar to the architecture description depicted in the energy urban planning phase. The major enhancements lie in :

- the definition of the new stakeholders' scopes using the red frames that can be seen in Figure 4-39, Figure 4-40, and Figure 4-42.
- the decomposition of the architectural description elements into atomic components.
- the allocation of the architectural description elements with the corresponding system static characteristics. This task was not performed in the PowerGrid project as suggested by the thesis because the common practice among the involved engineers is that the static properties are integrated into the behavioral models and not depicted into a separate architectural description model. A change of practice could not be considered for time limitations. Therefore, only one example of static properties will be given in the following.

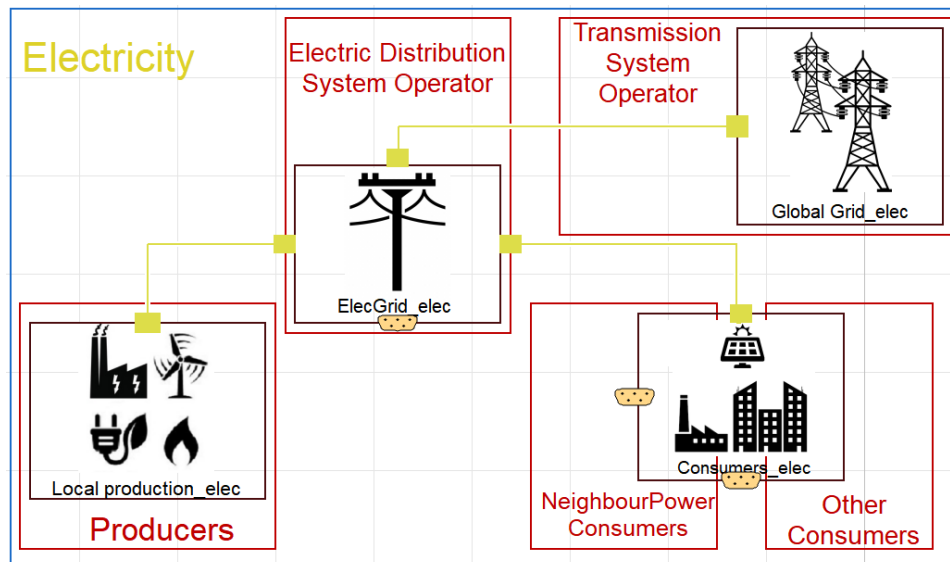


Figure 4-39: PowerGrid electric architecture

From the electricity point of view (Figure 4-39), the producers are in charge of operating the local electricity production units that are located in the PowerGrid District. The E-DSO is in charge of the operation of the electric grid. The TSO is in charge of the operation of the transmission grid. Finally, two kinds of consumers are identified, the ones who have subscribed to a contract provided by the supplier NeighbourPower, and others who chose other suppliers. The consumers' scope involves the buildings and the PV panels that will

be installed on their roofs. In this thesis, we will only focus on the consumers that have subscribed to NeighbourPower.

From the heat point of view (Figure 4-40), the producers are in charge of operating the local heat production units. NeighbourPower is considered to be the H-DSO and is in charge of operating the heat grid. Finally, the buildings connected to the heat grid are all considered clients of NeighbourPower because the later has the monopoly of heat-supply in the PowerGrid district.

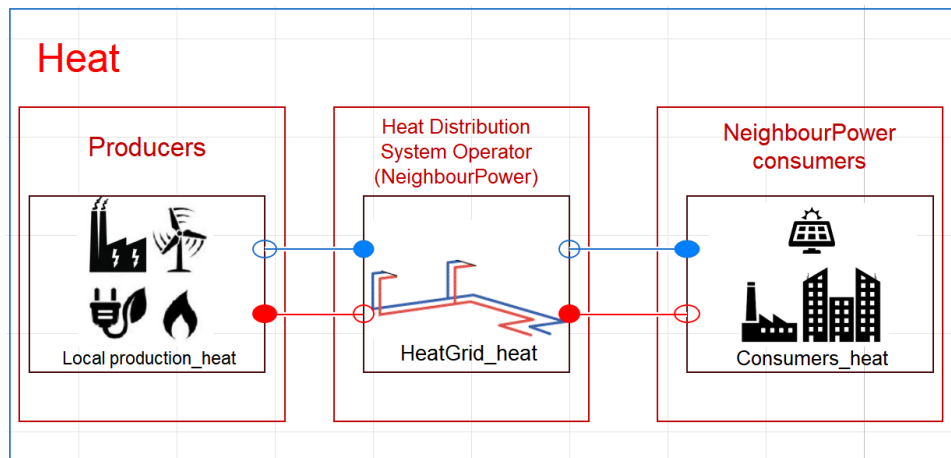


Figure 4-40: PowerGrid heat architecture

To show an example of the static system characteristics that can be captured in the architectural description level, the buildings' architectural description was allocated with a set of parameters indicating the type of the primary energy that is used for heating by each building. Figure 4-41 shows the Modelica text view of the architectural description including the buildings' parameters. These parameters will be later captured by the observation model which is dedicated to the computation of the inputs of the stakeholders' KPIs model.

Finally, and from the economical point of view (Figure 4-42), only a few scopes are clearly set at this point. NeighbourPower is in charge of managing the economical aspects related to the heat supply and a part of the electricity supply which is shared with other competitor suppliers. The balance manager is in charge of the financial aspects of balance management. The financial aspects of the other systems are harder to be assigned to any specific stakeholder because of various reasons. Regarding the economical aspects of the buildings, they are part of consumers' responsibility for obvious reasons, however, amongst the possible future scenarios can be found the case where the municipality or a national organism takes in charge the renovation costs of buildings or can provide grants to support the energy transition. Therefore, these entities would be the ones in charge of the financial aspects of the investments made for the district renovation. Concerning the financial aspects

```

model Buildings_Architecture_Description
  // Declaration of buildings energy systems parameters
  // Fuel type according to the GoeJSON file : 1 = DHN;
  // DHN = 1; Boiler = 2;Elec = 4; Biomass = 8
  // Declaration of buildings energy systems parameters
  energy_system energy_system_BATIMENT0000000005011407(heating_fuel_type=1)
  ;
  energy_system energy_system_BATIMENT00000000356487656(heating_fuel_type=1)
  ;
  energy_system energy_system_BATIMENT00000000322361106(heating_fuel_type=1)
  ;
  :
  :
  energy_system energy_system_BATIMENT00000000320456131( heating_fuel_type=4)
  ;
  energy_system energy_system_BATIMENT0000000005011465(heating_fuel_type=4)
  ;
  ;
end Buildings_Architecture_Description;

```

Figure 4-41: Parameters characterizing the architecture of the PowerGrid buildings

of the heat grid and the production units, they are also subjected to the same previous possibilities. The entities operating these systems are sometimes not the owners and strongly depend on other organisms such as state organizations for their financial balance. Finally and similarly to previous elements, the electric grid may require new investments in order to be adapted to the new constraints of the district. Thus, the municipality of the PowerGrid district may be involved in the financial aspect of the electric grid.

The definition of the economical boundaries will be therefore the fruit of discussions and negotiations between stakeholders all along the development process of the PowerGrid.

4.3.8 Behavioral level

The behavioral models of the various sub-systems constituting the PowerGrid were developed in the Modelica language using the Dymola tool in the framework of the FUI ModeliScale project.

4.3.8.1 Consumers scope

The consumers' scope including the buildings and the PV panels was represented using the Modelica library called BuildSysPro [125]. Figure 4-43 shows an overview of the behavioral model. This model includes two main parts:

- A physical model representing the heat transfer phenomena between the indoor air

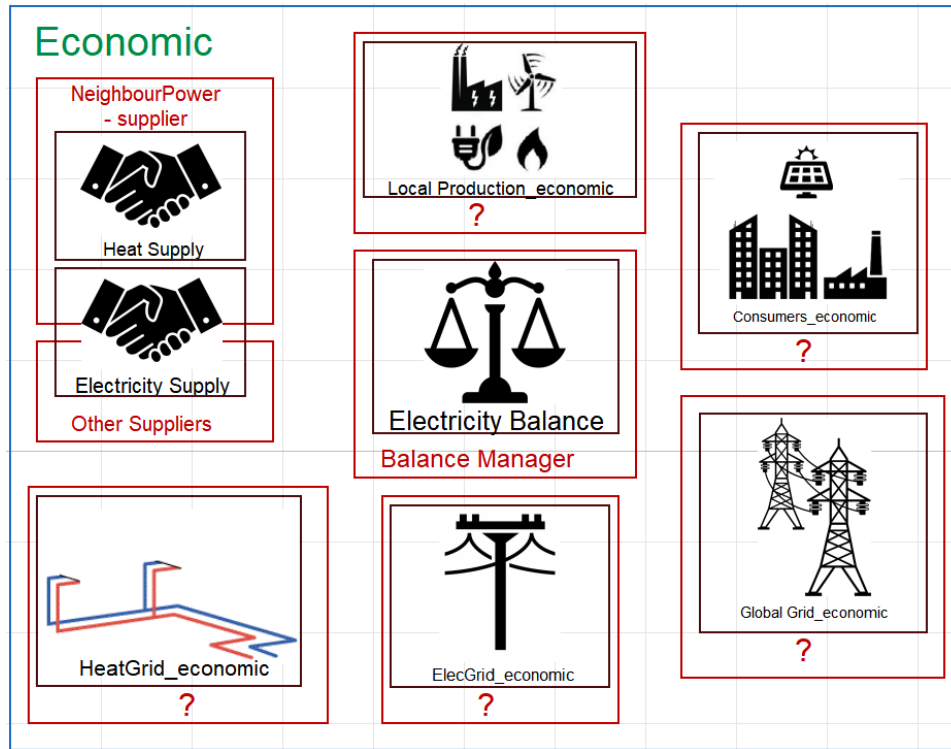


Figure 4-42: PowerGrid economic architecture

zone of buildings and the outdoor weather.

- A consumption/production model that represents on the one side all the energy needs of each building including the demands in terms of specific electricity, heating, and domestic hot water. On the other hand, the model represents the energy produced by the PV panels installed on the roof of each building while taking into account the masking and reflection phenomena between them.

4.3.8.2 E-DSO scope

The scope of the E-DSO including the electric distribution grid was modeled using the EPSL Modelica library [67]. This behavioral model aims to compute the electric load flow going through the network which contains the following components:

- 20 connection points with the considered buildings
- 1 primary HV ideal electric source
- 1 primary HV/MV transformer from High Voltage (225kV) to Medium Voltage (20kV)

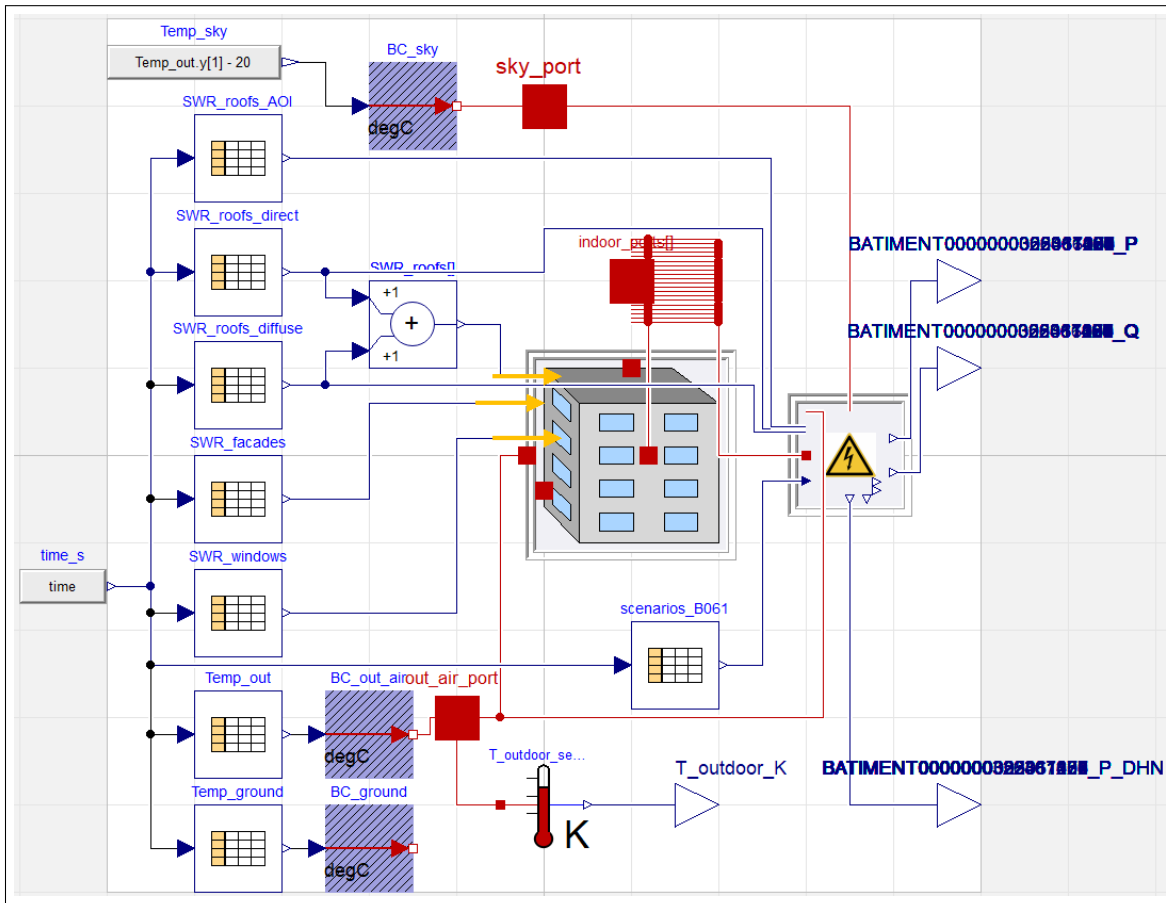


Figure 4-43: Behavioral model of the 20 buildings [75]

- 2 secondary MV/LV transformers from Medium Voltage (20kV) to Low Voltage (400V)
- 1 three phases HV electric line
- 14 three phases MV electric lines
- 9 three phases LV lines

Figure 4-44 shows a partial overview of the distribution grid's behavioral model. It illustrates a typical MV/LV transformer feeding electricity to buildings or receiving it when the buildings are equipped with PV panels. The electric lines that create electrical losses during distribution can also be seen in this same Figure.

4.3.9 Observation models and bindings

Once the different patterns describing the stakeholders' interests are set and a first version of the formal contracts established, and also the architectural and behavioral models of each stakeholder are developed, we proceed to the construction of the observation models and set

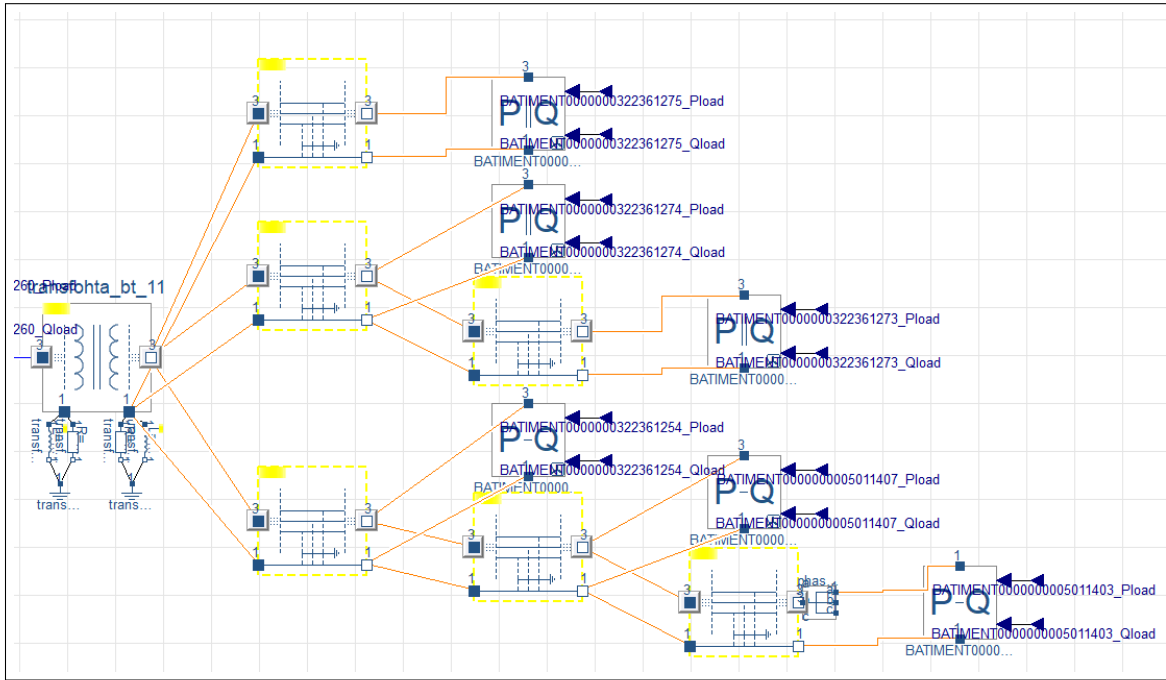


Figure 4-44: Partial overview of the distribution grid’s behavioral model for the 20 buildings [75]

the bindings between the different modeling bricks before setting the verification models in order to ensure the communication between the different modules. For each stakeholder, a set of bindings and an observation model are established for the verification of his own KPIs, and for each stakeholder sub-contract, another set of bindings and an observation model are established.

The observation models were first developed using FORM-L language and then translated into Modelica language using its standard mathematical and physical libraries.

4.3.9.1 Observation model for consumers KPIs

The goal of this observation model is to observe some specific physical states and parameters from the behavioral model of consumers and also to observe some parameters of the sub-contract established with the E-DSO and transform them into inputs feeding the KPIs model. This observation model was separated into two sections because the observation elements that compute the inputs of KPI_1 “Thermal comfort” and KPI_2 “Keeping costs manageable” have a different nature compared to the one computing the inputs of KPI_3 “Green energy” and KPI_4 “Local production”. Indeed, the evaluation of KPI_1 and KPI_2 is specific to each building and depends on its physical states such as the temperature inside each building or its power demand. Therefore, an observation model is necessary for each building. However, the evaluation of KPI_3 and KPI_4 requires the assessment of the energy consumption and

production at the scale of 20 buildings in order to measure the renewable energy and the local production rates. Therefore, a unique but common observation model will be set for the computation of the inputs of KPI_3 and KPI_4 of all consumers. This does not mean that KPI_3 and KPI_4 are identical for all consumers, it only means that the computation of their input is similar to all the consumers. Each consumer has the ability to set his satisfaction thresholds for KPI_3 and KPI_4 .

The first observation section that computes the inputs of KPI_1 and KPI_2 observes the following physical states from the behavioral model:

- The active power $P(W)$ demand of electricity for each building.
- The active power $Q(W)$ demand used for heating and for domestic hot water for each building.
- The heating setpoint $Setpoint(^{\circ}C)$ temperature of each building.
- The temperature $T(^{\circ}C)$ inside each building

Based on these physical states, the observation model computes the following elements:

- The budget $Budget(\text{€})$ corresponding to the electric and heat annual bill of each building
- The yearly duration of effective setpoint lower than kt (*notkt*)
- The heating setpoint $Setpoint(^{\circ}C)$ temperature of each building.
- The temperature $T(^{\circ}C)$ inside each building

The last two outputs have not undergone any transformation between the input of the model and its output. They were still included in the observation model the later constitutes the unique intermediate between the behavioral model and the requirement models.

The consumers bill (or budget) is computed according to (i) the type of electricity that he subscribed to, (ii) whether he is connected to the heat network, and (iii) whether the consumer have solar panels and injects his electricity surplus into the grid.

The electric and heat costs as well as the subscription fees are defined in the verification models as simulation parameters that will vary depending on the choices of the consumers.

Similarly to requirement models, the observation models are first written using FORM-L language and then translated manually into Modelica using standard libraries and coding. Figure 4-45 depicts the first section of the observation model written in FORM-L language.

```

Temperature      kt begin is specific; // Requested temperature setpoint
constant Boolean elecHeating is specific;
constant Money   elecContract.yearlySubscriptionFee is specific;
constant Money   heatContract.yearlySubscriptionFee is specific;
Money/Energy     elecContract.energyCost           is specific;
Money/Energy     heatContract.energyCost           is specific;
Temperature      setpoint is external; // Effective temperature setpoint
// provided to physical model
Temperature      temperature is external; // Room temperature, from
physical model

// Yearly duration of effective setpoint lower than kt
Duration notKt is yearlyDuration (setpoint < kt);

Money budget is
if elecHeating
then elecContract.yearlySubscriptionFee
  + yearlyIntegral((heatingPower+otherPower)*elecContract.energyCost)
else elecContract.yearlySubscriptionFee
  + yearlyIntegral(otherPower*elecContract.energyCost)
  + heatContract.yearlySubscriptionFee
  + yearlyIntegral(heatingPower*heatContract.energyCost);
end Budget;

```

Figure 4-45: The observation model for KPI_1 and KPI_2 written using FORM-L language

Figure 4-46 and Figure 4-47 respectively depict the Modelica model representing the first section of the observation model from a diagram view (using Modelica blocks) and from a text view (using Modelica code).

As mentioned earlier, this first section of the observation model is specific to each of the 20 consumers. Therefore, 20 instantiations need to be made for this section, while assigning the right attributes for each instance. Given the number of models, a manual instantiation seems to be time-consuming with high risks of making errors. Thus, a python code was developed to automatically generate a Modelica model containing the observation model instances corresponding to each building while setting their parameters. An overview of the python code can be seen in Appendix E, and more specifically in Figure E-4 and Figure E-5.

The second observation section that computes the inputs of KPI_3 and KPI_4 observes the following physical states from the behavioral model:

- The active power $P(W)$ demand of electricity for each building.
- The heating power $P_{DHN}(W)$ demand for each building (if connected to the district heating network).
- The active energy $E_{PV}(kWh)$ generated by the PV panels installed at each building (if equipped).

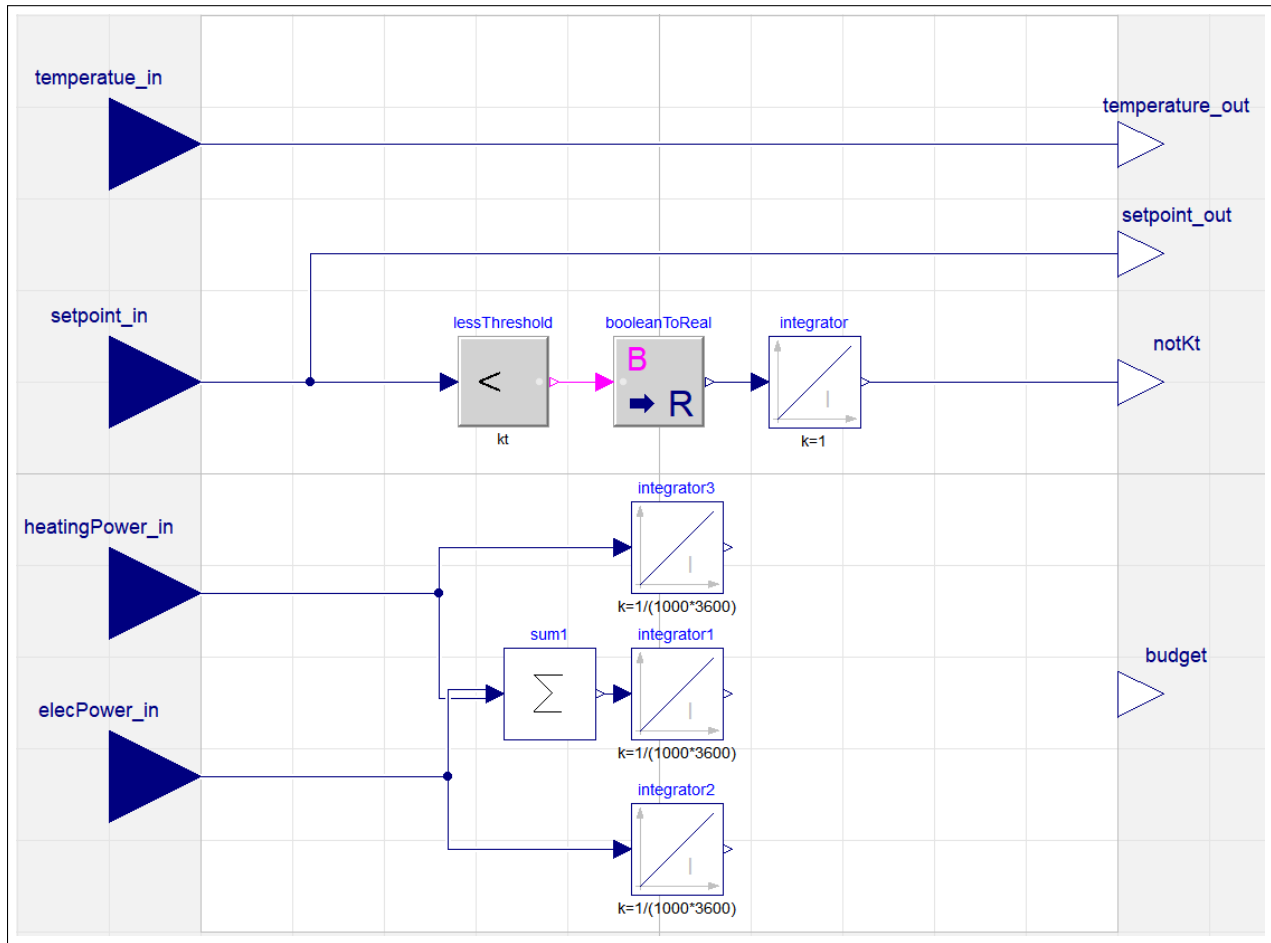


Figure 4-46: Modelica diagram view of the observation model for KPI_1 and KPI_2

Based on these physical states, the observation model computes the following elements:

- The total energy (heat and electricity) renewable rate $Tot_Energy_RES_rate$.
- The local production rate including heat and electricity $Local_production_rate$.

The Modelica description model of the second section of the observation model set for KPI_3 and KPI_4 can be found in Appendix E.

4.3.9.2 Bindings for KPIs evaluation

In order to connect the required modules for the evaluation of KPIs, a python script was developed to automatically generate the Modelica code that ensures the following connections:

- The parameters from the architecture description model with the observation model.
- The outputs of the behavioral model with the inputs of the observation model.


```

model Observer_per_consumer

  parameter Real kt=17 "Requested temperature setpoint";
  parameter Boolean elecHeating=true "whether the buiding is heated by electricity or not";
  parameter Real elec_yearlySubscriptionFee=1000 "electricity yearly subscription fee €";
  parameter Real elec_energyCost=0.16 "electricity cost €/kWh";
  parameter Real heat_yearlySubscriptionFee=1000 "heat yearly subscription fee €";
  parameter Real heat_energyCost=0.08 "heat cost €/kWh";

equation
  // computation of the annual budget
  budget= if elecHeating
    then elec_yearlySubscriptionFee + integrator1.y*elec_energyCost
    else elec_yearlySubscriptionFee + integrator2.y*elec_energyCost
      + heat_yearlySubscriptionFee + integrator3.y*heat_energyCost;
end Observer_per_consumer;

```

Figure 4-47: Modelica text view of the observation model for KPI_1 and KPI_2

- Internal physical states of the behavioral model with the inputs of the observation model. (The difference with the previous statement is that the behavioral model was not conceived with dedicated outputs for these physical states. Bindings are not meant to be intrusive to models by requiring any changes)
- The outputs of the observation models with the inputs of the KPIs model.

Figure 4-48 depicts the different elements composing the consumers' model in Modelica with an emphasis on the bindings connecting them.

4.3.9.3 Observation model for consumers sub-contract regarding E-DSO

The goal of this observation model is to transform some specific physical states observed from the behavioral model of consumers into inputs feeding the sub-contract regarding the E-DSO.

The sub-contract has two inputs for each building: the active power P and the reactive power Q . These outputs match with the outputs of the output of the consumers' behavioral model and therefore, there is no need for any observation model. A set of bindings connecting the two modules is sufficient.

4.3.9.4 Observation model for the E-DSO KPIs

The goal of this observation model is to transform specific physical states observed from the behavioral model of the electric distribution network into inputs feeding the KPIs model. This observation model capture the following physical states for every node of the electric network model:

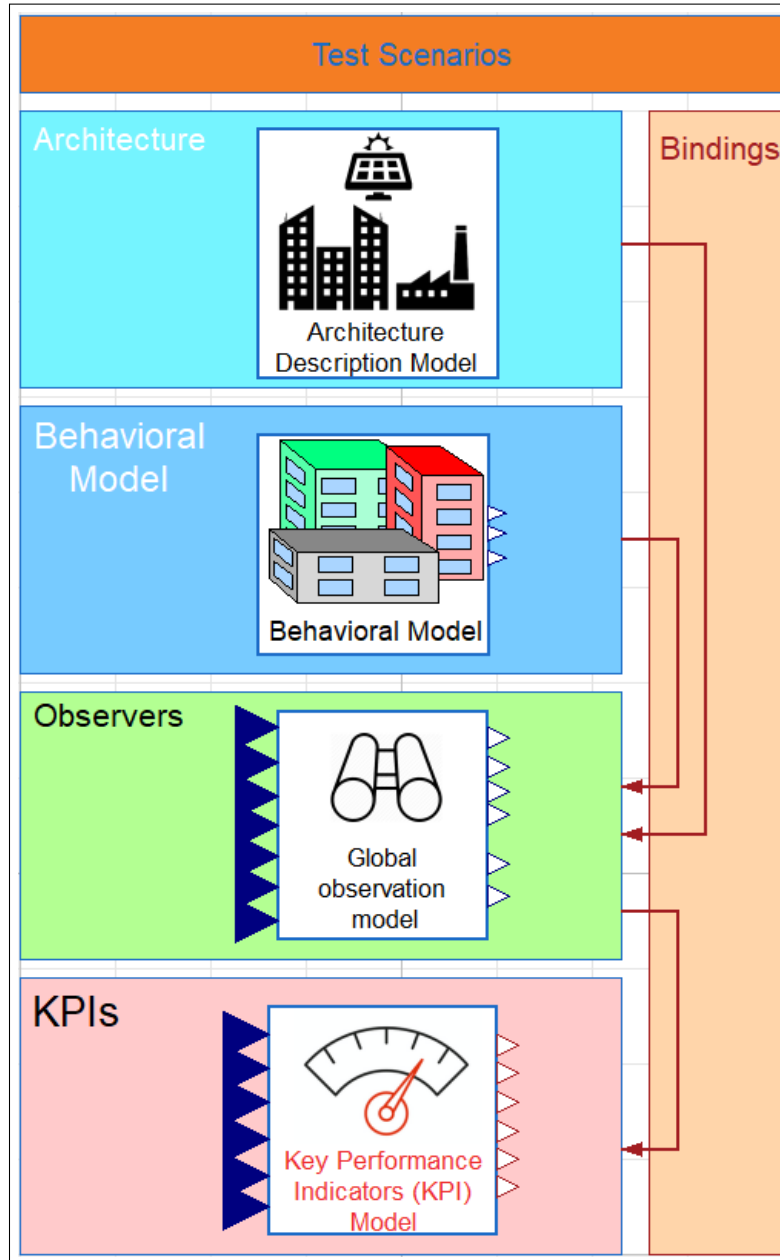


Figure 4-48: The elements composing the consumers description model with the bindings connecting them

- The real part Re of the electric voltage between the ground and the *terminal B* of the node. (terminal B represents a non-causal interface the node)
- The imaginary part Im of the electric voltage between the ground and the *terminal B* of the node.

Based on those two variables, the observation model computes the following elements for every node of the electric network:

- The electric voltage $V(V)$ between the ground and the *terminal B* of the node.
- The number of times $nbSatisfaction1$ the electric voltage, averaged over 10 minutes, stays within the $230V - 15\% + 10\%$ range over one week.

Similarly to the previous observation models, and given the number of 26 nodes, a python script was developed for the automatic instantiation of the observation models corresponding to each node while setting their attributes.

4.3.9.5 Observation model for the E-DSO sub-contract regarding consumers

The sub-contract and the KPIs of the E-DSO include the same properties but on different nodes of the electric grid. Therefore, the same observation model previously developed for KPIs will be used for the sub-contract. The instantiation will be limited to 20 nodes corresponding to the connection points between the electric grid and the buildings.

4.3.10 Verification models

4.3.10.1 Story-board

The renovation of the PowerGrid district into a more eco-friendly neighbourhood induces considerable changes in terms of energy consumption and production which directly impacts, on the one hand the consumers KPIs as well as their ability to fulfill their contracts, on the other hand new constraints are applied to the electric network and thus affect the E-DSO KPIs and obligations. Therefore, two separate verification models are introduced in this section, one from the consumers standpoint and one for the E-DSO standpoint. These verification models will be simulated under the same scenarios used at the urban energy planning phase of the PowerGrid. As a reminder, here are the three scenarios:

- **No measures (0% PV):** represents the landmark of the analysis as it refers to the initial state of the PowerGrid district before renovation. In this scenario, no effort is made towards the implementation of renewable energies in the area (i.e. 0% of buildings are equipped with PV panels). This scenario is tested in order to validate the

consistency of the PowerGrid modeling compared to reality where both the buildings and the electric grid satisfy their part of the contract.

- **Average (50% PV)**: represents an intermediate scenario with a “greener” effort made in regards to the renewable energies. (i.e. nearly 50% of buildings are equipped with PV panels).
- **High PV (100% PV)**: represents the case with the highest possible investments in regards to the renewable energies. (i.e. 100% of buildings are equipped with PV panels).

In the last two scenarios, we consider that the consumers who have equipped their buildings with solar panels have signed “connection contracts” that require from them not to exceed a maximum produced active and reactive powers injected into the grid. These maximum powers have the same values as the consumption power to which the consumer had previously subscribed to. The E-DSO and the consumers will later make verification to check whether the contracts are still satisfied or whether new contracts need to be negotiated.

The verification model of each stakeholder intends to verify three main elements: the satisfaction of his KPIs, his ability to fulfill his contract guarantees towards the other stakeholder, and the compliance of the hypothesis taken on the other stakeholder behavior in regards to the contract assumptions.

The simulation results were analyzed using a MATLAB script as it is convenient for processing large Dymola output files (.mat).

4.3.10.2 Verification model from the consumers’ point of view

The consumers verification model is established by the integration of the four following elements:

- The consumers description model including his KPIs, the architecture description model and behavioral model of his scope.
- The consumers’ sub-contract in regards to the E-DSO exclusively including their guarantees. For the sake of conciseness, we assumed that the E-DSO fulfills his missions and that assumptions on its behavior are always satisfied and hence are not tested within this verification model.
- The bindings linking the two previous elements.
- The test scenario model including the different parameters’ configurations that characterize the scenarios to be simulated.

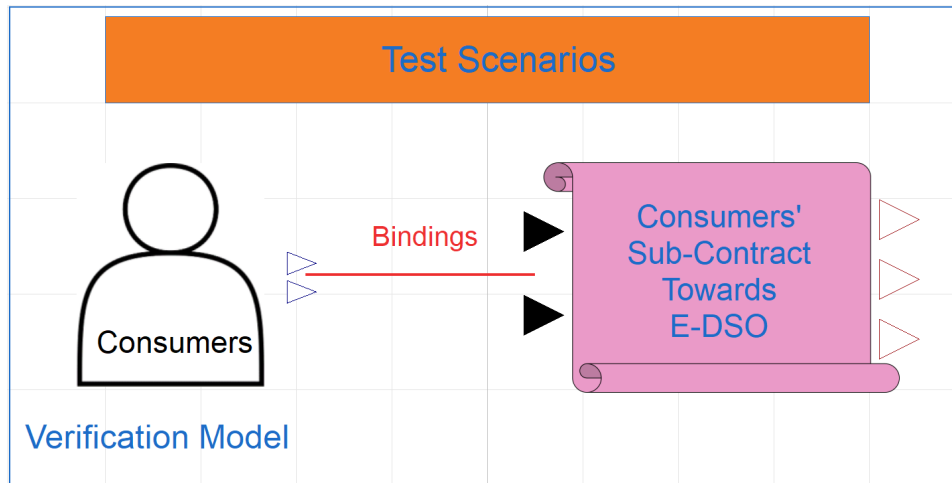


Figure 4-49: Consumers' verification model

Figure 4-49 depicts an overview of the consumers verification model.

In order to make an analysis taking into consideration the diversity of consumers who live in the PowerGrid district, three profiles of consumers were defined with different points of interests and different KPIs and will be used to stimulate the verification model:

- **Budget-conscious consumer:** puts all his priority on the budget ($kb = 100\%$) while putting a low temperature setpoint ($kt = 17^{\circ}C$). No importance is given for having a green and local energy ($kr = false$, $kl = false$).
- **Cold-sensitive consumer:** puts his priority on the heat comfort KPI and sets a high temperature setpoint ($kt = 23^{\circ}C$). Less priority is given to the other aspects : the cost of energy ($kb = 50\%$), the green energy ($kr = false$) and local energy ($kl = false$).
- **Green consumer:** puts his priority on having a green and local energy ($kr = true$, $kl = true$) and regardless of the cost of energy ($kb = 0\%$). The temperature is set at a medium level ($kt = 20^{\circ}C$).

The detailed description of the KPIs' parameters can be found in the earlier section 4.3.5.

4.3.10.3 Simulation of the consumers verification model

Simulation hypothesis :

- In order to make a relevant comparison between the 3 profiles, we will consider during each simulation that the 20 consumers have the same profile (with the same preferences) and thus assign the same KPIs parameters to all of them. This assumption has been made to facilitate the analysis and mainly to draw major trends at the scale of a 20

buildings block. Nevertheless, to be more accurate, a realistic model should consider the diversity of the consumers' KPIs, and also the rate of the simultaneous electrical demand (based on statistical data or surveys).

- In order to encourage the PowerGrid consumers to become “prosumers” (i.e. producers and consumers at the same time) and therefore increase the shares of renewable and local energies of the PowerGrid district, the supplier NeighbourPower introduces a new energy subscription offer for the district consumers. This offer consists in proposing to consumers the installation of solar panels on the roofs of their buildings with following conditions:
 - The initial investment costs of purchasing and installing the solar panels are at the expenses of NeighbourPower.
 - The electricity produced by solar panels is self-consumed at the price of $0.2\text{€}/kWh$. This price is 25% higher than the electricity market price. We consider that it allows NeighbourPower to cover a portion of the investment costs for solar panels.
 - The electricity surplus produced and injected into the network gives right to the consumers of a remuneration of $0.04\text{€}/kWh$. The difference between the consumers remuneration and the electricity sale price of NeighbourPower will allow him to cover a portion of the PV investment costs.
 - The electricity consumed from the electric grid is charged at the price of $0.16\text{€}/kWh$, which is a rounded estimation of the mean electricity price proposed by French suppliers [2].
 - The electricity subscription fee is correlated to the maximum subscribed power using the expression “ $Sub_fees = 8.3 * 10^{-3} * MaxP + 78.1$ ”. This estimation is made based on subscription fees of the “blue” and “green” offer of the supplier EDF for 2020.
- The heat price is set at $0.08 \text{€}/kWh$. Its value was taken from [1] while adding taxes and including subscription fees.
- The simulation of the verification model is made over a duration that is slightly higher than one year (1 year and 10 days) because the requirements were set for a one year period. Therefore, the decision on whether a requirement is satisfied or not can only be made after that period.

Simulation results: Table 4.3 gathers the results of the nine simulations of the verification model under the three scenarios 0% PV, 50% PV and 100% PV and for the three consumers profiles that were specified earlier. This table emphasizes the number of violated KPIs and guarantees of each simulation and depicts the types of the requirements that are violated.

Profiles Scenarios	1: Budget-conscious consumers ($K_t=17^\circ\text{C}$; $k_b=100\%$; $k_r=false$; $k_l=false$)	2: Cold-sensitive consumers ($K_t=23^\circ\text{C}$; $k_b=30\%$; $k_r=false$; $k_l=false$)	3: Green consumers ($K_t=20^\circ\text{C}$; $k_b=0\%$; $k_r=true$; $k_l=true$)
1 : 0% PV	R_{11} : - 0/80 KPI violated - 0/60 guarantee violated	R_{12} : - 12/80 KPIs violated : 12 KPI budget - 5/60 guarantees violation: 2 complyMaxP, 3 complyMaxQ	R_{13} : - 40/80 KPI violated: 20 KPI green, 20 KPI local - 0/60 guarantee violated
2: 50% PV	R_{21} : - 14/80 KPI violated: 14 budget KPIs - 6/60 guarantee violated: 6 complyMaxP	R_{22} : - 13/80 KPI violated: 13 KPI budget - 9/60 guarantee violated: 6 complyMaxP, 3 complyMaxQ	R_{23} : - 20/80 KPI violated: 20 KPI green - 7/60 guarantee violated: 6 complyMaxP, 1 complyMaxQ
3: 100% PV	R_{31} : - 14/80 KPIs violated: 14 budget KPIs - 10/60 guarantees violated: 10 complyMaxP, 4 complyMaxQ	R_{32} : - 18/80 KPI violated: 18 KPI budget - 13/60 guarantee violated: 10 complyMaxP, 3 complyMaxQ	R_{33} : - 0/80 KPI violated - 10/60 guarantee violated: 10 complyMaxP

Table 4.3: Summary of the simulation results of the consumers verification model

Profile 1: Budget-conscious consumers

- The simulation results R_{11} shows that all the KPIs and guarantees are satisfied, which is consistent with our expectations as the setpoint temperature is low and the consumers do not aim to have either green or local energy. Figure 4-50 shows an example of simulation results of the sub-contract guarantees for one specific consumer. In this figure, we can identify that at the end of the year, the three guarantees become true (and are therefore satisfied).
- The KPIs violation observed in results R_{21} can be explained by the fact that the NeighbourPower offer proposed to consumers is not viable for them and lead to higher cost, which is not compliant with their goals. However, when zooming into the results

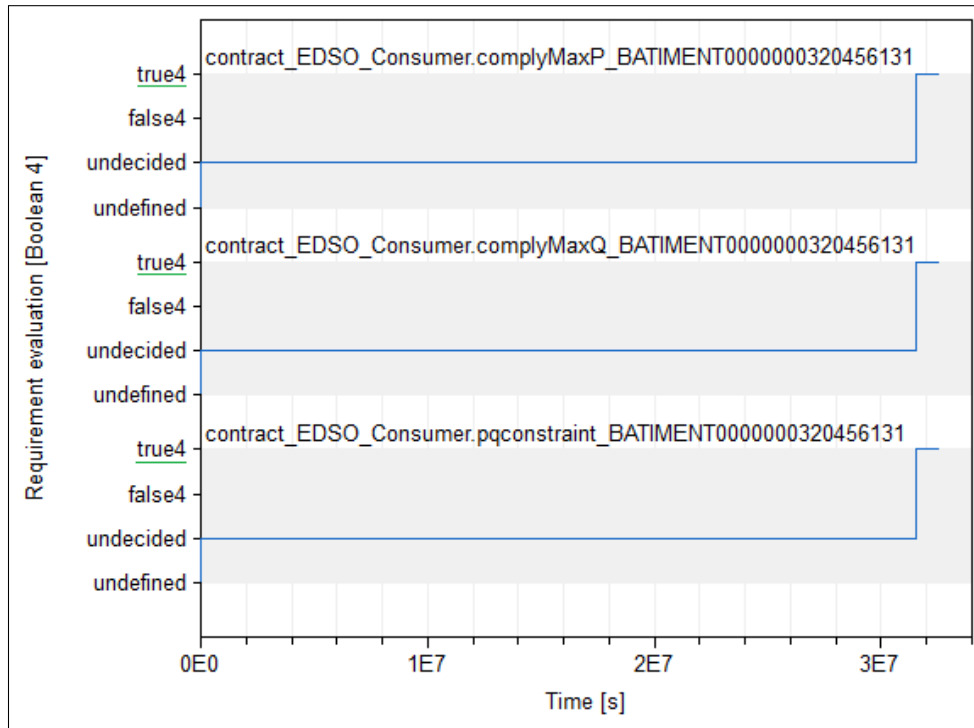


Figure 4-50: Simulation results of one specific consumer guarantees

and especially the violated KPIs, a contradiction has emerged and made us question our models and check the correctness of our behavioral models. Indeed, in the 50% PV scenario, 12 buildings were considered to be equipped with PV panels, which means that at least the KPIs of 8 consumers will not change compared to the first scenario. In contrast, the budget KPI of 14 consumers were violated. By analyzing the buildings behavioral model of the 50% PV scenario, we noticed it was made of two separate and non-communicating modules, one for the buildings equipped with PV panel and another one for the one with no PV panel, unlike the behavioral model used for the previous scenario where all buildings are gathered in one module. The separation of the two modules induced a loss of information related to the masking and reflection phenomena between the PV-equipped and non-equipped buildings, which leads to slightly higher energy consumption for some buildings even if their behavior has not undergone any change. This explains the small difference in terms of the consumers' bills at the end of the simulation which has induced the KPIs violation. The lesson learnt here is that the formal implementation of requirements into the design process allows to carry out reflections on the consistency of the models used during the design of a ME-CPS. Figure 4-51 shows an example of KPIs simulation results for one specific consumer with a property violation as highlighted with a red line.

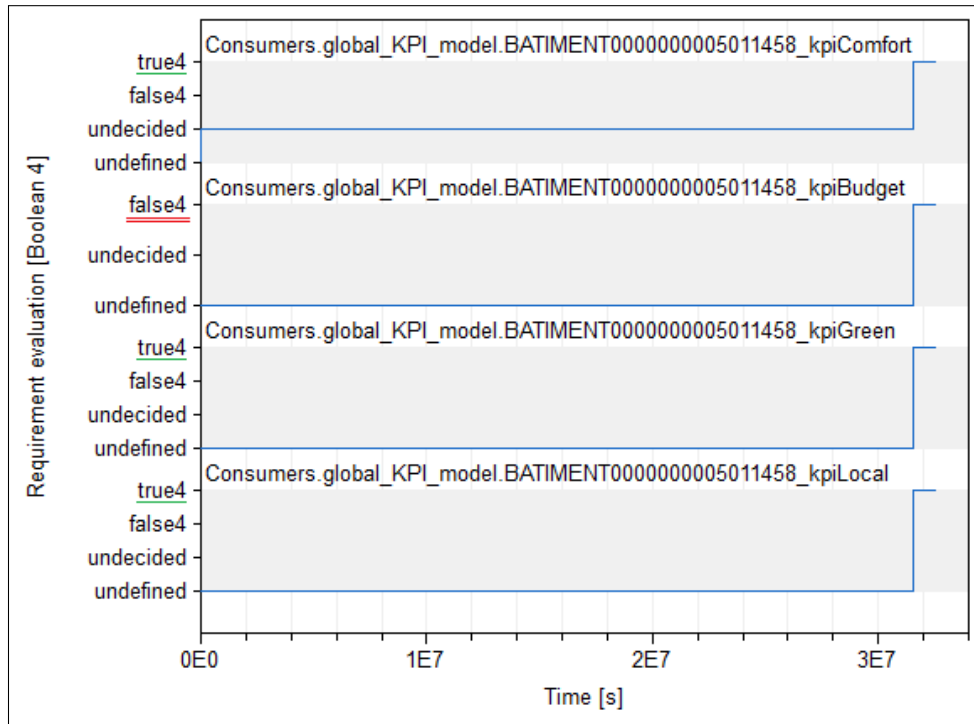


Figure 4-51: Simulation results of the KPIs of one specific consumer

The error that was induced by the separation of the two models will not have an impact on the other analysis given the small difference observed when comparing the part of the model of PV-equipped buildings with the model used for scenario 1, and the model part used for the non-equipped buildings with the model used for scenario 3.

The simulation results R_{21} also show that some of the PV-equipped buildings have violated the maximum subscribed active and reactive powers. This violation is explained by sharp PV production peaks that lead to maximum threshold overruns as displayed in Figure 4-52.

- This simulation results R_{31} show on the one hand that 14 budget-conscious consumers will not be interested in the NeighbourPower offer as their yearly bill will be higher than the minimum budget. On the other hand, 8 consumers may be attracted to the offer, as it will allow them to satisfy their main goal concerning their budget while having green and local energy. Furthermore, their bill may be reduced compared to a standard offer. These consumers have typically premises distributed over vast areas allowing them to produce much more than what they consume and therefore benefit from the revenue of selling the electricity surplus. The simulation results R_{31} also show that more guarantees violations are observed as more consumers are considered to be PV-equipped in this scenario.

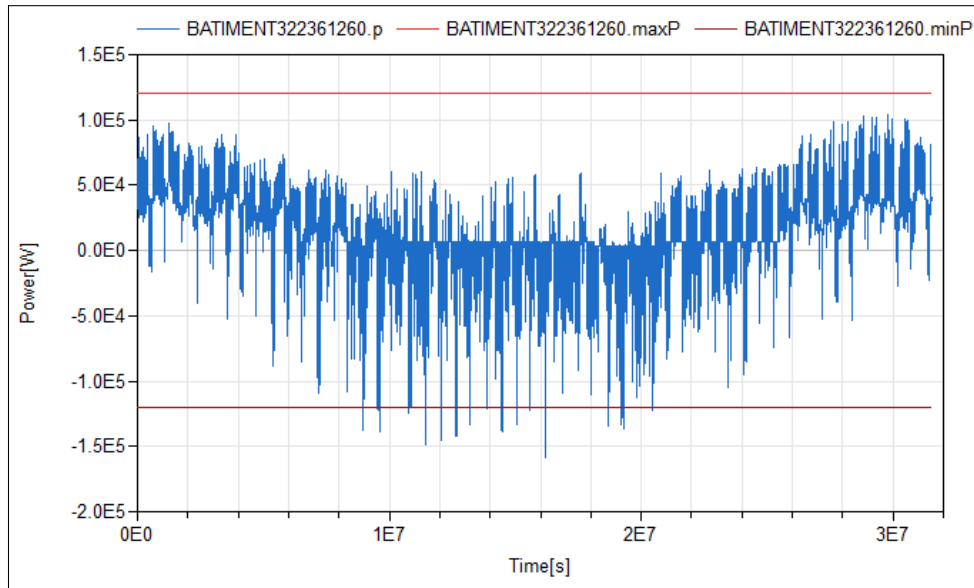


Figure 4-52: Maximum power range overrun

Profile 2: Cold-sensitive consumers

- The simulation results R_{12} show a violation of budget KPIs for 12 consumers. These violations are justified by the fact that the maximum budgets, which are used to set consumers budget targets, were evaluated by considering that over the simulation time, the consumers set their heating system to the maximum comfort conditions while having a relatively expensive supply offer of a green electricity, as mentioned in Section 4.3.5.1. Therefore, the maximum budget is mainly correlated to the costs of electricity and not with the heat energy price. However, amongst the consumers, some use the heat network for their heating, and therefore their energy bills are not much affected by the electricity cost. Consequently, the maximum budgets that were initially computed for the 12 consumers which are connected to the heat grid is close to the budget they had before applying any measures to the PowerGrid. Thus, the 12 KPIs violations are due to the hypothesis taken for the computation of the maximum budget which is not representative of the satisfaction state of the consumers. Two possibilities stand out, either revisiting the computation of the maximum budget of consumers or setting a specific rate (k_b) corresponding to the priority given to budget for the consumers heated by the heat grid.

In regards to the guarantees violation, it is explained by the fact that the maximum active and reactive powers were initially computed for a reasonable heating setpoint temperature of 20 degrees. If the consumers concerned by the violations are actually cold-sensitive, they will have to update their subscription powers.

- The simulation results R_{22} and R_{32} show an increase of the number of “cold-sensitive” consumers that see their budget KPI violated. This means that the NeighbourPower offer does not allow most of them to be economically satisfied with the implementation of PV panels.

These two simulation results also depict more guarantees violations due to PV production peaks.

Profile 3: Green consumers

- The first simulation results R_{13} depicts the violation of all consumers KPIs related to green and local energy (40 KPIs) which was expected as the goal of the renovation is to make the PowerGrid area greener and more self-sufficient. The violations are explained by the fact that the renewable energy rate and the local energy rate are lower than the specified targets of 25% and 50% respectively. The renewable rate computed for electricity and heat production is around 12% which derives from the 20% renewable rate of the electricity imported from the national grid. The local production rate is nearly 40% which results derives from the heat energy produced locally.

No guarantees violations are reported in this scenario.

- The simulation results of the 50% PV scenario show that the consumers’ local energy KPIs were satisfied as it has reached 51% of the total consumed energy. However, the green energy KPIs are still violated in this scenario as the renewable energy rate has only reached 22%.

In regards to the contract guarantees violations, a similar conclusion can be drawn as the two previous profiles, the more buildings get equipped with PV panels, the more guarantees regarding the maximum powers are reported due to PV production peaks.

- The last simulation results R_{33} show that this scenario allows the green consumers to attain all their KPIs. The renewables share have reached 27% of the total consumed energy and the self-sufficiency 58% with are sufficient to satisfy consumers KPIs, in addition to the budget and comfort KPIs that are also guaranteed.

As expected, more guarantees were violated compared to the previous scenario which is due to PV production peaks.

Common observations and conclusions:

- The consumers’ comfort requirements were always satisfied as we have taken the hypothesis that the electric grid constantly satisfies its obligations regardless of the considered scenario. Therefore, the temperature at consumers’ premises will always fit

with the setpoint that they have fixed. These KPIs will be mainly challenged when new electricity services will be studied for the scenarios where the earlier assumption does not hold due to intermittent energy. These services will include smarter monitoring strategies between the electric grid and the buildings with possible instantaneous cut-offs, load shifting, or energy storage in order to avoid maximum power thresholds overruns due to production or consumption peaks.

- The violation of the guarantees concerning the maximum active and reactive powers requires consumers to revisit their subscription contracts with their supplier or otherwise they will get fines or will have electricity cut-offs when exceeding the subscribed powers.
- The simulation results of the budget-conscious consumers' profile have shown that most of them will not be interested by the subscription to the supplier PV offer because it will lead to an increase of their bill and also to a violation of their contracts regarding the E-DSO due to PV production peaks. However, among the budget-conscious consumers some who will save money by subscribing to the offer. These consumers have typically premises distributed over vast areas allowing them to produce much more than what they consume and therefore benefit from the revenue of selling the electricity surplus. Nevertheless, these consumers will have to negotiate their old contract with the E-DSO in order to re-evaluate the maximum active and reactive powers that can go through the connection points with the electric grid. This can lead to an increase of the electricity subscription fees which should be compared to their gain in order to set their opinion regarding the NeighbourPower PV offer.
- The simulation results of the cold-sensitive consumers' profile have shown that almost all of them will not be interested by the NeighbourPower PV offer as their budget KPIs were violated. These results have also allowed us realize that the maximum budgets that were set are not really meaningful for the consumers connected to the heat grid and therefore need to be re-evaluated.
- The simulation results of the green consumers' profile have shown that the scenario 100% PV allows them to satisfy all their requirements. The results also show that the 50% scenario is sufficient to guarantee the consumers local KPIs.

In this case study, simplifying hypothesis were made on the costs of electricity provided from the PV panels installed on the roofs of the PowerGrid buildings, and the results are exclusively meaningful under these hypothesis. In fact, the costs of the initial investments to be made for purchasing and installing the PV panels as well as their depreciation were not considered in details in this analysis. They were reduced to an increase in the cost of a

kWh of electricity. In order to accurately assess the ability of PV panel to reduce consumers' bills (as shown in the first verification models), a finer economical model should be used for consumers KPIs evaluation. The power of CReMA methodology and the design by contracts is the capacity to switch from rough model to more refined ones without having to operate consequent modifications on the observation models and the requirement models.

4.3.10.4 Verification model from the E-DSO point of view

In contrast to the consumers verification model that makes abstraction of the distribution grid behavior as it is considered as being always stable, the E-DSO does represent the consumers behavioral model in order to evaluate the load flow passing through the network. Therefore, the E-DSO verification model is established by the integration of the five following elements:

- The EDSO description model including his KPIs, architecture description model and behavioral model.
- The consumers model represented from the E-DSO point of view. In this analysis, we have integrated the full consumers model as it exhibits reasonable simulation time. An abstract representation of their behavior such as consumption load profiles can be sufficient for the analysis conducted by the E-DSO.
- The E-DSO sub-contract in regards to the consumers exclusively including his guarantees. The definition of the E-DSO assumptions taken on consumers behavior are considered to be equivalent to the consumers guarantees. Therefore, in order to avoid redundancy with the verification of consumers guarantees made previously, we have restricted the distributor sub-contract to its guarantees.
- The bindings linking the three previous elements.
- The test scenario model including the different parameters' configurations that characterize the scenarios to be simulated.

Figure 4-53 depicts an overview of the verification model.

4.3.10.5 Simulation results of the E-DSO verification model

Simulation hypothesis :

- The simulations of verification models are made over a duration that is slightly higher than one year (1 year and 10 days) because the requirements were set for a one year period. Therefore, the decision on whether a requirement is satisfied or not can only be made after that period.

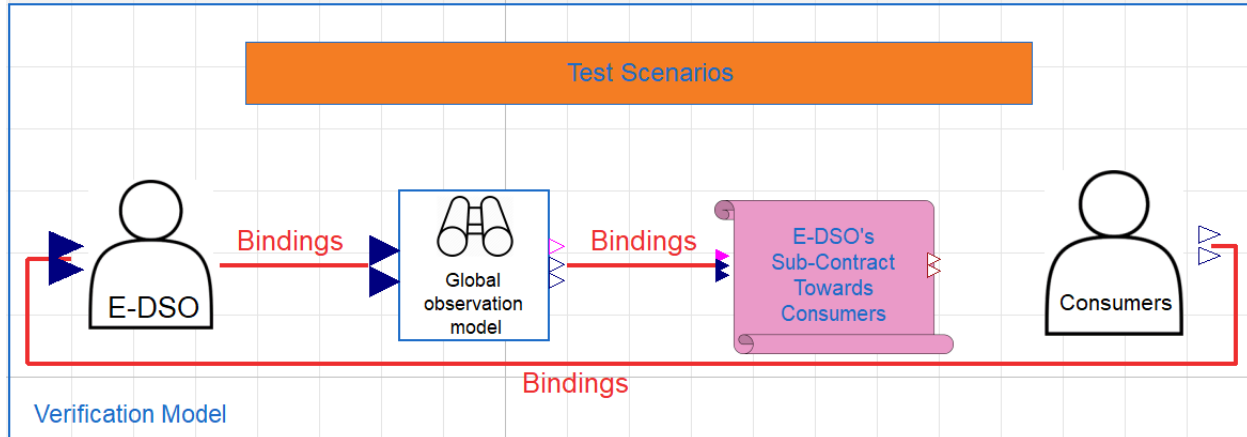


Figure 4-53: EDSO's verification model

- The E-DSO considers a medium consumers profile with a heating setpoint at $kt = 20^{\circ}C$.

Simulation results: Table 4.4 gathers the results of the three simulations of the E-DSO verification model under the three scenarios 0% PV, 50% PV and 100% PV. This table emphasizes the number of violated KPIs and guarantees of each simulation and depicts the types of the requirements that are violated.

Scenarios	Consumers ($kt=20^{\circ}C$)
1 : 0% PV	R_1: - 0/52 KPI violated - 0/40 guarantee violated
2: 50% PV	R_2: - 0/52 KPI violated - 0/40 guarantee violated
3: 100% PV	R_3: - 6/52 KPIs violated: 6 absoluteLimit - 6/40 guarantees violated: 6 absoluteLimit

Table 4.4: Summary of the simulation results of the E-DSO verification model

- The simulation results of scenario 1 representing the PowerGrid area with no specific renovation measures show that all the E-DSO KPIs are satisfied as well as all his

contract guarantees. These results seem consistent with reality and reflect the current state of the PowerGrid zone where only rare buildings are equipped with solar panels and where the electric network answers its requirements at the delivery points.

- The simulation results of the second scenario show that equipping the 12 selected buildings of the PowerGrid area with solar panels does not affect the proper functioning of the electric grid. All of the E-DSO guarantees and KPIs were satisfied.
- The last simulation results of the third scenario depict 6 KPIs and 6 guarantees violations. All of the 12 requirements violations concern electric delivery points to consumers. It is reminded that the E-DSO KPIs include similar requirements as the contract guarantee but applied on a larger scale (all the grid instead of the delivery points). The requirements violations mean that for the 100% PV scenario, the electric network is not able to maintain the electric voltage within the permissible operating range at the 6 nodes. A more thorough analysis of the results has shown that the voltage thresholds overruns are due to instantaneous PV production peaks which drive the voltage above the maximum requirements limits as displayed in Figure 4-54

Even if the voltage maximum limitations were violated, the average satisfaction requirements remained fully covered as the thresholds overruns were only punctual and have never exceeded more than 5% of the time averaged over a sliding period of one week during the simulation time.

Conclusions: Simulation results of the E-DSO verification model have shown that equipping the 20 consumers with PV panels leads to properties violations in terms of E-DSO KPIs and sub-contract. These violations occur mainly because the consumers did not commit to their obligations by exceeding the maximum specified active and reactive powers. In order to establish a new agreement that satisfies the requirements of both the consumers who wish to install solar panels and the E-DSO, they need to reconsider the old contracts established between them and re-visit their terms.

To mitigate these guarantees violations, the E-DSO can consider different options:

- Set up a new agreement for “small-producers” with adapted thresholds for the quantity of electricity that can be injected into the grid, especially if the majority of the consumers intend to equip their buildings with solar panels.
- Set up a limited number of possible agreements for “small-producers” allowing consumers to inject all their electricity surplus into the electric network, especially in the

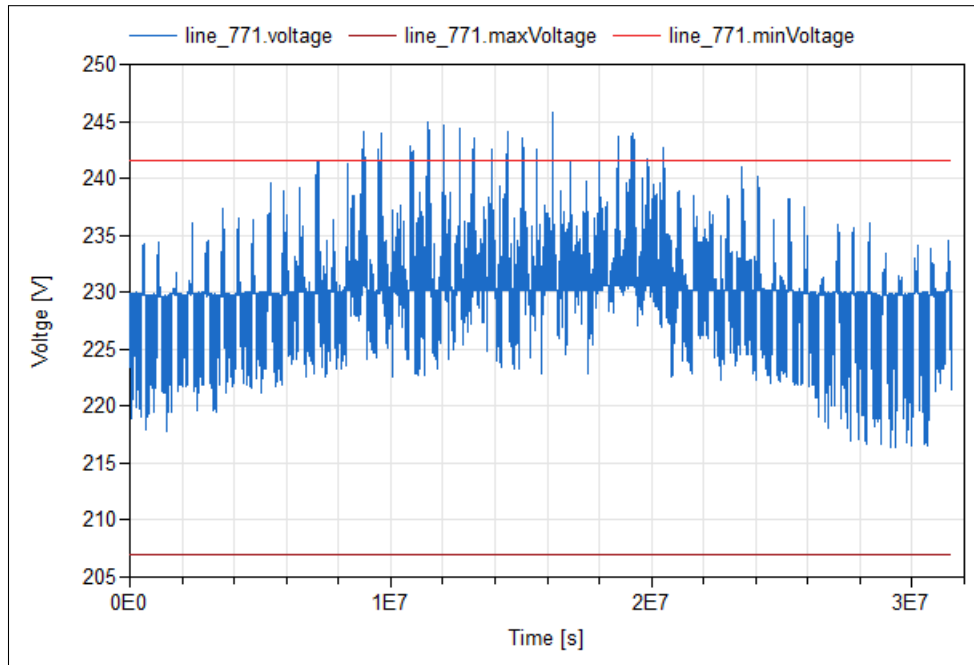


Figure 4-54: Maximum voltage range overrun

access points where the voltage thresholds overruns were observed. This option corresponds to the 50% PV scenario where half of the consumers were equipped with solar panels and all the E-DSO KPIs and guarantees were satisfied.

- Consider a smarter grid management system with possible instantaneous local electricity cut-offs when the voltage at the grid nodes is close to the requirements limits. This possibility is under analysis at the time of writing this thesis. The objective is to find the appropriate management strategy allowing to limit voltage overruns by instantaneously taking the control of the heating (or cooling) systems at consumers premises in order to turn them on or off and thus act on the power injected or drawn from the network. This option will require the approval of consumers and thus new obligations can emerge from both side during negotiations.
- Consider the implementation of storage units that act at specific time periods where the electric network undergoes production or consumption peaks in order to relieve the pressure and make the loads smoother.
- Invest in new safety mechanisms to better regulate the voltage.

Among these options (and maybe others), the E-DSO will make the choice that meets his KPIs (financial and safety) while ensuring the guarantees towards all the interacting stakeholders notably the consumers, the producers and the national grid. Besides, if this choice

induces a modification of the existing contract of the E-DSO with the other stakeholders, negotiations should take place between them in order to identify new agreements serving the common interests.

4.4 Summary

The application of CReMA methodology on the PowerGrid renovation have resulted in numerous achievements. Firstly, it allowed every stakeholder involved in the PowerGrid project to capture and structure its high-level goals and then progressively derive the dependencies regarding other parties and establish formal contracts with them. This phase has led to multiple discussions and debates within the project team for the formal definition of the properties characterising the stakeholders' goals and the contracts' assumptions and guarantees. These discussions were valuable for understanding the roles of the historic stakeholders operating the different elements of the neighbourhood energy system. It was also valuable for the rigorous characterization of the interfaces between stakeholders via the formal contracts and for coordinating the different modeling engineers. Secondly, feedback from the different engineers show that the "stakeholder view" of CReMA methodology including the intentional level, the stakeholders interaction level and the formal requirements level have encouraged them to take a step back and consider the design challenge from a goal-oriented point of view instead of the conventional practice where system objectives do not get much importance. Thirdly, although the design justification level was not applied during the PowerGrid design process, yet the chosen examples have shown its pertinence for capitalizing knowledge on the design argumentation chain, which is in general largely lost by the time the designers are no longer part of the project. Finally, the construction of the verification models has proven its modularity by giving stakeholders the flexibility to use models with different granularity resolutions depending on the knowledge they have at a certain phase of the design and also depending on what level of detail is required for their analysis. Moreover, the results obtained from the simulations of the verification models, especially when requirements violations occur, have led the project members to carry out reflections and seek for the rationales behind the evaluations by questioning the correctness of the requirements formulation, the relevance of the behavioral models and the hypothesis taken on the environment, and finally the fulfilment of the requirement. This process had valuable outcomes as some modeling errors were identified in terms of requirements as well as behavioral modeling.

To give a quick overview of the results obtained within the PowerGrid project, the simulation of multiple scenarios has revealed disagreements between stakeholders, and show how they can be mitigated to find mutual agreements. The simulation results have shown that the chosen scenarios have different impacts on the satisfaction of stakeholders KPIs and re-

quirements. The main challenge remains in the fact that there are no scenario allowing the satisfaction of all consumers profiles as well as the EDSO. A potential solution is the introduction of monitoring mechanisms for a smart management of the electricity consumption of buildings depending on the state of the electric grid. Works are currently being carried to make this implementation into the PowerGrid project to make it smarter. We stress that the simulation results are exclusively meaningful under the hypothesis taken for each verification model.

Lessons learnt from this experiment are that setting ambitious KPIs on ME-CPS and environment friendliness is not an easy task and requires careful cooperation between actors. Technical limitations often appear after thorough analysis and it is only then when stakeholders and more specifically the project initiators realize that their objectives are not feasible at reasonable costs.

Chapter 5

Conclusion and future works

In this conclusion, we first set up the state of at the beginning of this thesis by introducing a global overview of the context, the challenges that will be addressed, and the existing works related to the subject we dealt with. Subsequently, we expose the major contributions of this thesis in regard to our scientific challenges. In the same register, we highlight the main limitations and the perspectives of our methodology. Finally, the results of the PowerGrid case study will be discussed, and future applications are presented.

5.1 Overview

Until recently, energy systems were run by centralized production units that were sized and operated to meet the national energy demand at all times. This paradigm was appropriate in terms of economic, technical, and management aspects. However, it is no longer adapted to the new energy landscape with the growing awareness of the environmental footprint and the deregulation of the energy sector at a European scale. On the one hand, this has led to an increasing number of active actors on the energy grid having different objectives and perspectives in mind. On the other hand, the evolving landscape sketches an architecture with more and more decentralized units and independent management systems, which requires increasing communication, control, and information technologies into the existing physical world. The future challenge is to have “Smarter” systems that guarantee more flexibility, efficiency, reliability, and security. The challenge of the actors in the energy sector is to develop technologies with the right integration between the cyber and the physical dimensions that represents the core of the complexity of ME-CPS.

To fit with the new energy context, this thesis needs to introduce a new rigorous methodology for the coordination of multiple stakeholders involved in the design of Multi-Energy Cyber-Physical-Systems (ME-CPS) assuming that they will be correctly designed and operated only if all the stakeholders reach mutual agreements that satisfy their intrinsic goals.

In order to reach this objective, we started by identifying the specific characteristics of Multi-Energy-Systems (MES) and their evolution in the current shifting energy landscape. MES are developed to operate continuously over long periods of time and are therefore constantly subject to modifications and improvements at different stages of their lifecycle. They are also “open” systems that are subject to the uncertainties of their natural, political, societal, etc. environments which cannot be controlled or to a very low degree. Moreover, MES include components with high safety stakes for which a rigorous demonstration of the correct functioning of safety-critical elements all along their lifecycle is as important as achieving the functioning itself. Getting a system authorized to run is conditioned to the capacity of the developer to provide rigorous justifications that safety requirements are met.

To address the challenges facing the design of ME-CPS, the rigorous methodology proposed in this thesis shall be based on the formal modeling of requirements and system behaviors that can be used for verification through simulations all along the system lifecycle. The idea is assisting stakeholders throughout negotiations with means enabling them to verify, on the one hand, that the contracts they are negotiating answer their objectives, and on the other hand, their ability to commit to their obligations in regard to the other stakeholders. In addition, in order to fit with the dynamic changing context of ME-CPS, this framework is required to be modular and allow stakeholders to easily adjust their models depending on the desired level of granularity of the analysis.

Before proceeding to the development of our co-design methodology, we have started by exploring existing works from the literature dealing with the previous scientific challenges.

Firstly, in the state of the art, the notions of stakeholders and the capture of stakeholders’ requirements are strongly highlighted as being critical elements for the successful development of complex systems. However, a striking similarity was observed in the majority of the works dealing with these notions: Stakeholders are always considered from a single designer or analyst point of view which considers them as sources of requirements established at the preliminary studies, and not as active actors during the whole development process. These requirements are therefore subjective to the designer’s perspective and can be in contradiction with the real needs of the actual stakeholders.

Secondly, a common limitation concerning Systems Engineering (SE) methodologies that deal with the design of complex CPS was identified. A wide gap was observed between the early design phases when systems objectives and requirements are defined using informal or semi-formal modeling paradigms, and the more detailed design phases when engineering teams use domain-specific tools for behavioral modeling. These latter tools are based on formal modeling paradigms and are thus hardly connected to the informal ones containing requirements. As a result, V&V activities cannot be performed automatically and require a manual check to map requirements to formal behavioral models. Some state of the art works

have tried to bridge this gap but only achieved a mitigated success, which we suppose is due to the lack of formal tools for requirement engineering and also a lack of methodological guidelines connecting the techniques used in early design phases to those used in detailed design phases.

To bridge this gap, we need to formally express requirements and among the few identified paradigms enabling the capture of formal requirements, we have chosen FORM-L language. We also needed to formalize the interaction between stakeholders, and the design based on “Formal contracts” seemed a good candidate for structuring the relationships and interfaces between stakeholders. These two elements are the pillars for the new co-design methodology developed in the thesis named: Common Requirement engineering Modeling Approach (CReMA).

5.2 Contributions

The direction in which we conducted our research was driven by the observations made during the state of the art analysis on stakeholders’ coordination, Systems Engineering (SE) methodologies, and the formal approaches dealing with requirements and Verification and Validation (V&V).

The novelty of CReMA methodology concerns the introduction of a **decentralized** vision for the stakeholders’ coordination problem compared to the classical vision which was generally centered on a single analyst or designer. In the new framework, every stakeholder is considered to have his own point of view with specific goals, and gradually builds relationships with other stakeholders. These relationships emerge throughout an iterative process where discussions and negotiations take place between the stakeholders. They are structured and formalized through the different levels of CReMA methodology.

CReMA methodology is built upon five modeling levels separated into two main views: the Stakeholder view and the System view. An additional level called the design justification level spreads over the two views and serves as a bridge that ensures the traceability between the stakeholder view and the system view of CReMA methodology.

The “Stakeholder view” focuses on capturing what we refer to as “social aspects” by the means of three levels: (i) the stakeholder intentional level that captures the stakeholders’ high-level goals and the dependencies between them (ii) the stakeholders’ interaction level that describes the interactions between stakeholders such as financial exchange, physical exchange or data exchange, and (iii) the formal requirement level that formalizes, on the one hand, the stakeholders’ high-level goals giving rise to what is called Key Performance Indicators (KPIs) and, on the other hand, the requirements related to the relationships between them which gives rise to formal contracts. The main asset of the “Stakeholder view”

is giving stakeholders the means to elicit their requirements towards each other and identify potential relationships by starting from their high-level interests and progressively refining them down into atomic needs.

The “System view” focuses on capturing the design aspects of the systems under the responsibility of each stakeholder, going from the preliminary studies until detailed system design. Two methodology levels synthesize these design aspects through (i) the “Architecture description level” that has the particularity of describing the static engineering knowledge concerning the design solutions, and (ii) the “Behavioral level” that represents the dynamic behavior of these solutions. Both levels are necessary for the formal V&V of systems. These two levels do not constrain designers to use a specific tool or language, and are adapted to the dynamic context of the constant changes and improvements undergone by models throughout the system lifecycle.

Having the “Stakeholder view” and the “System view” at hand, modular verification models can be built by stakeholders in order to evaluate the satisfaction of their goals and their commitment to their obligations. The use of contracts for formalizing the relationships and interfaces between stakeholder plays a major role in having the flexibility to build “on-demand” verification models. Indeed, the contracts allow each stakeholder to define assumptions that represent behavioral envelopes of their external environment (involving the other stakeholders). Thus, each stakeholder can undertake his own verification on his system separately, under the reasonable assumption that the behaviors of the other systems remain within the scope of the contracts.

Also, the introduction of the notion of negotiations between stakeholders in the development process allows detaching designers from the classic vision of the rigid system requirements that are defined at early design phases and are not revisited until very late in the development cycle if no design solution is found and where modifications are very costly. Negotiations give way to compromises between parties, which serve the common interest.

In contrast to the state-of-the-art methodologies, CReMA has allowed bridging the gap between the early conceptual studies where stakeholders define their high-level objectives and requirements, and the more detailed design phases where stakeholders, supported by engineering teams, develop behavioral models using domain-specific tools. This challenge was overcome using two main elements : (i) the methodological guidelines that brought together the “Stakeholder view” dealing mainly with early design phases and the “System view” which is more centered on detailed design, and (ii) the rigorous techniques supporting the formal capture of the early design phases outcomes which are mainly the stakeholders’ goals and requirements, giving rise to KPIs and formal contracts. These formal models are interoperable with domain-specific tools through the use of the observation operators and bindings. They are non-intrusive modeling concepts that are used to extract pieces of

information from behavioral and architecture description models. These pieces of information are then transformed into variables and functional states that are compatible with the inputs of requirement models, allowing executable V&V.

All the concepts that are used in CReMA and the relationships between them are structured using a unified meta-model. It is used as a “pivot” model from which are allocated the concepts to the different levels of CReMA while making sure that all these concepts are implemented consistently, with no redundancy, and with no oversight of a key item of the methodology.

Finally, we have introduced a theoretical contract formalism characterizing the various concepts of CReMA methodology such as contracts, KPIs, environment, and implementation, etc. as well as the logical operations that can be applied to these objects throughout the design process. The ultimate goal of this theoretical formalism is to rigorously define the key operations that can be carried out by stakeholders throughout the negotiation process and during the validation of their agreements. The aim is to be able to systematically evaluate these operations through simulation. To name some operations, we can cite the stakeholders’ goals satisfaction, the stakeholders’ agreement to sign a contract, and the consistency of a contract refinement. The introduced formalism was defined to be independent of any tool, language, or modeling approach. This formalism is a necessary foundation for the development of a tooling framework supporting the design by contracts of CReMA methodology.

Up to now, CReMA methodology levels are mainly prototyped using Modelica, except the intentional level developed using the i^* framework and the design justification level using CAE framework. This choice was made for the simple reason that the only available implementation of FORM-L language (the language chosen for formal requirement capture in CReMA) is made possible through the ReqSysPro Modelica library [61]. The other toolchain elements pushing the scope beyond Modelica are under development within the international project EMBRrACE [21].

5.2.1 Limits

CReMA methodology was built under the critical hypothesis that all stakeholders are willing to negotiate and are open to making concessions in order to reach common agreements. This condition might not always be satisfied in industrial projects involving multiple stakeholders, especially when they come from more than one entity. The first challenge lies in the willingness of the different stakeholders to take part in the application of CReMA methodology and agree to make negotiations and concessions. In general, the adoption of new approaches by designers and people in a broader sense is not an easy task which requires careful attention to prove the relevance of its implementation and its easiness to be mastered by its users. The second challenge is that stakeholders should have the minimum technical background

required to be able to conduct a scientific study involving formal methods for requirement engineering and behavioral modeling. The main problem is that the use of formal methods tends to repel engineering teams that are not familiar with the concept, especially when dealing with requirement engineering. Thirdly, not all stakeholders are willing to conduct or invest in these scientific studies because they do not see their necessity for their analysis.

The benefit of CReMA methodology remains in its modularity that allows each stakeholder to carry out the study he wishes to conduct as long as he has a formal contract with other stakeholders in which the interfaces and mutual obligations are rigorously defined. Therefore, the application of CReMA methodology can tolerate a mix between stakeholders with different opinions on its implementation, yet the negotiations may be less relevant.

5.3 Methodology application

This methodology was applied in the framework of an industrial research program called PowerGrid at EDF Lab with the aim to coordinate stakeholders for the renovation of a local multi-energy district of Vélizy which is a city located in the Paris suburban area. The objective of the project is to make it “greener” and “smarter”, greener because it is intended to involve more renewable energy, smarter because it is intended to have a more efficient energy management system.

The expectations of this application were met. First, the “Stakeholder view” levels have allowed stakeholders to set their goals, and from there, progressively elicit their requirements towards others. The discussions that took place in this phase within the project team has allowed a better understanding of the context and also to define the roles and scopes of each stakeholder. Positive feedback from engineers highlighted the potential of the “Stakeholder view” for encouraging them to take a step back and consider the design challenge from a goal-oriented point of view rather than the conventional practice where system objectives do not get much importance.

Secondly, the “System view” levels have also allowed stakeholders to set a clear architecture description of their system, delimit the scopes of each one of them, and develop behavioral models with the tools they are used to work with.

Subsequently, the design justification framework has shown its pertinence for capitalizing knowledge on the design argumentation chain, which is in general largely lost by the time the designers are no longer part of the project.

In addition, CReMA methodology has proven its modularity by giving stakeholders the flexibility to build “on-demand” verification models that can support behavioral models with different granularity resolutions depending on the design phase and the required level of detail of the analysis.

Finally, analyses carried out on the requirements' simulation results have led the project members to conduct various reflections revealing the rationale behind these results. They have questioned the correctness of the requirements formulation, the relevance of the behavioral models, the relevance of the hypotheses taken on the environment, and finally the fulfillment of the requirements. This process had valuable outcomes as some errors were identified in terms of requirements and behavioral modeling. Interesting conclusions were also drawn on the satisfaction of stakeholders' goals at every simulated scenario.

Lessons learned from this experiment are that setting ambitious KPIs on ME-CPS and environmental friendliness is not an easy task and requires careful cooperation between actors. Technical limitations often appear after thorough analysis and it is only then that stakeholders and more specifically the project initiators realize that their objectives are not feasible at reasonable costs.

5.4 Future works

5.4.1 Potential enhancements concerning CReMA methodology

The development of CReMA methodology is part of a research program that has started years before this thesis, which in turn has opened the way for new perspectives. The tooling aspects of CReMA methodology represents one of the major challenges for the stakeholders' coordination especially when various domain-specific tools are used. Therefore, the next step essentially concerns the improvement of the formal requirement modeling language FORM-L to be interoperable with the key disciplines that are involved in the design of large ME-CPS. The idea is to develop compilers allowing the automatic transformation of requirements written using FORM-L language into requirement models that are compatible with "target" disciplinary tools, thus allowing formal testing of requirements.

Still in line with the requirement verification, a major enhancement of the methodology features can be made through the implementation of a test sequence generator that will allow designers to verify the correct establishment of their requirements and reveal potential contradictions before proceeding to the design of the system answering them. This generator will be used to automatically produce test sequences that are compliant with the assumptions (behavioral trajectories considered to be always satisfied) taken by stakeholders on their system and also on their environment. These test sequences are later substituted by behavioral models in the detailed design phases providing realistic values for physical states of the system.

These two outlooks are currently under development in the framework of the international project EMBRrACE[21].

To go one step further, a high-potential track concerns the development of a common platform gathering the different levels of CReMA methodology, while being interoperable with existing domain-specific tools used by designers. This platform should include the following elements:

- The i^* framework that was used for capturing stakeholders' goals and dependencies in the stakeholders' intentional level. Some adaptations are necessary to fit with the terminology used in CReMA.
- A specific tool allowing graphical architectures description while having the capacity to store static engineering data and communicate it when necessary. This tool will be dedicated to two methodology levels: the stakeholders' interaction level and the architecture description level.
- The formal requirement language FORM-L while being equipped with compilers and automatic test sequence generator.
- A co-simulation platform enabling to bring together the different components of a verification model for making automatic testing.

The contracts formalism introduced in this thesis is the first foundation of the automation process of key operations for stakeholders' coordination, which can be subject to further development. The next recommended steps are the following:

- Setting up applications for every definition of the contracts' formalism that were not tested yet such as the contracts conjunction and contracts' refinement.
- Optimizing the definition of "contracts conjunction" to reduce redundant constraints between assumptions and guarantees.
- Implementing the formalism into CReMA methodology framework and make adaptations to the formalism if necessary.

5.4.2 Potential enhancements concerning the CReMA methodology application

The next step of the PowerGrid case study concerns the implementation of "smart" monitoring strategies that involve multiple stakeholders in order to optimize their goals and their formal contracts' satisfaction.

A promising route concerns the formal contract between the PowerGrid prosumers (i.e. producers and consumers at the same time) and the Electricity Distribution Network Operator (E-DSO) by setting up a grid management strategy with possible instantaneous electricity cut-offs for the prosumers when the voltage at the electric grid nodes is close to the requirements limits.

A second promising monitoring strategy under study concerns two formal contracts simultaneously: the contract between the consumers and the Heat Distribution System Operator (H-DSO), and the contract between the H-DSO and the local producers. The idea is to set up a price incentive for consumers to encourage them to lower the temperature of the water returned to the heat network, which will allow the producers to have a more efficient condensation of the steam, leading to a considerable raise in the heat production efficiency. This scenario requires using detailed physical and financial models for a relevant evaluation of the goals and requirements of the three stakeholders, namely the H-DSO, the local producers, and the consumers.

In parallel to the PowerGrid case study, a future application of CReMA methodology to identify new services to make energy savings at a European scale is under consideration. This is part of a recently submitted European project named “NEWTON” (NEXt Wave of smarT On-demand eNergy services based on efficiency and flexibility)

Beyond the energy sector, the innovative perspective introduced in this work can be applied to the design of all Systems of Systems and CPS inherently involving a large number of stakeholders (e.g. Industry 4.0, networks of any kind such as railway and aeronautics, the autonomous vehicle in its very complex context, the urban planning especially for smart districts, defense systems, etc.). In addition, the methodology can be of valuable added value for systems having a moving context with lot of uncertainties and where the scope of the system is progressively delimited throughout the design.

5.5 In a nutshell

The CReMA provides a more complete and realistic understanding of complex systems: More complete because all viewpoints are considered. More realistic because the behavior of the systems under consideration are modeled, simulated and compared to the expectations of all stakeholders. This helps design complex systems that fulfill better the missions within the prescribed resource envelop. This has been tested successfully on the PowerGrid. Lessons learned from that experiment is that setting ambitious KPIs on a urban district are difficult to fulfill without proper coordination between stakeholders. This methodology was validated on an energy test case, and can be extended to other industrial domains.

Bibliography

- [1] Average price of heat for French networks. Available at http://carmen.developpement-durable.gouv.fr/18/competitivite_tarifaire.map.
- [2] Electricity prices of French operators. Available at <https://www.kelwatt.fr/guide/prix-electricite-france>.
- [3] French thermal regulation. Available at <https://www.ademe.fr/expertises/batiment/elements-contexte/politiques-vigueur/reglementation-thermique>.
- [4] Principes d'étude et de développement du réseau pour le raccordement des clients consommateurs et producteurs BT, ID : Enedis-PRO-RES_43E. Available at https://www.enedis.fr/sites/default/files/Enedis-PRO-RES_43E.pdf.
- [5] 20-sim - M&S software package for mechatronic systems. Available at <http://www.20sim.com/>.
- [6] AdvoCATE: Assurance Case Automation Toolset. Available at <https://ti.arc.nasa.gov/tech/rse/research/advocate/>.
- [7] Anylogic - Multimethod simulation modeling tool. Available at <https://www.anylogic.com/>.
- [8] ASCE - the Assurance and Safety Case Environment. Available at <https://www.adelard.com/asce/choosing-asce/index/>.
- [9] Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems. Available at <https://cordis.europa.eu/project/id/100016>.
- [10] D-Case Editor : A Typed Assurance Case Editor. Available at <http://deos.or.jp/technology/D-CaseEditor/>.
- [11] DARPA META Program. Available at <https://cps-vo.org/group/avm/meta>.
- [12] DYMOLA Systems Engineering - Multi-Engineering Modeling and Simulation based on Modelica and FMI. Available at <https://www.3ds.com/products-services/catia/products/dymola/>.
- [13] Efficacity: a research and development centre dedicated to urban energy transition. Available at <https://www.efficacity.com/>.

-
- [14] Eiffel programming language. Available at <https://www.eiffel.com/>.
- [15] Guidance on Claims, Arguments and Evidence. Available at <https://claimsargumentsevidence.org/>.
- [16] Harmonised Assessment of Reliability of MODern Nuclear I&C Software. Available at <http://harmonics.vtt.fi/>.
- [17] Integrated Tool Chain for Model-based Design of Cyber-Physical Systems. Available at <https://projects.au.dk/into-cps/>, also at <https://into-cps.org/>.
- [18] Introduction à l'Ingénierie système. Available at <https://www.fun-mooc.fr/courses/course-v1:parisseine+105002+session01/about>.
- [19] ISO 9000:2015- Quality management systems - Fundamentals and vocabulary. Technical report.
- [20] ITEA 2 MODRIO – Model Driven Physical Systems Operation. Available at = <https://itea3.org/project/modrio.html>.
- [21] ITEA 3 EMBRACE – Environment for model-based rigorous adaptive co-design and operation of CPS. Available at = <https://itea3.org/project/embrace.html>.
- [22] Modelio - open source UML / BPMN modeling tool. Available at <https://www.modelio.org/>.
- [23] Natural Environment. Available at = https://en.wikipedia.org/wiki/Natural_environment.
- [24] OPENMODELICA - an open-source Modelica-based M&S environment. Available at <https://www.openmodelica.org/>.
- [25] Organization Modelling Environment (OME) tool for i* framework. Available at = <http://www.cs.toronto.edu/km/ome/>.
- [26] Overture Tool - Formal Modelling in VDM . Available at <http://overturetool.org/>.
- [27] Overview of Rational DOORS. Available at https://www.ibm.com/support/knowledgecenter/SSYQBZ_9.5.0/com.ibm.doors.requirements.doc/topics/c_welcome.html.
- [28] ReqIF tool. Available at <https://www.omg.org/spec/ReqIF/About-ReqIF/>.
- [29] RT-Tester - a test automation tool for automatic test generation. Available at <https://www.verified.de/products/rt-tester/>.
- [30] SAVONA: Design, Specification Verification of Embedded Systems. Available at <https://www.expleo-germany.com/en/products/savona/>.
- [31] SpaceEX - State Space Explorer . Available at <http://spaceex.imag.fr/>.
- [32] Structured Assurance Case Metamodel (SACM). Available at <https://www.omg.org/spec/SACM/About-SACM/>.

-
- [33] System Driven Product Development. Available at <https://www.plm.automation.siemens.com/global/en/products/collaboration/mbse-model-based-systems-engineering.html>.
- [34] Teamcenter PLM software. Available at <https://www.plm.automation.siemens.com/global/en/products/teamcenter/>.
- [35] UnCoVerCPS - Unifying Control and Verification of Cyber-Physical Systems . Available at <https://cps-vo.org/group/UnCoVerCPS> .
- [36] MIL-STD-499, MILITARY STANDARD: SYSTEM ENGINEERING MANAGEMENT. Technical report, 1969.
- [37] IEEE 1220:1994 Standard for application and Management of the Systems Engineering Process. Technical report, 1994.
- [38] ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering. Technical report, 2011.
- [39] IEEE 1012-2012 - IEEE Standard for System and Software Verification and Validation. Technical report, 2012.
- [40] ISO/IEC/IEEE 29119:2013 Software Testing. Technical report, 2013.
- [41] ISO/IEC 15026-3:2015(E) Systems and software engineering — Systems and software assurance —. Technical report, 2015.
- [42] ISO/IEC/IEEE 15288:2015 - Systems and software engineering – System life cycle processes. Technical report, 2015.
- [43] Bilan prévisionnel de l'équilibre offre-demande d'électricité en France. Technical report, RTE, 2017.
- [44] *Dependability Assessment of Software for Safety Instrumentation and Control Systems at Nuclear Power Plants*. Number NP-T-3.27 in Nuclear Energy Series. INTERNATIONAL ATOMIC ENERGY AGENCY, Vienna, 2018. Available at <https://www.iaea.org/publications/12232/dependability-assessment-of-software-for-safety-instrumentation-and-control-systems-at-nuclear-power-plants>.
- [45] Afis. Découvrir et comprendre l'Ingénierie Système. *Ouvrage collectif préparé par le Groupe de Travail Ingénierie Système, Association Française d'Ingénierie Système*, 2009.
- [46] Adel Aloui, Khaled Saadaoui, and Manal Wehbi Sleiman. Le concept de parties prenantes : proposition d'une modélisation systémique par le modèle SAGACE. *Management & Sciences Sociales*, (19):120–137, 2015.
- [47] E. Azzouzi, D. Bouskela, A. Jardin, Jean-Yves Choley, and Faïda Mhenni. A Survey on Systems Engineering Methodologies for Large Multi-Energy Cyber-Physical Systems. *13th Annual International Systems Conference, SysCon 2019 - Proceedings*, 2019.

-
- [48] E. Azzouzi, D. Bouskela, A. Jardin, Jean-Yves Choley, and Faïda Mhenni. A model-based engineering methodology for stakeholders coordination of multi-energy cyber-physical systems. submitted, 2020.
- [49] Christel Baier and Joost-Pieter Katoen. *Principles Of Model Checking*. 2008.
- [50] Manas Bajaj, Dirk Zwemer, Russell Peak, Alex Phung, Andrew Scott, and Miyako Wilson. Satellites to Supply Chains, Energy to Finance—SLIM for Model-Based Systems Engineering: Part 1: Motivation and Concept of SLIM. In *INCOSE international Symposium*, volume 21, pages 368–394. Wiley Online Library, 2011.
- [51] Sébastien Bardin. Introduction au Model Checking. Technical report, 2008.
- [52] Nicolas Belloir, Jean-Michel Bruel, and Raphaël Faudou. Modélisation des exigences en UML/SysML. *Revue Génie Logiciel*, (111):6–12, December 2014.
- [53] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, and Al. Contracts for Systems Design: Theory. Technical report, INRIA, 2015.
- [54] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. Contracts for system design. *Foundations and Trends® in Electronic Design Automation*, 12(2-3):124–400, 2018.
- [55] P. G. Bishop, R. E. Bloomfield, S. Guerra, and N. Thuy. Safety justification frameworks: Integrating rule-based, goal-based and risk-informed approaches. In *8th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies 2012, NPIC and HMIT 2012: Enabling the Future of Nuclear Energy*, volume 2, pages 1283–1290. American Nuclear Society, Illinois, USA, December 2012.
- [56] BKCASE. *Guide to the Systems Engineering Body of Knowledge (SEBoK)*. 2016.
- [57] R. E. Bloomfield and K. Netkachova. Building blocks for assurance cases.
- [58] Robin Bloomfield. *Invited Talk: Structured Engineering Argumentation*, pages 335–335. 06 2018.
- [59] B W Boehm. Guidelines for Verifying and Validating Software Requirements and Design Specifications. *Euro IFIP 79*, 1979.
- [60] Mathias Bouquerel, Enrique Kremers, Jonathan van der Kamp, Nguyen Thuy, and Audrey Jardin. Requirements modelling to help decision makers to efficiently renovate energy systems of urban districts. In *Proceedings of the 2019 Summer Simulation Conference*, page 44. Society for Computer Simulation International, 2019.
- [61] D Bouskela and A Jardin. ETL: A new temporal language for the verification of cyber-physical systems. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–8, 2018.

-
- [62] D. Bouskela, T. Nguyen, and A. Jardin. *MODRIO Project : Modeling Architecture for the Verification of Requirements - MODRIO deliverable D2.1.1, EDF R&D EDF RD, H-P1C-2014-15188-EN*. 2014.
- [63] D. Bouskela, T. Nguyen, and A. Jardin. Toward a rigorous approach for verifying cyber-physical systems against requirements. *Canadian Journal of Electrical and Computer Engineering*, 2017.
- [64] Davide Bresolin. HyLTL: a temporal logic for model checking hybrid systems. 2013.
- [65] M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 1988.
- [66] Lilian Burdy and Jean-Marc Meynadier. Automatic refinement. *Proceedings of BUGM at FM*, 99, 1999.
- [67] Mathieu Caujolle, Markus ANDRES, Gabriel GAU, Clément COIC, Najj NASSAR, and Victor-Marie LEBRUN. Hybrid ac and dc distribution networks modelling and planning using epsl modelica library : Preliminary results. *CIREN 2019 Conference*, 06 2019.
- [68] Mohammad Chami and Jean Michel Bruel. Towards an integrated conceptual design evaluation of mechatronic systems: The SysDICE approach. *Procedia Computer Science*, 51:650–659, 2015.
- [69] Frederic Chauvin and Gauthier Fanmuy. System engineering on 3DEXPERIENCE platform - UAS use case. In *CEUR Workshop Proceedings*, 2014.
- [70] Jean-Yves Choley, Faïda Mhenni, Nga Nguyen, and Anis Baklouti. Topology-based Safety Analysis for Safety Critical CPS. *Procedia Computer Science*, 95:32–39, 2016.
- [71] Johan Eker, Jörn W. Janneck, Edward A. Lee, Jie Liu, Xiaojun Liu, Jozsef Ludvig, Stephen Neuendorffer, Sonia Sachs, and Yuhong Xiong. Taming heterogeneity - The ptolemy approach. *Proceedings of the IEEE*, 2003.
- [72] Electronic Industries Alliance. EIA STANDARD Processes for Engineering a System, EIA-632. *Journal of Equine Veterinary Science*, 1999.
- [73] S K Eric, Paolo Giorgini, Neil Maiden, and John Mylopoulos. *Social modeling for requirements engineering*. Mit Press, 2011.
- [74] M. Bouquerel et al. *Démonstrateur PowerGrid - Premiers prototypes pour la planification urbaine et le dimensionnement, EDF R&D, 6125-2406-2018-02163-EN*. 2018.
- [75] M. Bouquerel et al. *Démonstrateur PowerGrid - Synthèse et prototypes de co-conception pour aller de la planification urbaine au dimensionnement dynamique et à la conduite, EDF R&D, 6125-2406-2019-03347-EN*. 2019.
- [76] H. Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28, 2010.

-
- [77] Jacques Fleuriot. Formal Verification : Computation Tree Logic (CTL).
- [78] E.R. Freeman. Strategic Management: A Stakeholder Approach. *Pitman Publishing Inc.*, 1984.
- [79] R. Freeman. The stakeholder approach revisited. *Zeitschrift für Wirtschafts- und Unternehmensethik*, 5, 01 2004.
- [80] R. Edward Freeman. A stakeholder theory of the modern corporation. *Perspectives in Business Ethics* Sie, 3:144, 2001.
- [81] Peter Fritzson and Vadim Engelson. Modelica — a unified object-oriented language for system modeling and simulation. In Eric Jul, editor, *ECOOOP'98 — Object-Oriented Programming*, pages 67–90, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
- [82] Markus Grabowski. *Why Templates on System Behavior Are Not Used in Practice Yet: A Proposal for Enhancements, Application and Formalization*. jul 2017.
- [83] Markus Grabowski, Bernhard Kaiser, and Yu Bai. Systematic Refinement of CPS Requirements using SysML, Template Language and Contracts. 2018.
- [84] Thomas A. Henzinger, Pei Hsin Ho, and Howard Wong-Toi. HyTech: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1997.
- [85] Marija Ilic, Le Xie, Usman Khan, and Jose Moura. Modeling of future cyber–physical energy systems for distributed sensing and control. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40:825 – 838, 08 2010.
- [86] INCOSE. Survey of Model-Based Systems Engineering Methodologies. Technical report, 2008.
- [87] ISO/IEC/IEEE. *Systems and software engineering — Architecture description 42010*. 2011.
- [88] Melissa Issad, Leïla Kloul, and Antoine Rauzy. A model-based methodology to formalize specifications of railway systems. In *Model-Based Safety and Assessment*, pages 28–42. Springer, 2014.
- [89] Bertrand Jeannot and Fabien Gaucher. Debugging embedded systems requirements with stimulus: an automotive case-study. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.
- [90] Jeff C. Jensen, Danica H. Chang, and Edward A. Lee. A model-based design methodology for cyber-physical systems. In *2011 7th International Wireless Communications and Mobile Computing Conference*, 2011.
- [91] Manfred A. Jeusfeld. *Metamodel*, pages 1727–1730. Springer US, Boston, MA, 2009.

-
- [92] Bilal Kanso and Safouan Taha. Temporal constraint support for OCL. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.
- [93] Tim Kelly and Rob Weaver. The goal structuring notation—a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases*, page 6. Citeseer, 2004.
- [94] Timothy Patrick Kelly. Arguing Safety – A Systematic Approach to Managing Safety Cases. *Science*, 1998.
- [95] S. K. Khaitan and J. D. McCalley. Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems Journal*, 9(2):350–365, 2015.
- [96] Sven Kleiner, Reiner Anderl, and Robert Gräß. A collaborative design system for product data integration. *Journal of Engineering Design*, 14(4):421–428, dec 2003.
- [97] Matthew Klenk, Johan Kleer, Daniel Bobrow, Sungwook Yoon, John Hanley, and Bill Janssen. *Guiding and Verifying Early Design Using Qualitative Simulation*, volume 2. aug 2012.
- [98] Benjamin Kuipers. Qualitative reasoning: Modeling and simulation with incomplete knowledge. *Automatica*, 1989.
- [99] Benoît Lebeaupin, Antoine Rauzy, and Jean Marc Roussel. A language proposition for system requirements. In *11th Annual IEEE International Systems Conference, SysCon 2017 - Proceedings*, 2017.
- [100] Edward A. Lee. Cyber Physical Systems: Design Challenges. *Proc. of 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'08)*, 2008.
- [101] L. Lemazurier, V. Chapurlat, and A. Grossetête. An MBSE Approach to Pass from Requirements to Functional Architecture. *IFAC-PapersOnLine*, 2017.
- [102] Lori Lemazurier. *Conception d'un système avancé de réacteur PWR flexible par les apports conjoints de l'ingénierie système et de l'automatique*. PhD thesis, 2018.
- [103] C. A. Macana, N. Quijano, and E. Mojica-Nava. A survey on cyber physical energy systems and their applications on smart grids. In *2011 IEEE PES CONFERENCE ON INNOVATIVE SMART GRID TECHNOLOGIES LATIN AMERICA (ISGT LA)*, pages 1–7, 2011.
- [104] Mark W. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1998.
- [105] Pierluigi Mancarella. MES (multi-energy systems): An overview of concepts and evaluation models, 2014.

-
- [106] Y. Matsuno. A design and implementation of an assurance case language. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 630–641, 2014.
- [107] Mehdi Mcharek, Toufik Azib, Moncef Hammadi, Jean Yves Choley, and Cherif Larouci. Knowledge sharing for mechatronic systems design and optimization. *IFAC-PapersOnLine*, 2018.
- [108] Bertrand Meyer. Design by contract, technical report tr-ei-12/co. *Interactive Software Engineering Inc.*, 1986.
- [109] Faïda Mhenni, Jean Yves Choley, Olivia Penas, Régis Plateaux, and Moncef Hammadi. A SysML-based methodology for mechatronic systems architectural design. *Advanced Engineering Informatics*, 28(3):218–231, 2014.
- [110] Davy Monticolo, Julien Badin, Samuel Gomes, Eric Bonjour, and Dominique Chamoret. *A meta-model for knowledge configuration management to support collaborative engineering*, volume 66. jan 2014.
- [111] Ted Bapty Sandeep Neema and Janos Sztipanovits. META-X DESIGN FLOW TOOLS. Technical report, Institute for Software Integrated Systems, Vanderbilt University, 2013.
- [112] Shiva Nejati, Mehrdad Sabetzadeh, Davide Falessi, Lionel Briand, and Thierry Coq. A sysml-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies. *Information and Software Technology*, 54(6):569 – 590, 2012. Special Section: Engineering Complex Software Systems through Multi-Agent Systems and Simulation.
- [113] T. Nguyen. A modelling simulation based engineering approach for socio-cyber-physical systems. In *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, pages 702–707, 2017.
- [114] T. Nguyen. Formal requirements and constraints modelling in form-l for the engineering of complex socio-technical systems. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 123–132, Sep. 2019.
- [115] Thuy Nguyen. A modelling & simulation based engineering approach for socio-cyber-physical systems. In *Proceedings of the 2017 IEEE 14th International Conference on Networking, Sensing and Control, ICNSC 2017*, 2017.
- [116] Thuy Nguyen. An improved approach to traceability in the engineering of complex systems. *2018 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–6, 2018.
- [117] Pierluigi Nuzzo, Alberto L. Sangiovanni-Vincentelli, Davide Bresolin, Luca Geretti, and Tiziano Villa. A Platform-Based Design Methodology With Contracts and Related Tools for the Design of Cyber-Physical Systems. *Proceedings of the IEEE*, 103(11):2104 – 2132, 2015.

-
- [118] Keith OLDHAM, Stephen KNEEBONE, Martine CALLOT, Adrian MURTON, and Richard BRIMBLE. *MOKA - A Methodology and tools Oriented to Knowledge-based engineering Applications*. dec 2018.
- [119] Omg. OMG Systems Modeling Language (OMG SysML TM) v.1.2. *Source*, 2010.
- [120] OMG-OCL. Object Constraint Language. *International Business*, 2014.
- [121] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.
- [122] Christiaan J J Paredis, Yves Bernard, Roger M Burkhart Hans-peter De Koning, and Sanford Friedenthal. An Overview of the SysML-Modelica Transformation Specification. *Jet Propulsion*, 2010.
- [123] Robert Phillips, R. Edward Freeman, and Andrew C. Wicks. What Stakeholder Theory is Not. *Business Ethics Quarterly*, 2003.
- [124] G. Piccoli and F. Pigni. *Information Systems for Managers: Without Cases, Edition 4. 0*. Prospect Press, 2018.
- [125] Gilles Plessis, Aurelie Kaemmerlen, and Amy Lindsay. Buildsyspro: a modelica library for modelling buildings and energy systems. pages 1161–1169, 03 2014.
- [126] Philipp Reinkemeier, Ingo Stierand, Philip Rehkop, and Stefan Henkler. A pattern-based requirement specification language: Mapping automotive specific timing requirements. In *Software Engineering (Workshops)*, volume 184, pages 99–108, 2011.
- [127] Hendrik Roehm, Jens Oehlerking, Thomas Heinz, and Matthias Althoff. STL model checking of continuous and hybrid systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 412–427. Springer, 2016.
- [128] Pascal Roques. MBSE with the ARCADIA Method and the Capella Tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.
- [129] Alberto Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems*. *European Journal of Control*, 2012.
- [130] Wladimir Schamai. Modelica Modeling Language (ModelicaML): A UML Profile for Modelica. 2009.
- [131] Wladimir Schamai, Peter Fritzon, Chris Paredis, and Adrian Pop. Towards Unified System Modeling and Simulation with ModelicaML. 2009.
- [132] S Schuler, A Walsch, and M Woehrle. Assessment of languages and tools for the automatic formalisation of system requirements. Technical report, 2015.

- [133] Zane Scott and David Long. One Model, Many Interests, Many Views - Vitech. Available at <http://www.vitechcorp.com/resources/whitepapers/onemodel.pdf>.
- [134] Jean-Philippe TAVELLA. *Library PowerSysPro, Livrable ModeliScale M3.6.1*. 2014.
- [135] Haydn THOMPSON, Radoslav PAULEN, Michel RENIERS, Christian SONNTAG, and Sebastian ENGELL. Analysis of the State-of-the-Art and Future Challenges in Cyber-physical Systems of Systems. Technical report, 2016.
- [136] Stephen Toulmin. The uses of argument. *The Cambridge Law Journal*, 1958.
- [137] AU Victor Bandur, AU Peter Gorm Larsen, AU Kenneth Lausdahl, AU Sune Wolf, UNEW Carl Gamble, LIU Adrian Pop, ST Etienne Brosse, VSI Jörg Brauer, VSI Florian Lapschies, CLP Marcel Groothuis, and CLP Christian Kleijn. INTO-CPS Tool Chain User Manual. Technical report, 2015.
- [138] David D. Walden, Garry J. Roedler, Kevin J. Forsberg, R. Douglas Hamelin, and Thomas M. Shortell. *Systems Engineering Handbook*. 2015.
- [139] K. Willcox, D. Allaire, J. Deyst, A. Babuscia C. He, and E. Clements. STOCHASTIC PROCESS DECISION METHODS FOR COMPLEX CYBER-PHYSICAL SYSTEMS. Technical report, Massachusetts Institute of Technology, 2012.
- [140] E. S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, pages 226–235, Jan 1997.
- [141] Eric Yu and John Mylopoulos. From e-r to "a-r" - modelling strategic actor relationships for business process reengineering. *International Journal of Intelligent and Cooperative Information Systems*, 4, 10 1995.
- [142] Eric S. Yu. *Social Modeling and i**, pages 99–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Appendix A: Model checking

A.1 Characteristics of Model Checking

Model checking deals in particular with reactive systems with having concurrent components. These systems are generally distributed and interact with their environment through sensors (for information) and actuators (for action) [51]. Model checking handles reactive systems in the form of state machines. Multiple definitions of their syntax and semantic exists in the state-of-the-art. Reference [66] presents state machines such as transition systems (TS) based on the notion of tuple $(S, Act, \rightarrow, I, AP, L)$, with:

- S : set of states (idle, moving...)
- Act : set of actions which are logical formulas on variables (increments, resets, ...)
- $\rightarrow \subseteq S \times Act \times S$: transition relation
- I : set of initial states
- AP : set of atomic propositions / variables (x, paid...)
- $L : S \rightarrow 2^{AP}$: labeling function, that gives the set of AP that are true for each state

Refer to chapter 2 of [49] for more details.

A.2 Model Checking processes

The model checking process requires a rigorous organization and planning with a use of configuration and version management, which proved to be relevant in industrial applications. Verifying a system using model checking implies going through the following steps [66][67]:

1. Modeling

- Model the system behavior by enumerating in a precise and unambiguous way all reachable states of the system and transitions between them. The transitions

describe how the system evolves from one state to another. Moreover, a formal description of the property to be verified must be provided.

- To ensure an accurate description of properties, a specification language is used. This language is based on temporal logic which extends the traditional propositional logic with operators that allow the verification of multiple temporal properties such as:
 - Functional correctness (does the system works as it is supposed to?)
 - Reachability (is it possible to reach a specific state?)
 - Invariance (the property is true for all states)
 - Safety (a given state should never be reached)
 - Liveness (a given state will eventually be reached)
 - Fairness (under certain conditions, does an event occur repeatedly?)
 - Real-time properties

2. Running

- Initializing simulation settings for an exhaustive verification
- Launching simulation for all states of the system model

3. Analyzing the results

Three results are possible:

- The property (or properties) is checked, and the model conforms to the desired behavior.
- The verification property is false, for different reasons:
 - There may be a modeling error \Rightarrow a need to modify the model and reverify all properties, even the ones already checked.
 - The property may not represent the requirement to be validated. In this case, it should be rectified. The design is verified only when all properties have been checked with a valid model.
- The model is too large to be handled. In this case, an abstraction of the model can be used. Alternatively, there are some methods for compressing the model structure by identifying regularities. Another possible solution is to decrease the precision of verification results.

A.3 Comparison between temporal logic languages

There exist multiple temporal logic languages in the state-of-the-art, like LTL [49], CTL [77], CTL* [65], STL [127], OCL [120], HyLTL [64], etc. The main difference between them remains in the temporal operators of language, the rules by which finite state machines are represented and structured, and the capacity to represent temporal aspects.

- Linear Temporal Logic (LTL) is a widely used logic formalism for specifying finite state discrete systems. It is based on a linear time perspective with an infinite sequence of states and where each state has a unique successor. LTL uses the logical operators: (\neg) negation, (\wedge) conjunction, (\vee) disjunction, (\rightarrow) implication and (\leftrightarrow) equivalence, true, false. It also uses the temporal operators: (O) next, (U) until, (\diamond) eventually, (\square) always.
- Computation tree logic (CTL) is based on a branching time view that is structured as a tree. It also deals with discrete finite state systems. In CTL, at a certain moment in time, a state has different paths and possible states in the future, and any one of them might be the one taken. This language is characterized with path quantifiers: (\forall) meaning “Along all paths” and (\exists) meaning “there exists one path”. It uses the same temporal operators of LTL language: (O) next, (\diamond) eventually, (\square) always, (U) until. In CTL, counterexamples may be very complicated to debug and interpret when the system presents complex combinations. CTL can express some properties that LTL cannot do and vice versa, depending on the property. Hence, LTL allows verifying properties over a single path and CTL allows verifying properties over multiple paths. However, CTL cannot express some path properties [77]. An extension of CTL called CTL* comes to address this issue.
- CTL* combines both operators used for linear time and branching time languages. It can also express all formulas that can be described by both LTL and CTL. LTL, CTL, and CTL* have a semantic based on a finite Kripke structure [65] for representing state machines that is compatible with the transition representation presented earlier in this chapter: $M = (S, \rightarrow, L)$, with:

- S : the set of states
- $\rightarrow \subseteq S \times S$: the transition relation
- $L : S \rightarrow 2^{AP}$: the labeling function, that gives the set of AP (atomic propositions) that are true for each state

As mentioned before, LTL and CTL only deal with discrete state machines. However, most complex systems are hybrid systems that gather discrete event, continuous dy-

namics and time functions. Several works have tried to deal with this limitation and developed extensions.

- Metric Temporal Logic (MTL) is a linear temporal logic with integrated timing aspects which enable representing properties concerning a particular time instant in which a transition will take place [121].
- HyLTL is also an extension of LTL to hybrid traces presented in [65]. This logic handles discrete and continuous aspects of systems using a formal hybrid automata semantic. It is defined with a transition system containing both continuous and discrete variables. The paper presents a good description of the theory.
- Signal Temporal Logic (STL) [127] is a real-time logic that extends MTL logic by providing timing aspects and further expressing continuous properties. It allows expression properties like “After 100 seconds from *Alpha*, the temperature *T* must be below 60 degrees”. These logics are supported by multiple existing tools. UPPAAL and Kronos support verifications with the CTL logic. The latest versions of UPPAAL have been extended to handle timed automata [49]. Other tools like SMV developed by CMU and SPIN developed by Bell Labs were used in several applications and protocols for automatic control. HyTech [84] is a known model checker for linear hybrid automata. An interesting general theory about timed automata is available in chapter 9 of [49]. It presents a detailed description of how a time clock variable is integrated into the logic.
- Object Constraint Language (OCL) is an expression-based language used to express formal constraints on models using precise and unambiguous semantics. OCL is most used in UML diagrams as well as in SysML and Capella. The particularity of OCL as compared to other formal languages presented above is that besides being also formal, it can be easily written and read without requiring a strong mathematical background, such benefit for the user not being the case with the other languages. OCL is a purely declarative language where expressions have no effects on models, they only return values about attributes constraints states [120]. OCL expressions are composed of two main parts, the context and the constraint with the following syntax: *Context-name* : <stereotype> *Constraint-name* : Constraint expression
 - The context refers to the element on which the constraint will be applied.
 - <stereotype> refers to the type of the constraint which can be: Invariant (inv), Pre-conditions & Post-conditions (pre & post), Feedback from a query operation (body) and Initial and derived values (init derive).

- The constraint expression part may include variables and operations on objects instances. The OCL standard version 2.4 presents an exhaustive view of possible values and operations on predefined types notably Integer, Real, Boolean, and String. They may be arithmetic or Boolean operations such as $*$, $+$, $-$, $/$, and, or, xor, etc. or conditional expressions on properties such as if, then, else, or, etc. Constraint expressions can refer to instances of the context using “self”, for example: “context Person: inv self.age >0”; Self.age refers to an instance of Person. As presented in [92], the main limitation of OCL is its incapacity of modeling constraints on dynamic system behaviors. The OCL language is a first predicate logic which evaluates system properties over one single state at a specific instant in time. OCL neither supports temporal dimension nor events.
- Temporal Object Constraint Language (TOCL): several works were proposed to extend OCL with temporal constraints. Most of them have been inspired from temporal logic languages such as LTL and CTL. Based on this approach, [92] proposes a temporal extension of OCL using a pattern-based language in order to avoid the complexity of mathematical representations of CTL and LTL. The introduced patterns are the following: Absence, Existence, Bounded Existence, Universality, Precedence, Response, Chain Precedence, and Chain Response. The paper also introduces the notion of “Scopes” representing the path qualifiers over which the system holds: Globally, Before Q, After Q, Between Q and R, After Q until R. Although multiple extensions of OCL to temporal aspects exist, they have not been much used in practice.

A.4 Strengths and weaknesses of model checking

The key strengths of model checking are:

- It is a formal approach for automatically verifying a system model against requirements.
- It is used for various applications in different engineering fields such as software engineering and hardware design.
- It allows partial verification of requirements by giving the possibility of testing each property individually.
- It eliminates ambiguities and identifies errors in the earliest development phases.
- It provides diagnostic methods for debugging and understanding the source of non-validated properties.

- It can be easily integrated into existing development cycles.

The weaknesses of model checking are:

- It is only suited for systems with finite range variables and a limited number of transitions and states which is not the case for most industrial applications (e.g. time-dependent systems or internet communication terminals, or physical processes with infinite number of states).
- It is not very intuitive to the user and requires a strong mathematical background.
- It allows the verification of the system model but not the actual system. Complementary methods are necessary for validating the product or prototype.
- It requires expertise for creating an abstraction of the system with the smallest possible model size in order to avoid combinatory explosion.

Appendix B: Systems Engineering methodologies

B.1 Comparison criteria of SE methodologies

In order to evaluate the methodologies identified in the state of the art, Table B.1 summarises the criteria that were chosen and present their meaning and their levels of quality.

APPENDIX B: SYSTEMS ENGINEERING METHODOLOGIES

Criteria	Quality level	Comments
Appropriateness for large CPS	<ul style="list-style-type: none"> - Appropriate - Partial - Not-appropriate 	<ul style="list-style-type: none"> - Appropriate: Adapted for handling large CPS. - Partial: can only handle small-sized systems with a limited number of interactions. - Not-appropriate: cannot handle complex systems.
Appropriateness for modeling physical systems	<ul style="list-style-type: none"> - Yes - No 	<ul style="list-style-type: none"> - Yes: The methodology has the capacity of representing physical dynamic behaviors. - No: Physical behavior is not in the scope of the methodology.
Lifecycle phases covered by the methodology	<ul style="list-style-type: none"> - All - Medium - Very limited 	<ul style="list-style-type: none"> - All: The methodology affords tools, diagrams, and methods for all design phase allowing requirements decomposition and solution emergence. - Medium: The methodology presents broad guidelines that cover several but not all design phases - Very limited: The methodology is limited to one design phase (e.g. only requirements modeling).
Formalization degree for requirements handling	<ul style="list-style-type: none"> - Informal - Semi-formal - Formal 	<ul style="list-style-type: none"> - Informal: Natural language requirements - Semi-formal: Guided requirements using a template language or having hypertext links. - Formal: Requirements are executable.
Verification of system properties	<ul style="list-style-type: none"> - Guided modeling - Experts reviewing - Formal proof - Model checking - Simulation 	<ul style="list-style-type: none"> - Guided modeling: The methodology affords tools for assisting engineers in developing their models - Expert reviewing: Experts have the final word on whether the system respects properties or not. - Formal proof: Formal demonstration of concepts - Model checking: The platform has the ability to represent the system using model checking methods. - Simulation: The platform enables doing systems simulation.
Automatic verification for system requirements	<ul style="list-style-type: none"> - Model checking - Simulation - None 	<ul style="list-style-type: none"> - Model checking: The methodology automatically verifies requirements using model checking. - Simulation: The methodology automatically verifies requirements using system and requirements simulation. -None: No automatic verification is done.
Traceability through system design	<ul style="list-style-type: none"> - Yes - No 	<ul style="list-style-type: none"> - Yes: The methodology is supported by functionalities for traceability through design phases. - No: The methodology is not supported by traceability functionalities.
Tools and languages used by the methodology		The platform and the set of domain-specific tools proposed by the methodology.
Openness for pairing with other tools (for different system facets)	<ul style="list-style-type: none"> - Yes - Limited - No 	<ul style="list-style-type: none"> - Yes: The possibility of creating linking with tools proposed by other editors. - Limited: Limited to certain tools (not useful). - No: Closed tool, no possible pairing with other platforms.
License terms	<ul style="list-style-type: none"> - Open source - Commercial 	
Ease of use	<ul style="list-style-type: none"> - Easy - Medium - Complex 	<ul style="list-style-type: none"> - Easy: Can be used without special training - Medium: Needs moderate training to be used. - Complex: Needs special skills and consequent training in order to be used

Table B.1: Methodologies comparison criteria

B.2 Evaluation of SE methodologies

Amongst SE methodologies identified in the state of the art and not mentioned in section 2, here are some interesting works:

B.2.1 An MBSE Approach to Pass from Requirements to Functional Architecture

B.2.1.1 Overview

L. Lemazurier introduces in [101][102] a systems engineering method for bridging the gap between the first two phases of the V cycle which are “requirements capture and analysis”, and “architecture design” by framing their processes and making them coherent. The author introduced in this methodology three different views: (i) requirement view, (ii) context view and (iii) behavioral view, supported by five Domain Specific Modeling Languages (DSML) in order to guide engineers and managers to go from requirements to functional architecture using a Product Breakdown Structure (PBS) that is applied iteratively at different levels of system hierarchy. We should keep in mind that this methodology was developed for Instrumentation and Control (I&C) purposes. It was dedicated to be applied on parts of a system based on a metamodel implemented in Teamcenter. Before presenting the methodology, it is interesting to emphasize the focus of this work on verifying model properties at every task and every step taken in design modeling phases. As a reminder, model properties refer to the rules of constructing a model and linking one model to another using the DSML and their guidelines that are provided. Model properties are verified by construction using the DSMLs that guide architects and engineers while creating models. However, coherence between models can be done through simulation and verification tools.

B.2.1.2 Methodology

The methodology is about creating three main system views successively presented hereunder. These views were developed using DSMLs supported by the Capella tool.

i. Requirement view: The input of the methodology is a requirement referential exported from the Teamcenter PLM platform where requirements are specified in a unified view. The author developed a DSML for writing and managing requirements based on natural language. This DSML is supported by a template language with boilerplates formalism to guide and ease engineers to specifying requirements (e.g. <system> shall do <action> before <event>). The reason for using this paradigm is keeping the natural expressiveness

of the requirements for the different stakeholders, while at the same time defining textual predefined guides. This DSML is not meant to be simulated.

ii. Contextual view: The contextual view shown in Figure B-1 aims at placing the system of interest (SoI) into its environment and defining its relationships with the other systems, interfaces, services provided and associated flows and items description. A context DSML is developed for supporting this view with traceability features. DSML enables connecting a functional need (services) to design elements (flows and items) that can be of different types: Material, Energy or Information.

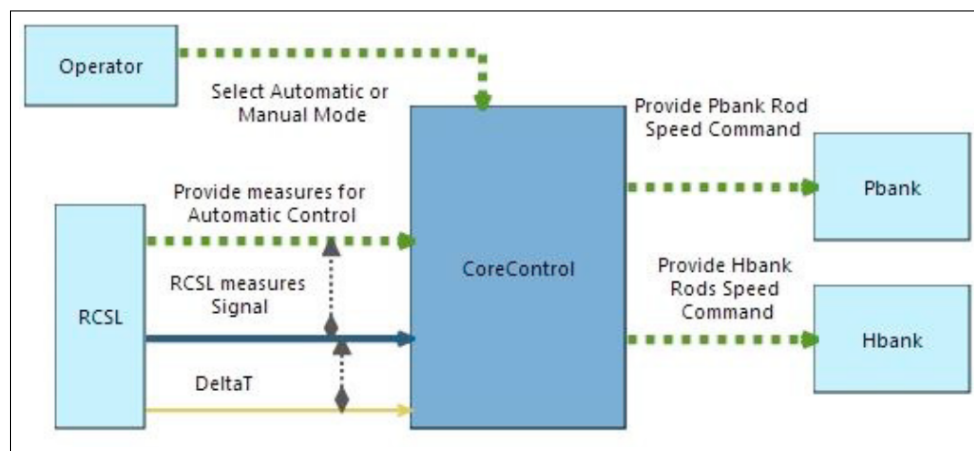


Figure B-1: Core Control Context Diagram (from [101]) - Capella

iii. Behavioral view: The behavioral view comes as a result of the combination of three diagrams created using three developed DSMLs.

- **Modes and transition DSML:** the aim of this DSML is defining functional modes and transitions. Each mode is a gathering of functioning states of the system with specific requirements that are generally different from one mode to the other. This DSML also has the particularity of guided modeling with construction rules. It has an important role in structuring the design. It allows simulating the created diagram and gives the possibility to observe system behavioral sequences and therefore allows to check if they match the ideas at the source of its creation.

Figure 19 shows an example of Modes and transitions diagram taken from [99]. It shows the automatic and manual modes, the transition T1 to T4 with the related conditions, and the default modes (represented by the grey diamonds).

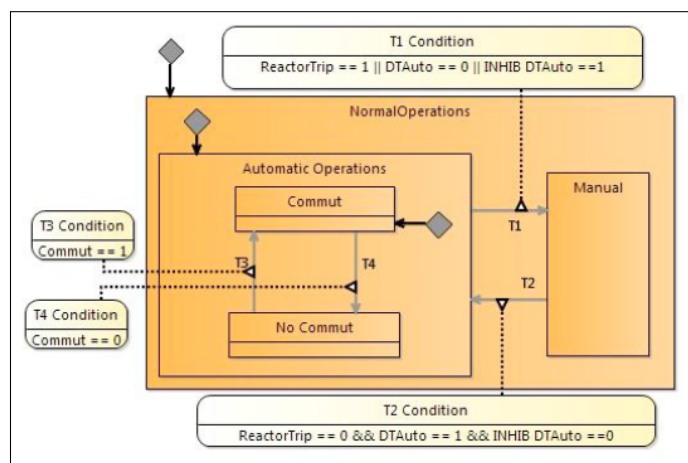


Figure B-2: Modes and transitions diagram (from [101]) - Capella

- Conditional Black-Box Function Flow Diagram (CBBFFD) DSML: For each specified mode, a functional diagram is defined in order to explicit the set of functional requirements related to each mode. This diagram defines the relation between the input and output items of the SoI using a graphical diagram. This DSML allows a traceable relationship between requirements and behavioral view as the CBBFFD diagram describes the functional requirements that will be implemented in the functional architecture. Any requirement modification will be propagated to functions through CBBFFD. At the same time, modifications in the model will enable a signal warning at the requirement level. This functional diagram is done while keeping a black box view on the SoI. Each functional diagram is then linked to its related mode in the Modes and transitions diagram.
- Operational scenarios DSML: The purpose of this view is to define operational uses cases where the designer can specify the expected behavior of their system. The goal is to make verification by comparing the functional architecture simulation results with behavior that emerges from the last two diagrams (Modes and transition, and CBBFFD). Operational scenario diagrams show the interaction of the SoI considered as a black box with the elements from its environment.

Other DSMLs developed by the author were presented in [102] that are useful for bridging the gap between requirement and architectural phases.

B.2.1.3 Summary

This methodology follows a functional breakdown of requirements that is adequate to the automatic field problematic in the origin of this work [101]. It is similar to the SysML

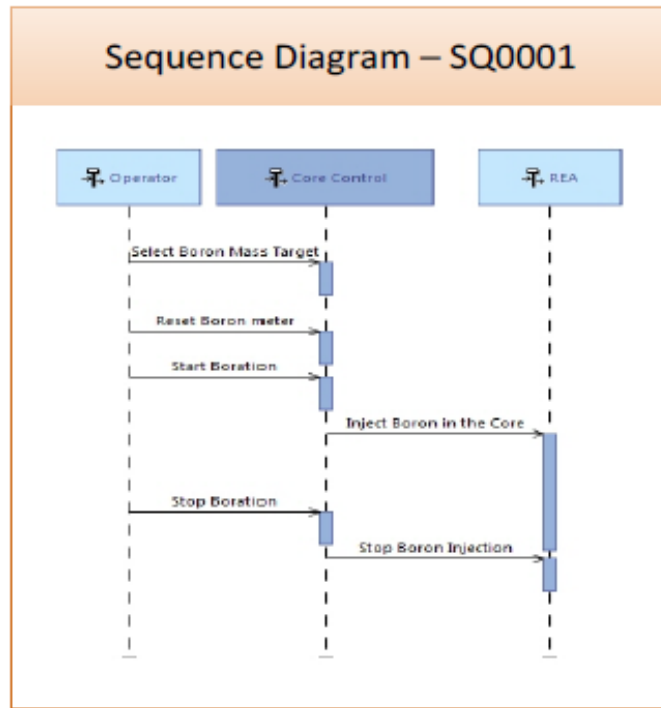


Figure B-3: Illustration of the sequence diagram (from [101]) - Capella

approach presented above, and both are not suitable for CPS design with important physical aspects. Using the DSMLs presented by the author is very interesting when dealing with the system as a black box. It allows covering all its aspects and specifying what is exactly needed for its design. Nevertheless, CPS design never starts from scratch as its general processes are strongly influenced by return of experience on existing systems (e.g. when designing a nuclear plant, processes related to nuclear fission need to be considered from the start). Therefore a full top-down approach is not very suited for CPS design. The main asset of this methodology as compared to the SysML methodology is that it is integrated into the Teamcenter PLM platform which can allow creating relationships with physical behavior modeling tools and other domain-specific ones, hence ensuring a holistic framework for system design. However, as requirements are not executable, it is not possible to make automatic verifications of system behavior against specifications.

B.2.2 A Model-Based Design Methodology for Cyber-Physical Systems

This methodology was introduced by J. Jensen et al. in [90] as a model-based approach dealing with CPS following the steps below:

- Step 1: State the Problem: Informally describe the problem to be solved.
- Step 2: Model Physical Processes: It is the first formal description of the system environment and physical processes to be controlled using differential equations or Laplace transfer functions. Further refinements are done in future steps.
- Step 3: Characterize the Problem: Identify the systems adjustable parameters to be controlled. Understand the interactions between physical processes and computations.
- Step 4: Derive a Control Algorithm: Specify the hypothesis under which physical processes will be controlled and develop the control algorithm that will be executed. The specified constraints concerns latencies, delays, sampling rates and jitters that should be satisfied by the chosen computational platform in order to have accurate measures of physical dynamics to be suitably controlled.
- Step 5: Select Models of Computation: According to the CPS problematic to be analyzed, different models of computations can be chosen under different types of CPS aspects to be represented as discrete events, continuous parameters, timing aspects, etc. Combinations of models of computations are generally considered depending on the CPS complexity. PTIDES (Programming Model for Distributed Real-Time Embedded Systems) was introduced as a tool focusing on specifying and checking timing requirements over networked distributed systems. PTIDES models specifications as discrete-event models with possible extensions for timing constraints.
- Step 6: Specify Hardware: Choose hardware that (i) is adapted to the described physical processes, (ii) supports the environment and (iii) can implement the control algorithm.
- Step 7: Simulate, using simulation tool to analyze the system. If the system is described by several computation models, it becomes necessary to use a tool capable of composing them.
- The Ptolemy II tool was introduced as a support for modeling, simulating and designing systems with a composition of different models of computation [71].
- Step 8: Construct: build the system according to the specifications and test individually the components against the theoretical models.
- Step 9: Synthesize Software: Develop a code executable across different platforms.
- Step 10: Verify, Validate and Test: Perform formal verifications and validations by testing each component individually. The separation between physical systems and

computational systems may be done using hardware in the loop (HIL) that simulates the feedback from the physical processes to the I&C. Reference [90] emphasizes translating requirements into formal specifications for verification and validation.

This methodology has interesting processes for conceiving CPS, however, it presents the following drawbacks:

- It suggests handling the physical aspects of CPS at the early design phases. However, no special emphasis nor proposed dedicated tools for this facet are introduced. This paper focuses on the control and computation part of CPS.
- It only presents broad and general processes with no further application details. Regarding the case of study, it seems to be more adapted to small-sized CPS.
- The tools introduced [90], in particular, Ptolemy and PTIDES are not adapted to our vision for CPS design. Indeed, Ptolemy allows co-simulation by enabling the composition of multiple models of computations in the same platform, and PTIDES is a real-time operating system that specifies timing constraints of discrete event models which can only handle limited aspects of large CPS.
- Verification and validation processes come as final steps of the methodology which can lead to costly modifications when detecting errors at advanced design stages.

B.2.3 OpenMETA / CyPhy

META[11] is a program launched by the Defense Advanced Research Projects Agency's (DARPA) in order to radically improve existing systems engineering methods for defense systems design and shorten dramatically the timelines necessary for CPS design. META is not dedicated to a special approach or tool, it aims at developing design methods for the handling of large heterogeneous CPS.

During this program, a tool suite named CyPhy was proposed as well as different design methodologies. They all aim at affording formal frameworks for supporting design through different abstraction levels with early analysis capacities for efficient solution-space exploration.

CyPhy tool gives the possibility of simulating physical systems using OpenModelica in order to enhance dynamic aspects and verify the static design against a set of operational requirements. CyPhy supports formal requirements verifications at different abstraction levels using stochastic "Probabilistic Certificate of Correctness" (PCC) [139] incorporating uncertainty analysis. PCC is about analyzing potential system faults using probabilistic

state machines. They are used in order to measure the impacts of components properties variations on the ability of the system to comply with the requirements [111].

The methodologies proposed in the META program started from the idea that new designs are based on existing successful systems and components designs. Developing a new system requires exploring a vast combination of available systems. The idea is not to perform a time-consuming detailed analysis of each combination of components, but to do fast but efficient solution space exploration.

One of the proposed approaches consists in testing multiple components deemed by designers as potential solutions, satisfying requirements using a “Qualitative reasoning” [98] for verifying the conformance between numerical results obtained from specific domains models simulation and the required behavior for each case. The purpose of the approach is to enable reasoning with a limited set of data. No numerical parameters are required or neither computed during the analysis. The only possible quantitative treatment is the probabilistic analysis of a qualitative variable and the graphical representation of data.

The concept of qualitative reasoning uses variables that can take continuous values such as real numbers that depend on time, or discrete ones such as enumerated values representing qualitative aspects.

This approach does not require having special linear systems representations. Given a systems model, a qualitative analysis generates automatically all possible behavior trajectories of system variables. Analyzing these trajectories and challenging requirements against them will support design choices.

The idea of this approach is translating Modelica blocks only once into qualitative models which will become a new models library that will be used for the analysis. However, it is mentioned that complex models need further human assistance. Requirements are specified using temporal logic languages as LTL or CTL.

This approach catches up with existing formal methodologies dealing with requirements verification using state machines representations of system behavior. However, it has the particularity of being based on concrete physical representation using Modelica models that are translated by giving qualitative values to model variables [97].

B.2.4 INTO-CPS

INTO-CPS [17] is an abbreviation for Integrated Tool Chain for Model-based Design of Cyber-Physical Systems. It is a European project that aims at developing an integrated tool chain that supports cross-disciplinary domains and collaborative modeling for CPS while considering all phases from requirements down to hardware and software realization.

The INTO-CPS methodology gathers multiple domain-specific tools from different providers on a common platform in order to create a “multi-model”. The idea is to achieve a global

co-simulation by using Functional Modeling Units (FMUs) exported from different tools, each one of them being dedicated to a special phase of the co-simulation activity. This methodology uses a SysML/INTO-CPS profile in order to develop the architecture of the multi-model by either starting from scratch and specifying the characteristics as well as the connections between the FMUs to be developed later, or importing existing FMUs and creating the connections graphically.

The INTO-CPS tool-chain is presented in [137]. It is made of a central Co-simulation Orchestration Engine (COE) that gathers and controls the FMUs provided by (i) **OpenModelica** [24], an open source tool for modeling multi-domain physical systems, (ii) **Over-ture** [26], an open source tool that supports modeling discrete system using VDM models (Vienna Development Method) which is an object-oriented paradigm based on formal mathematical construction for computer-based design enabling proofs for models properties and executable items, (iii) **20-sim** [5], a commercial tool for modeling continuous multi-domain dynamic systems using different approaches. Models can be developed using equations, graphical representations such as block diagrams, physical components and bond graphs that can be all integrated into a common model. The bond graph paradigm allows developing a domain-independent description of the physical systems. (iv) **Modelio** [22], an open source modeling platform supporting UML and SysML technology. Modelio will be used as a top-level architecture using SysML language profiles.

RT-Tester [29] is also a tool taking part in the INTO-CPS tool-chain but is not under control of the COE in co-simulation time. It supports model-based testing with a product named RTT-MBT which is an automatic test generator that allows test executions and real-time test evaluations while generating traceability for requirements data, test cases and results. RTT-MBT only supports test goals in Linear Temporal Logic LTL or specifications represented using transition and logical systems. We will not focus on this methodology as co-simulation is not the purpose of our study.

B.2.5 UnCoVerCPS

UnCoVerCPS [35] is an abbreviation for Unifying Control and Verification of Cyber-Physical Systems. It is a European project that aims at proposing guidelines for developing critical cyber-physical systems.

This program introduced a tool-chain that aims at formally capturing requirements and challenging them against system behavior to verify its compliance. A tool named Formel-Spec was developed in this context that aims at formalizing requirements using a graphical template language that is automatically translated into hybrid automata using the STL language [127]. These formal properties are then integrated into the SpaceEX [31] tool that performs verification using behavioral models [132].

This approach used in this work catches up with the existing formal methods in terms of using logical representations and automata-based design. We will not present further details about it.

Comparison criteria Methodologies	Appropriateness for large CPS	Appropriateness for modeling physical systems	Lifecycle phases covered by the methodology	Formalization degree for requirements handling	Verification of system properties	Automatic verification for system requirements	Traceability through system design	Tools and languages used by the methodology	Openness for pairing with other tools	License terms	Ease of use
TeamCenter / SDPD	Yes	Yes	All	Semi-formal	Guided modeling, Expert reviewing and simulation	No	Yes	Teamcenter, Amesim (for 0D / 1D), NX (for 3D), Made (for safety and reliability), etc.	Yes	Commercial	Medium
3Dexperience / RFLP	Partial	Yes	Medium	Semi-formal	Guided modeling, Expert reviewing, and simulation	No	Yes	Requirement app, Functional & Logical design application, integrated Dymola, 3D modeling app	No	Commercial	Medium
INRIA (Contact-based design)	Partial	No	Medium	Formal	Guided modeling + Model checking	Model checking	No	MICA, and other Model checking tools.	Limited	Open source	Complex
Contract-based design, University of California, Berkeley [117]	Partial	No	Medium	Formal	Guided modeling + Model checking	Model checking	No	ARCHEX/CPLEX for architecture design, TULIP Toolbox for LTL, SIMULINK for control design	Limited	Open source	Complex
SysML based methodology [109]	Partial	No	Medium	Informal	Guided modeling + Expert reviewing + simulation	No	Yes	SysML	Limited	Open source	Medium
DSML methodology [101]	Yes	No	Medium	Semi-formal	Guided modeling + simulation	No	Yes	Teamcenter + Capella	--	Open source + Commercial tools	Medium

Comparison criteria Methodologies	Appropriateness for large CPS	Appropriateness for modeling physical systems	Lifecycle phases covered by the methodology	Formalization degree for requirements handling	Verification of system properties	Automatic verification for system requirements	Traceability through system design	Tools and languages used by the methodology	Openness for pairing with other tools	License terms	Ease of use
B Method	No	No	Limited	Formal	Guided modeling + Model checking	Model checking	Yes	ADA, C, C++	--	Open source	Complex
Knowledge based	Yes	Yes	Medium	Formal + Semi-Formal	Guided Modeling	Simulation	Yes	SysML + DSLs, Refer to [68], [50]	Yes	commercial	Medium
Model-Based Design Methodology for CPS [90]	No	Yes	Limited	Informal	No	No	No	Ptolemy II + Ptypes + LabVIEW	Yes	Open source	Medium
UnCoVerCPS	No	No	Limited	Formal	Guided Modeling+ Model checking	Model checking	No	SpaceEx + CORA	Limited	Open Source	Complex
OpenMETA / CyPhy	Yes	Yes	Medium	Formal	Model checking + simulation	Model checking	Yes	CyPhy + OpenModelica + Python + OpenMDAO	Yes	Open source	Complex
INTO-CPS	Partial	Yes	Medium	Informal	Guided-model + Model checking + Simulation	Model checking	Yes	Modelio + Overture+ OpenModelica + 20-sim + RT-Tester (RTT_MBT)	Yes	Open Source + commercial tools	Medium
Combining SysML/UML to Modelica	No	Yes	Limited	Semi-Formal	Guided modeling	No	Yes	SysML + Modelica	Limited	Open source	Medium
The Harmony Process – IBM / SysML	Partial	No	Medium	Informal	Guided modeling + Expert reviewing + simulation	No	Yes	SysML	Limited	Open source	Medium
Capella / Arcadia	Yes	No	Medium	Semi-formal	Guided modeling + Expert reviewing + simulation	No	Yes	Capella	Limited	Open source	Medium

Table B.2: Summary table of methodologies from the state-of-the-art

Appendix C: A formal contract based framework

This section is not currently available as it will be submitted for the scientific journal “Advanced Engineering Informatics” ADVEI during the year 2021. Once published by the journal editor, this section will be publicly available.

Appendix D: Formal properties in ReqSysPro

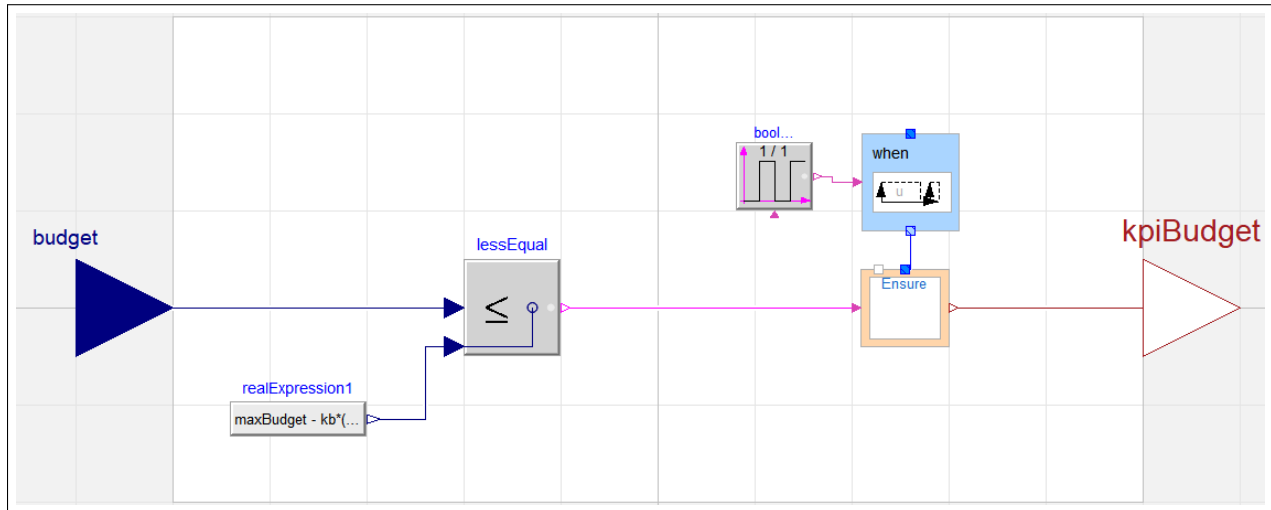


Figure D-1: Budget KPI represented using ReqSysPro blocks in Modelica

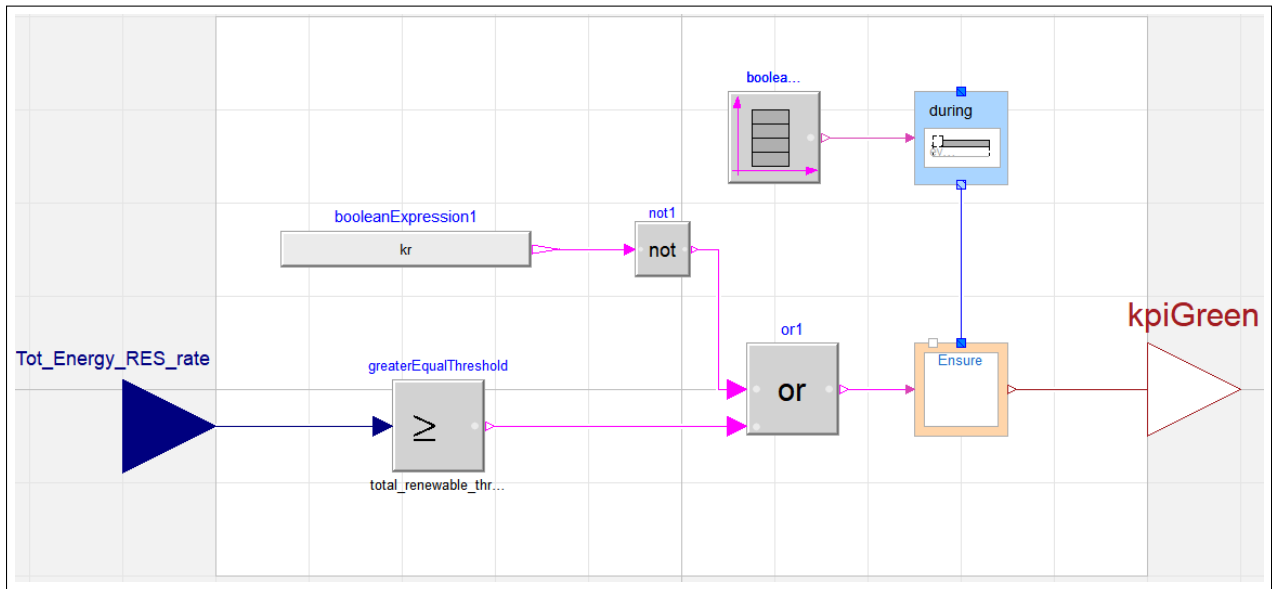


Figure D-2: Green KPI represented using ReqSysPro blocks in Modelica

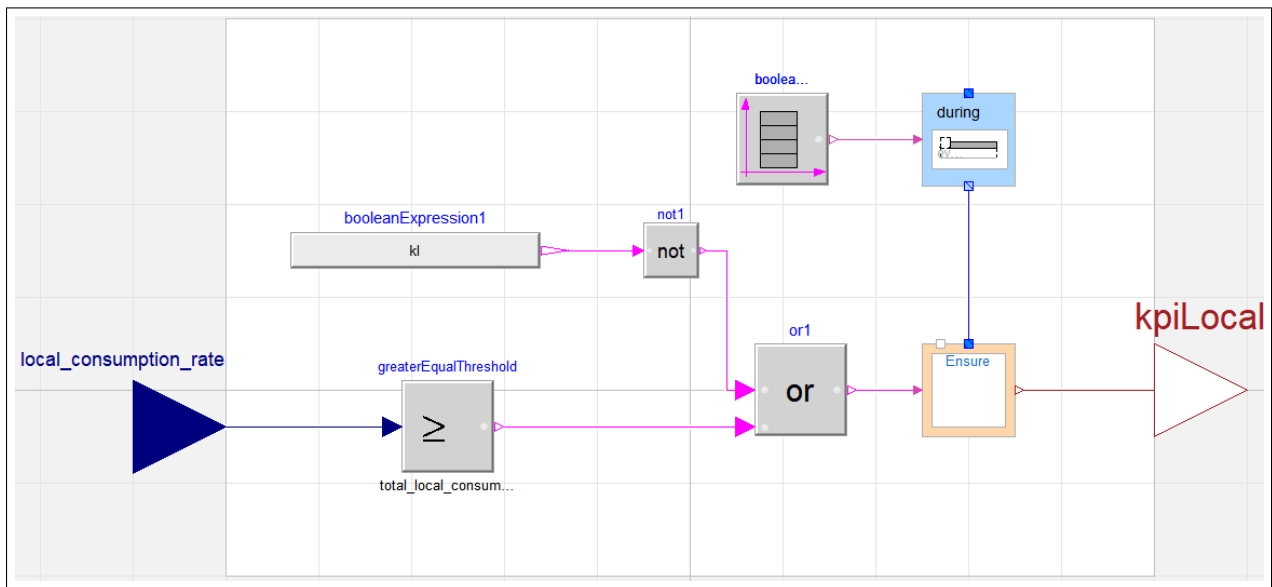


Figure D-3: Local consumption KPI represented using ReqSysPro blocks in Modelica

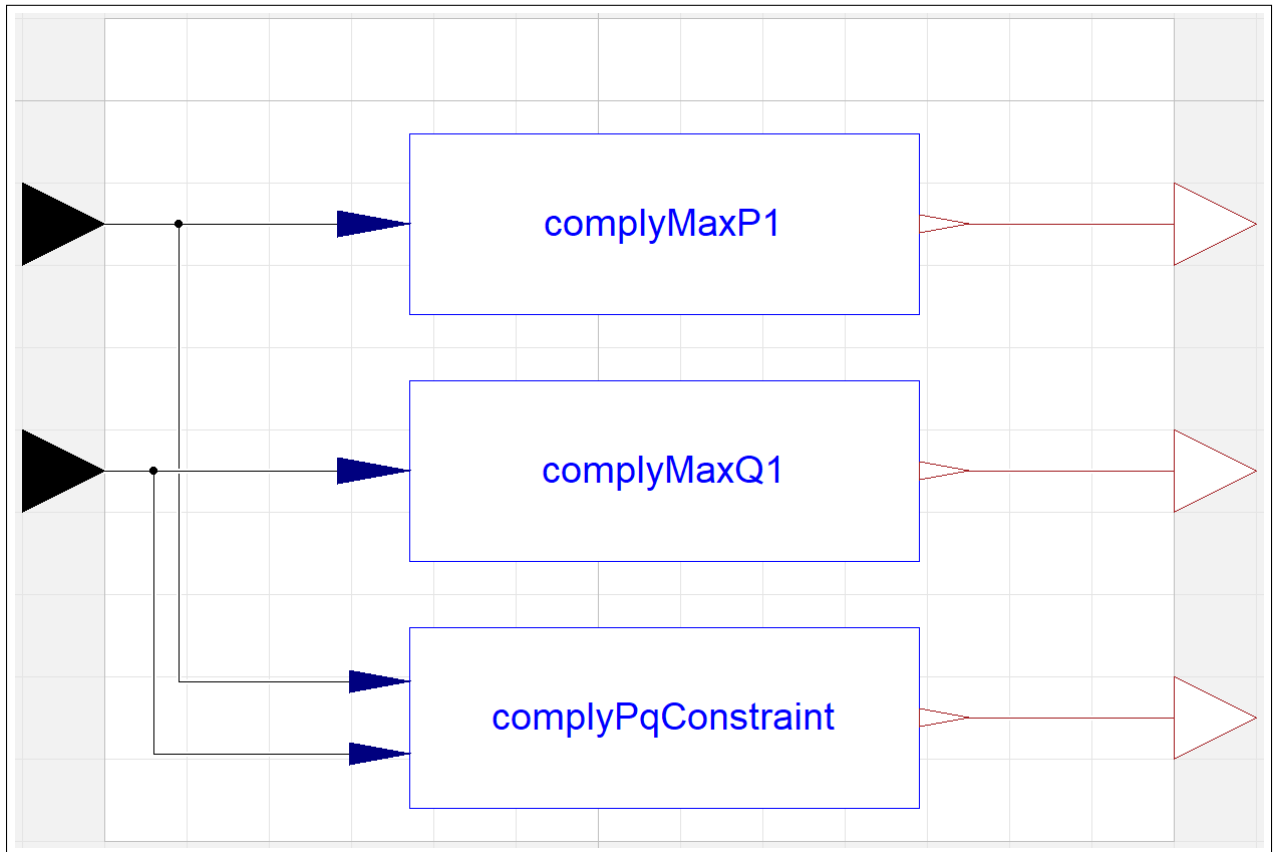


Figure D-4: The requirements corresponding to consumers guarantees and the E-DSO assumptions

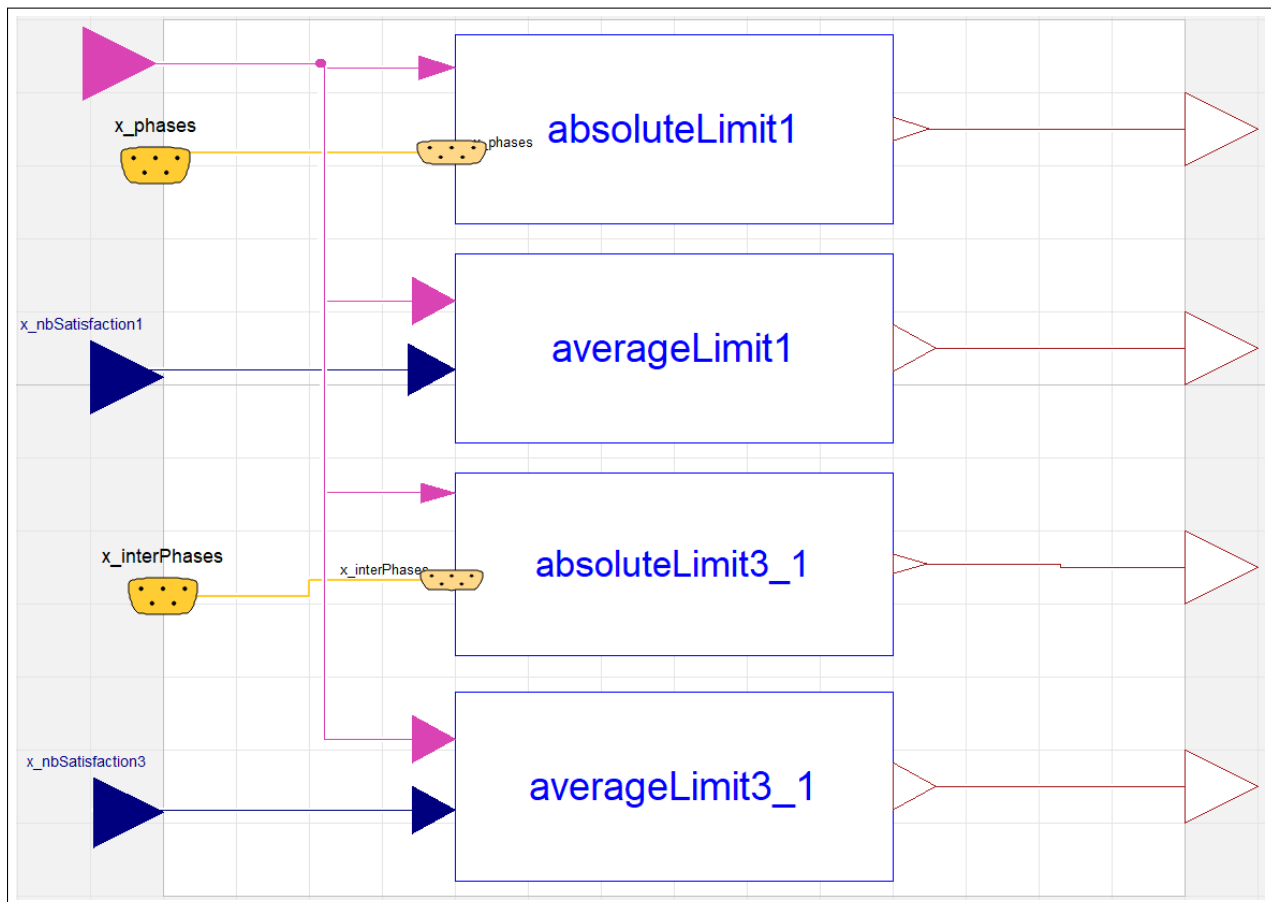


Figure D-5: The requirements corresponding to E-DSO guarantees and the consumers assumptions

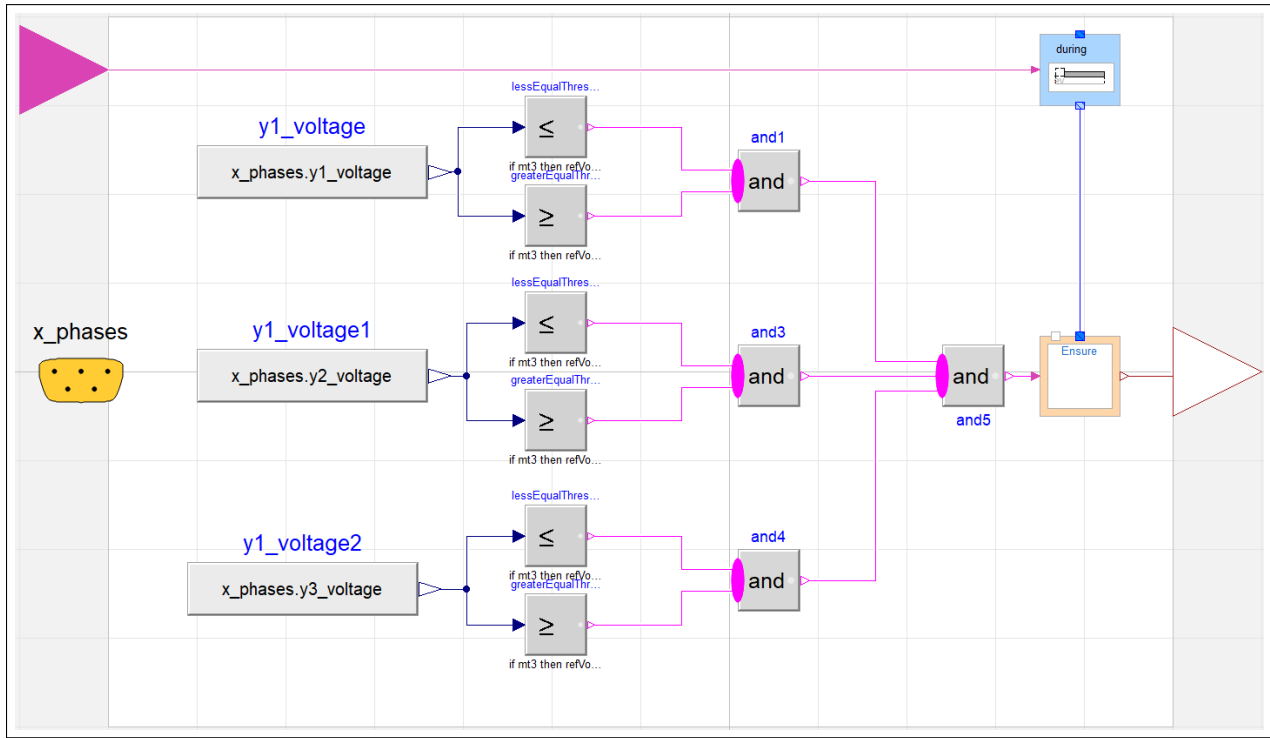


Figure D-6: AbsolutLimit requirement represented using ReqSysPro blocks in Modelica

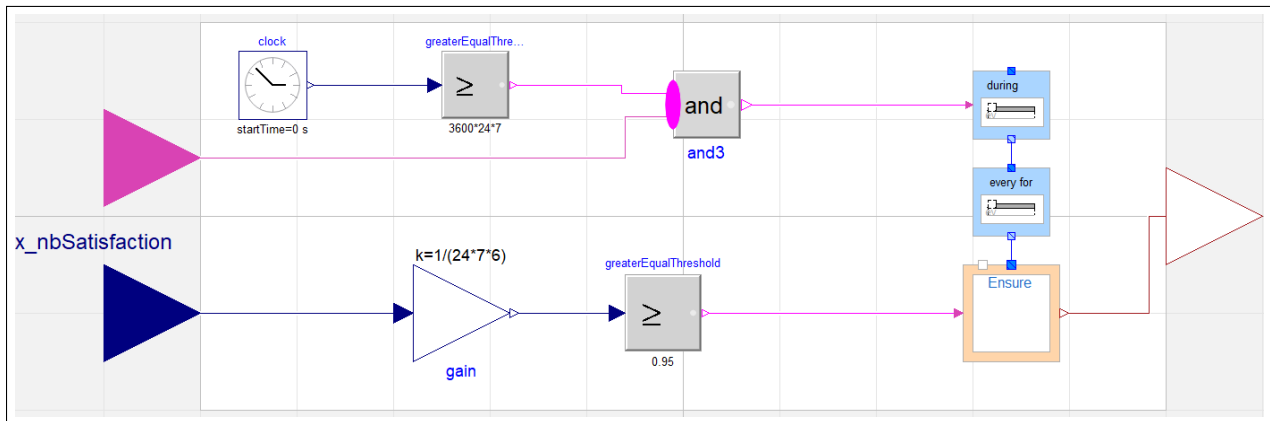


Figure D-7: AverageLimit requirement represented using ReqSysPro blocks in Modelica

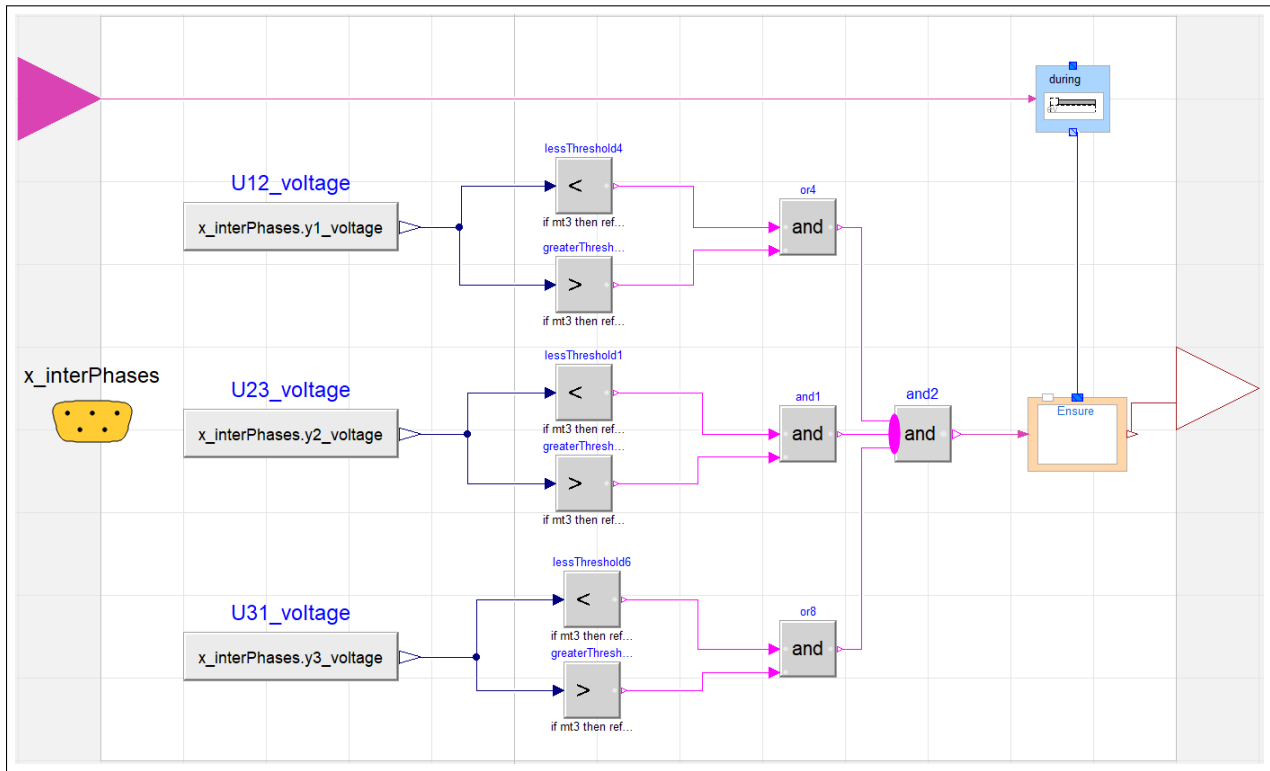


Figure D-8: AverageLimit3 requirement represented using ReqSysPro blocks in Modelica

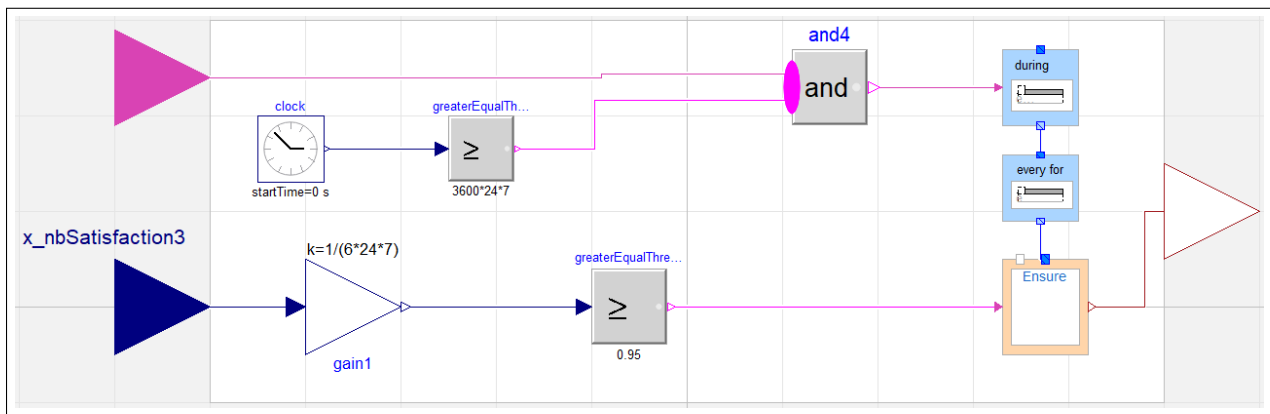


Figure D-9: AverageLimit3 requirement represented using ReqSysPro blocks in Modelica

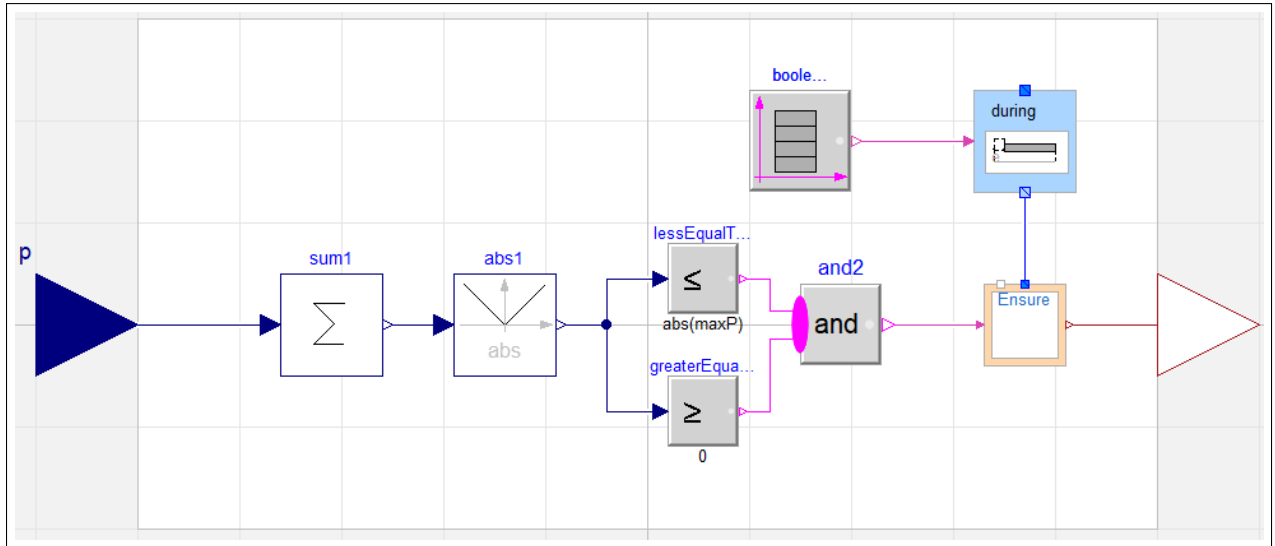


Figure D-10: ComplyMaxP requirement represented using ReqSysPro blocks in Modelica

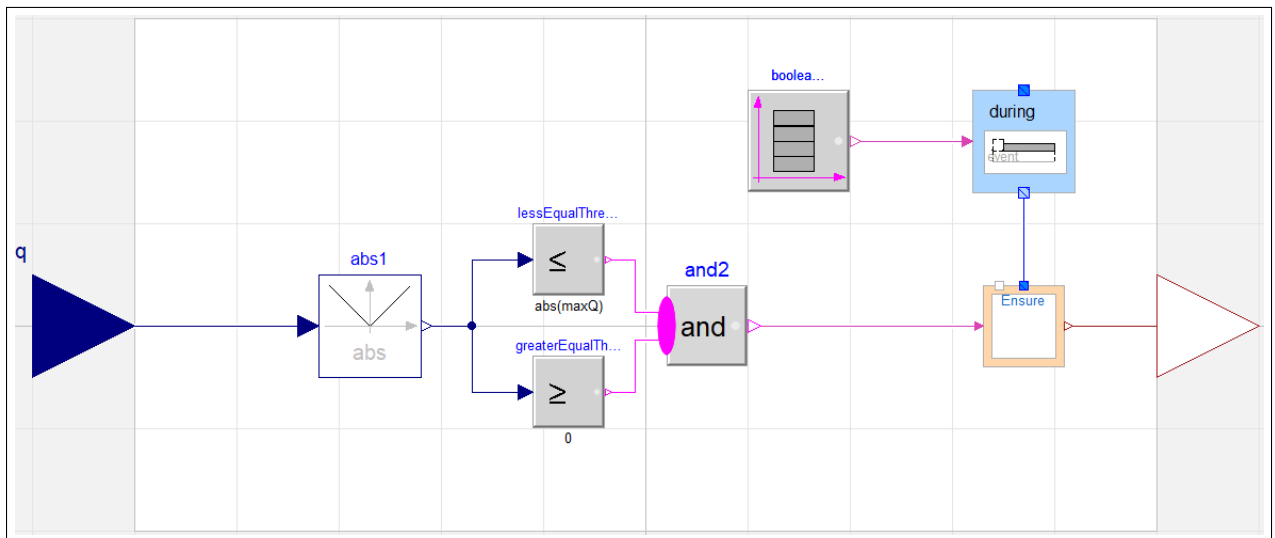


Figure D-11: ComplyMaxQ requirement represented using ReqSysPro blocks in Modelica

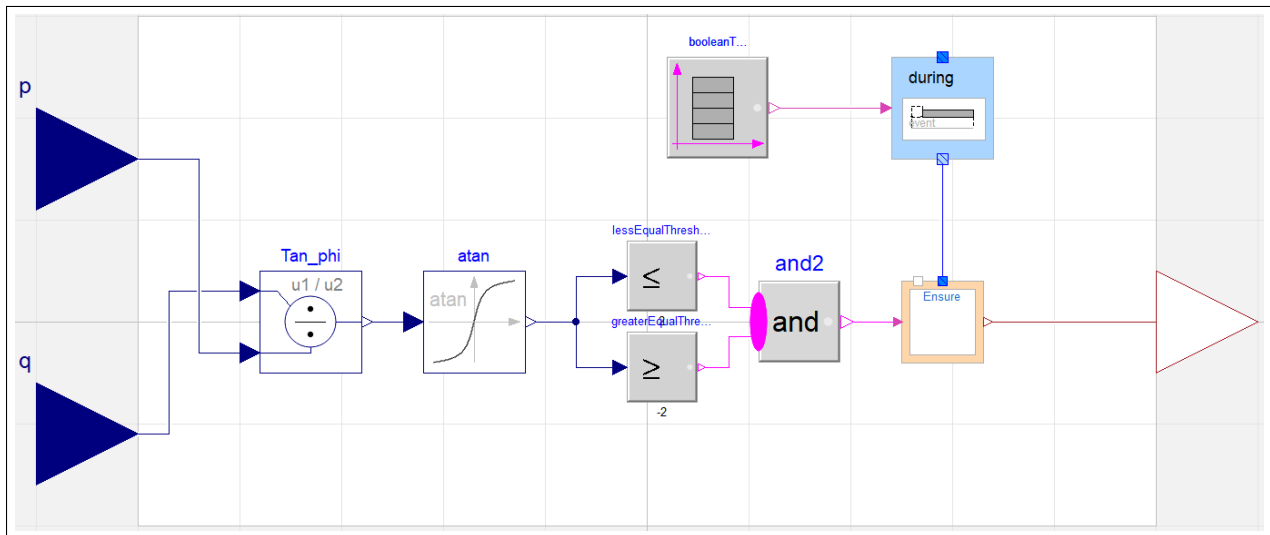


Figure D-12: ComplyPQconstraint requirement represented using ReqSysPro blocks in Mod-
elica

Appendix E: Observation models and bindings

```
model Observer_for_all_consumers
  parameter Real renewableFactorGrid=0.2
    "The Renewable factor in the global grid (imported energy)";
  Real Curtailment_P_kW "the power produced by PV and not consumed locally";
  Real Curtailment_E_kWh "the energy produced by PV and not consumed locally";
  Real sum_P_kW "Total consumption of all buildings";
  Real Tot_imported_gen_kWh "the total imported electricity";
  Real PV_Generation_E_kWh "energy produced by PV and consumed locally";
  Real DHN_gen_P_kW "DHN consumed power";
  Real DHN_gen_E_kWh "DHN consumed energy";
  Real Tot_Energy_RES_rate "Total energy (electric+heat) renewable share";
  ■
equation
//Computation of the total consumption of all buildings
sum_P_kW = BATIMENT0000000005011465_Pload + BATIMENT00000000322361260_Pload
+ BATIMENT00000000322361254_Pload + BATIMENT0000000005011407_Pload
+ BATIMENT0000000005011403_Pload + BATIMENT00000000322361227_Pload
+ BATIMENT00000000322361089_Pload + BATIMENT00000000322361274_Pload
+ BATIMENT00000000322361273_Pload + BATIMENT00000000322361275_Pload
+ BATIMENT00000000320456131_Pload + BATIMENT00000000356487656_Pload
+ BATIMENT00000000322361106_Pload + BATIMENT00000000322361280_Pload
+ BATIMENT0000000005011458_Pload + BATIMENT00000000322361256_Pload
+ BATIMENT0000000005011480_Pload + BATIMENT00000000322361232_Pload
+ BATIMENT00000000322361121_Pload + BATIMENT00000000356487694_Pload;

// Computation of the energy (consumed minus produced) by all the buildings
// considering only the PV that was consumed locally
der(Tot_imported_gen_kWh) = (1/(1000*3600))*(max(0,sum_P_kW));

// Computation of the power produced by PV and not consumed locally
Curtailment_P_kW =min(0,sum_P_kW);

// Computation of the energy produced by PV and not consumed locally
der(Curtailment_E_kWh)=(1/(1000*3600))*Curtailment_P_kW;
```

Figure E-1: Modelica text view of the observation model for KPI_3 and KPI_4 (part 1)

```

// Computation of the energy produced by PV and consumed locally
PV_Generation_E_kWh =energy_system_0001_E_PV_kWh +energy_system_0002_E_PV_kWh
+energy_system_0003_E_PV_kWh +energy_system_0004_E_PV_kWh
+energy_system_0005_E_PV_kWh +energy_system_0006_E_PV_kWh
+energy_system_0007_E_PV_kWh +energy_system_0008_E_PV_kWh
+energy_system_0009_E_PV_kWh +energy_system_0010_E_PV_kWh
+energy_system_0011_E_PV_kWh +energy_system_0012_E_PV_kWh
+energy_system_0013_E_PV_kWh +energy_system_0014_E_PV_kWh
+energy_system_0015_E_PV_kWh +energy_system_0016_E_PV_kWh
+energy_system_0017_E_PV_kWh +energy_system_0018_E_PV_kWh
+energy_system_0019_E_PV_kWh +energy_system_0020_E_PV_kWh
-abs(Curtailment_E_kWh);

//Computation of DHN consumed power (we have selected only the buildings
//that are connected to the DHN)
DHN_gen_P_kW = BATIMENT0000000005011407_P_DHN + BATIMENT0000000005011403_P_DHN
+ BATIMENT0000000322361227_P_DHN + BATIMENT0000000322361089_P_DHN
+ BATIMENT0000000322361274_P_DHN + BATIMENT0000000322361273_P_DHN
+ BATIMENT0000000322361275_P_DHN + BATIMENT0000000356487656_P_DHN
+ BATIMENT0000000322361106_P_DHN + BATIMENT0000000005011458_P_DHN
+ BATIMENT0000000322361232_P_DHN + BATIMENT0000000322361121_P_DHN;

// Computation of DHN consumed Energy
der(DHN_gen_E_kWh) = (1/(1000*3600))*DHN_gen_P_kW;

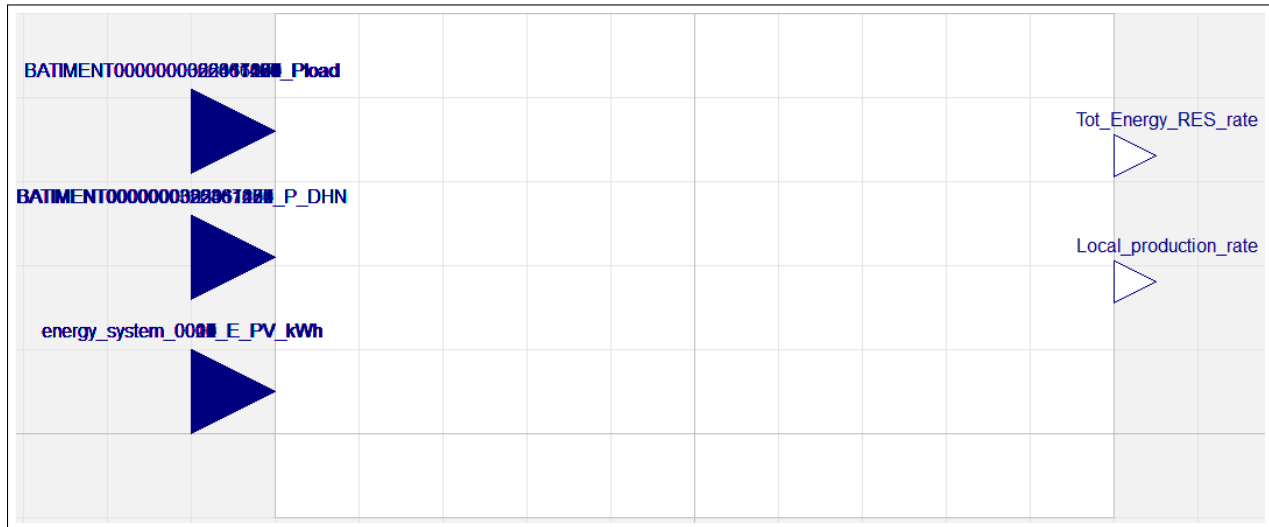
//Computation of the electric renewable share (Tot_local_gen_needed_kWh+PV_Generation_E_kWh)
//is the total electricity of the PG)
Electricity_renewable_rate = (PV_Generation_E_kWh + renewableFactorGrid*
Tot_imported_gen_kWh)/max(1e-4, (Tot_imported_gen_kWh + PV_Generation_E_kWh));

//Computation of the energy (elec+heat) renewable share
Tot_Energy_RES_rate = (PV_Generation_E_kWh + renewableFactorGrid*
Tot_imported_gen_kWh)/max(1e-4, (Tot_imported_gen_kWh + PV_Generation_E_kWh +
DHN_gen_E_kWh));

// Computation of self efficiency rate
Local_production_rate=(PV_Generation_E_kWh+DHN_gen_E_kWh) /max(1e-04, (Tot_imported_gen_kWh
+ PV_Generation_E_kWh+DHN_gen_E_kWh));
end Observer_for_all_consumers;

```

Figure E-2: Modelica text view of the observation model for KPI_3 and KPI_4 (part 2)

Figure E-3: Modelica diagram view of the observation model for KPI_3 and KPI_4

```

7 #List of buildings IDs
8 with open('list_MODULO_C.txt','r') as mon_fichier1:
9     list_build=mon_fichier1.read()
10    list_build=list_build.split()
11 #List of the maximum power per building
12 with open('list_MODULO_C_maxP.txt','r') as mon_fichier2:
13     list_param_maxP=mon_fichier2.read()
14     list_param_maxP=list_param_maxP.split()
15
16 Obv_model=open('observation_model.txt','w')
17
18 # Opening code for the observation model
19 Obv_model.write('model KPIs_observation_model\n')
20
21 # parameters declaration
22 Obv_model.write('\n // max powers\n')
23 for i in list_param_maxP :
24     Obv_model.write('parameter Modelica.SIunits.ActivePower '+i+';\n')
25
26 Obv_model.write('\n // whether the buiding is heated by electricity or not\n')
27 for i in list_build :
28     Obv_model.write('parameter Boolean '+i+'_elecHeating=true;\n')
29 # Inputs for each observation model (4 per instance )
30 Obv_model.write('\n // list of inputs \n')
31 for i in list_build:
32     Obv_model.write('Modelica.Blocks.Interfaces.RealInput '+i+'_temperature_in \n \
33         annotation (Placement(transformation(extent={{-140,64},{-100,104}}))); \n'+
34         'Modelica.Blocks.Interfaces.RealInput '+i+'_setpoint_in \n \
35         annotation (Placement(transformation(extent={{-140,30},{-100,70}}))); \n'+
36         'Modelica.Blocks.Interfaces.RealInput '+i+'_heatingPower_in \n \
37         annotation (Placement(transformation(extent={{-140,-2},{-100,38}}))); \n'+
38         'Modelica.Blocks.Interfaces.RealInput '+i+'_elecPower_in \n \
39         annotation (Placement(transformation(extent={{-140,-30},{-100,10}}))); \n')
40

```

Figure E-4: Python code developed for the instantiation of the first section of the observation model used for consumers KPIs evaluation (part1)


```

41 # Creation of outputs (4 per instance )
42 Obv_model.write(' \n // list of outputs \n')
43 for i in list_build:
44     Obv_model.write('Modelica.Blocks.Interfaces.RealOutput '+i+'_temperature_out \n \
45         annotation (Placement(transformation(extent={{100,64},{120,84}}))); \n'+
46         'Modelica.Blocks.Interfaces.RealOutput '+i+'_setpoint_out \n \
47         annotation (Placement(transformation(extent={{100,36},{120,56}}))); \n'+
48         'Modelica.Blocks.Interfaces.RealOutput '+i+'_notKt \n \
49         annotation (Placement(transformation(extent={{100,12},{120,32}}))); \n'+
50         'Modelica.Blocks.Interfaces.RealOutput '+i+'_budget \n \
51         annotation (Placement(transformation(extent={{100,-10},{120,10}}))); \n')
52
53 # Instantiation of the observation model on the 20 buildings
54 Obv_model.write('// list of instances of observation models per consumer\n')
55 for i in list_build:
56     Obv_model.write('Observer_per_consumer observers_on_'+i+' (elecHeating='+i+'_elecHeating) \n \
57         annotation (Placement(transformation(extent={{-40,2},{44,68}}))); \n')
58
59 # Connection between the inputs and the instances of the observation model
60 Obv_model.write(' \n equation \n')
61 Obv_model.write(' \n // input connections \n')
62 for i in list_build:
63     Obv_model.write('connect(observers_on_'+i+'.temperature_in, '+i+'_temperature_in); \n'+
64         'connect(observers_on_'+i+'.setpoint_in, '+i+'_setpoint_in); \n'+
65         'connect(observers_on_'+i+'.heatingPower_in, '+i+'_heatingPower_in) ; \n'+
66         'connect(observers_on_'+i+'.elecPower_in, '+i+'_elecPower_in); \n')
67 #Connection between the outputs and the observation instances
68 Obv_model.write(' \n // outputs connection \n')
69 for i in list_build:
70     Obv_model.write('connect(observers_on_'+i+'.temperature_out, '+i+'_temperature_out); \n'+
71         'connect(observers_on_'+i+'.setpoint_out, '+i+'_setpoint_out); \n'+
72         'connect(observers_on_'+i+'.notKt, '+i+'_notKt) ; \n'+
73         'connect(observers_on_'+i+'.budget, '+i+'_budget); \n')
74
75 # end of the observation model
76 Obv_model.write(' \n end KPIs_observation_model; \n')
77
78 Obv_model.close()

```

Figure E-5: Python code developed for the instantiation of the first section of the observation model used for consumers KPIs evaluation (part2)