Bridging the Gap Between Requirements and Object-Oriented Models

Marat Abilov

Business Informatics / Very Large Business Applications
Carl von Ossietzky University of Oldenburg
Ammerlnder Heerstr. 114-118
26129 Oldenburg
marat.abilov@uni-oldenburg.de

Abstract: The processes of requirement engineering and software design take place during a software development. Business processes, goals, organizational models of company can be considered for specification of software requirements. Then analysis models that are used in software design are derived from functional and non-functional software requirements. Waterfall model suggests the existence of complete requirements before moving to software design stage. In iterative software development, that is more popular nowadays, requirement engineering and software design processes occur on different stages during the project. Thus, software design is constantly refined and minor improvements are made in it. The high-level architecture, on the other hand, should stay untouched in order of preventing of its destruction. The major focus of this paper are the processes of derivation of high-level architecture and software design during the iterative model of software development.

1 Introduction

Business analytics in companies use Business Process Models (BPM) to specify product development workflow and other processes that can appear in business practice [IRRG09]. These models are essential in requirement engineering (RE) process of developing information systems (IS) for companies. Additionally, goals that have long been recognized to be essential components of the RE process [VL01], point out the reasons and advances of new IS for companies. As long as IS is a part of wider system, called the organizational system [Mon99], organizational models that help to understand the organization for which a software is going to be developed [dlVG11] are also important during RE stage.

In the following software development phases, the IS is designed and implemented using different methodologies and programming languages. The Object-Oriented (OO) Methodology that helps to describe domain and software using OO models, nowadays is the most popular one.

Since domain area influences the development of IS, there is a need in approach that helps to derive OO design models from goals, BP and organizational models. There are still open problems in methods and approaches in bridging the gap between requirements and

OO design models, that are investigated in the following sections.

2 Literature Review

2.1 Goal-Oriented Requirement Engineering

Maps [Rol07], an approach of using conceptual modelling in RE, allows to model multi-faceted purpose of a system To-Be. Each Map diagram, that represented as a directed graph, consists of sections. Section is an aggregation of two intentions, linked with a strategy. The first and the last intentions are the start and the stop one. Sections in diagram may have three types of relationships: thread, path and bundle. Thread relationship represents that several strategies can be used to achieve the same intention. Path relationship shows that in order to achieve the next goal, the previous one should be achieved first. Bundle relationships restricts the way of achieving the intention by using exclusive OR for sections. The process of constructing To-Be diagram starts from building As-Is one and continues by refining it with new requirements and bridging the gaps. The information presented in Map diagram helps to model To-Be BPD which then can be used for improving business processes of companies and for defining software requirements.

The i* (distributed intentionality) framework [Eri09] attempts to shift focus from activities and informational flow to social analysis. The actor is the central modelling construct in i*. In this context they are autonomous real-world phenomena, that system designers must deal with. Intentional modelling in i* provides technique for explaining and characterizing actor's behaviour. Strategic Dependency Model displays the dependency of one actor on another for something. Strategic Rational Model attributes goals, tasks, resources and softgoals to each actor. Each goal can be achieved by different tasks. Tasks also contribute to achievement of softgoals positively or negatively with different strengths. High-level softgoals are then divided into more specific ones through AND or OR partial contribution.

KAOS [vLDDD91, DvLF93] is a systematic approach for discovering and structuring requirements. KAOS was designed to fit problem description, to improve problem analysis, to clarify responsibilities and to improve communication among stakeholders. In KAOS requirements are connected to high-level goals which they help to achieve. For each requirement and expectation there is a responsible agent that performs operations in order to satisfy the requirement. KAOS is independent of the type of software development process. For each phase of the iterative or incremental process, the system analytic need to build a model and a tool will generate a requirements document for him. KAOS consists of 4 models: goal model, responsibility model, object model, operation model. The goal model helps to define hierarchy of goals and requirement of the system. The responsibility model shows the responsible agent for each low-level requirement. The object model shows the entities, agents and association among them, and it is represented using UML Class Diagram notation. The operational model describes the agents' behaviour in the way of fulfilment of their requirements.

The analysis of goal-based RE shows differ results. An empirical evaluation of i* frame-

work in model-based software generation environment based on feature evaluation framework [ERPM06] shows existence of issues in supporting seven of nine analysed features. Three of them are not supported at all by i*, they are modularity, reusability, scalability. Maps approach, on the other hand, fails in emphasizing social dimension of intentionality [Eri09]. KAOS approach can be considered as very complicated because strategies are modelled as goals [dlVG11]. But from the other point of view, KAOS is the only approach among presented that is independent from software development process, which is the problem that this research tries to address. In addition, Object model in KAOS is represented using UML Class Diagram notations, so it will require less effort for translation to target OO software design models. Moreover, KAOS syntax and semantics shows the ability to formally represent the conceptual primitives of the strategic models of the framework i* [MCPE03].

2.2 Organizational Modelling

Tropos project [CKM02] is the whole software development process with the result of implementation in Beliefs-Desire-Intentions environment. Tropos uses i* along the process: on early RE stages of requirement acquisition and definition, on design stages of architectural and detailed design. In the frame of Tropos project there is also a development of i* framework extension, named Formal Tropos. Tropos suggests usage of existing organizational architectural styles and selection among them based on rationality presented as Non-Functional Requirement (NFR) diagram. Then the existing strategic dependency and strategic rationale models should be changed according to chosen organizational style.

ARchitecture of ntegrated Information Systems (ARIS) [Sch00] is the framework for connecting business requirements and IS. ARIS uses Event-Driven Process Chains (EPC) for specification of business processes. ARIS aims in providing univocal basis for the IS development. ARIS contains of 5 views: organization view for representing the structure of organization; data view for representing business information; function view for representing process tasks; product/service view for representing structures of products and services; process view for depicting dynamic behaviour of processes.

UML-based approach for modelling a business [EP00] describes business using Eriksson-Penker Business Extensions of the UML. The approach starts with presenting overall meta-model, that describes the business modelling concepts. For business processes the approach uses activity diagrams with new process element specified in extension. Business events are represented as class diagrams with new stereotype for send and receive actions. Business events can then be allocated into packages. Goals are specified in object diagram with special stereotype. Business rules can be specified as stereotyped note, OCL constraint or multiplicity in relationship on each of the diagrams.

Another OO approach for business modelling [Mon99] tries to describe the organizational system by means of objects. The approach analyses general organization and presents the components of an organizational system and their interrelationships as UML class diagrams. All high-level objects were also detailed using inheritance relationships. This

model can be used for organizational structure analysis, and can proceed to RE phase in software development. But the actual instances of business processes were not analysed in this work. So connection between business processes and their implementation was not established. The idea of OO organizational structure can be used in current research, because some information from this structure can be used for deriving OO models for the software.

2.3 Derivation of Software Design

2.3.1 Approaches for Functional Requirements

"Methodological Approach" [dlVG11], tries to address challenges existed in software development process. This approach has 4 main stages: organizational modelling, purpose analysis, specification of system requirements, and derivation of OO diagrams. On each stage, different artefacts have to be developed. The artefacts from the last stage influences the subsequent software development stages. The most interesting stages, in connection to current research, are the last ones. During the System Requirement stage, "To-Be" Business Process Diagrams (BPD) in the form of Extended Business Process Diagrams (EBPD) (an extension of the Business Process Modelling Notation, BPMN [OMG11]) have to be developed, that show how process should occur in organization after software system implementation. After that, the System Requirement Specification (SyRS) has to be developed in the form of Extended Task Description (ETD) templates. This proposed style of SyRS is a combined and improved combination of other approaches: among them are Lauesenâs Task & Support Descriptions [Lau02], essential use cases [CL99], Info Case approach [FWB08], and quality requirements from ISO 9126-1 standard [ISO01]. The information presented in these templates is derived not only from BPD, but from previous stage's artefacts as well. These templates show mainly the functional requirements of the software system. After having the ETD specified, OO diagrams have to be derived from them according to set of rules presented in this work. Rules help to derive two main diagrams: the class diagram, and the state diagram. Among overall contribution of the reviewed work, there are several open issues in connection to current research:

1. This approach can mainly be used in waterfall model of software development, when all work on previous stages should be done in order to proceed to next stage. The connection to iterative, spiral or agile model was not specified in this work.
2. On every stage of this approach the validation is need to be done by customer stakeholders, other validation techniques were not specified.
3. The process of analysing other non-functional requirements was not formalized and left for system analyser to consider them in software architecture.
4. In class diagram a rule for specifying the control class was not defined.
5. The problem of parallel works in BPD was not analysed, and rules to consider them in state diagrams were not specified.

Analysis of this work shows that the next ideas can be effectively applied to current research:

1. Enriched BPD, as an intermediate model between BPM and OO models.
2. ETD templates, as a presentation of the expected behaviour of the software
3. Rules to derive OO diagrams from ETD, as a background, that can be extended to formalize this process for iterative methodology of software development

The approach of deriving UML analysis models from use-case models (UCM) is presented in the work of [YBL10].The approach deals with text-based, specified in restricted natural language (NL), UCM (RUCM) [YBL09]. It contains 2 steps:

Step 1: The RUCM models have to be parsed and object model of them has to be generated. At first, the top level objects have to be identified: Use case, Use case specification, actors, brief description, the flows of events, pre- and postconditions. Then, the sentences, objects are constructed from, have to be parsed by NL parser, the Stanford Parser [DMMM06].

Step 2: The UCMeta model have to be transformed into a UML analysis model. The overall algorithm of transformation and the set of rules were specified: for generating overall structure, for generating a class diagram, for generating a sequence diagram, and for validation.

When deriving the class diagram, three types of objects were analysed:

- Boundary objects, that handle interaction between actors and the system
- Entity objects, that are responsible for storing and providing access to data
- Control objects, that control the interaction of participating objects

In UML these type of classes were specified with stereotypes: <<Boundary>>, <<Entity>>, and <<Control>>. The first set of rules presented in this work deals with deriving this three types of classes and relationships between them. The idea of different stereotypes for classes and rules for deriving them can be found useful in current research.

Component Bus System Property (CBSP) approach [GEM04] that helps to connect requirements and architecture can be used in iterative software development model: the output of the first iteration can be used as an input in the second. In this approach, the decision technique is based on voting among multiple architects or experts. The CBSP process consists of 5 steps:

Step 1: Selection of requirements for next iteration. On this step, the most important requirements have to be analysed and prioritized among 2 criteria: importance, feasibility based on stakeholder decisions.

Step 2: Architectural classification of requirements by the relevance to CBSP dimensions, have to be decided by experts.

Step 3: Identification and resolution of classification mismatches, have to be decided by multiple experts independently. Level of consensus for requirement has to be calculated, and if the level is equal or more than largely then requirement should be accepted.

Step 4: Architectural refinement of requirements. On this step, requirements have to be rephrased and split, redundancies have to be eliminated. Then relevance coefficients of properties for each CBSP dimensions have to be decided among 4 common architectural styles.

Step 5: Trade-off choices of architectural elements and styles with CBSP. On this step requirements have to be refined and rephrased to CBSP model elements in such a manner, that no conflicts exist and all model elements at least largely relevant to one of the CBSP dimension.

Among overall contribution of this work, there are some open issues that are related to current research:

1. Analysed architectural styles are very high level, the derivation of detailed OO was not given in this work;
2. Business process as a type of requirements were not analysed in this work.

An approach for integrating UML and EPC [LA98] tries to connect them on three types of granularity: object internal, object system inside and beyond the scope of an object system. The design of EPC diagrams is also performed in three steps: high-level EPC with connection to IS packages, medium-level EPC with connection to classes, detailed EPC with connections to operations and attributes. The detailed EPC diagrams help to derive the UML class diagrams, the state-chart diagrams of classes and other UML diagrams. The approach defines procedural model for applying the integration between EPC and UML and targets RE and software design stages. Although the connection of different level between EPC diagrams and UML diagrams were presented, the concrete guidelines for establishing such diagrams were not specified. In class diagrams the procedure for finding relationships between classes was not specified.

Technique, presented in [RDtHI07], connects OO and process-oriented (PO) models using formal approach. The procedure of translation from OO to PO models has 5 steps:

Step a) Translation from State Machine Diagrams (SMD) to Heuristics Net using algorithm I;

Step b) Translation from SMD to Annotation Repository using algorithm I;

Step c) Translation from Heuristics Net to Petri Net using algorithm II;

Step d) Creation of a skeleton structure of a process-centric model, using existing conversion plugins

Step e) Completion the process-centric model in the from of Yet Another Workflow Language (YAWL) [VDATH05] model

Obtained PO models are highly detailed. This work can be used to connect two different development approaches, also for analysing OO design from procedure perspective. Here are also some open issues in connection to current research:

1. This approach requires finalized OO models to be translated into PO ones.
2. This approach describes only one-way translation from OO to PO models. Although, the authors promised to start working on backward algorithm.
3. Validation technique was not provided in this work.

The approach, presented in this work can be used as a validation technique for the current research.

2.3.2 Approaches for Non-Functional Requirements

The approach of integrating non-functional requirements and conceptual models: Entity-Relationship (ER), OO model [CdPLdMSN01], has several steps: on the first step the Language Extended Lexicon (LEL) [LF$^+$93] have to be developed. LEL registers the vocabulary of a given University of Discourse (UofD) â general context, where the software have to be developed or operated [LF$^+$93]. While developing the LEL two principles have to be followed: maximum circularity, and minimum vocabulary. Each NFR has to be detailed using NFR graph, that defines the goal on the root and subgoals on the leaves. Subgoals can contain attributes, which can be general and data ones. General attributes characterize the whole system, while data attributes can be used for deriving ER and OO models. The paper presented the extensions to ER and OO models. In the OO model:

- Additional information, in the form of attached rectangles, was added to the class to show the names of used UofD and NFR.
- To improve traceability, NR_ prefix has to be added to classes, attributes and operations driven from NFR.
- Class names have to by symbols of the LEL

Some heuristics to integrate the NFR into conceptual models were presented. This heuristics are not showing the straightforward derivation of models, but the some hints that can help the software engineer in this process. Some ideas in this work can be used in current research:

1. Using LEL to describe the NFR;

2. NFR graph, to explore the influence of the NFR on design and;

3. Traceability idea: to present the name of used NFR and UofD in class diagram, that can be done, by using notes or comments.

The influence of the NFR over the decision of using software design patterns were analysed by the work [GY01]. For each system there might be multiple NFRs, and some decisions that support one of the requirement can hurt another one. To address this problem NFR graph was used, that helps to reflect the decision trees for several NFR. This tree was used to describe the reason of usage of design pattern in the system architecture. The design pattern forces changes in the architecture, so author presented the way of connecting NFR graph with functional decomposition of the system.

3 Open Issues

To summarize the above research, there are several challenges in connecting or bridging the requirement- and OO viewpoints:

1. Development of software system can be iterative, and not all information about business processes in organizations can be known before the design phase. Therefore, the new proposed approach in this work should be able to deal with incomplete information, and overcome the absence of some details within it.

2. Business processes of organization and non-functional requirements can influence the final design of the software system, so this approach should be able to deal with different types of requirements.

3. Relationships between classes are very hard to determine, and the problem increases when classes can be organized to perform some special behaviour. Some of these organizations are the examples of software design patterns. As a result, this approach should be able to define any kind of relationships in addition to show what kind of patterns can be applied.

4 Research Methodology

This study will be done by research project following the design research process [VKJ07].

Empirical methods can be used in current research for validation of the approach results. There are several empirical methods that can be used in software engineering context [ESSD08].In most cases, several of them can be used simultaneously in order to achieve more rigidity of the research.

The next tasks are considered as support for answering the above-mentioned research questions:

1. Analyse BPM and other types of requirements, and their influences on the overall software design;
2. Analyse the types of relationships between classes and their derivation from BPM and other requirements;
3. Analyse the types of design patterns and their derivation from BPM and other requirements and
4. Design the approach for derivation OO models from BPM and other requirements that can be used in iterative, spiral and agile models of software engineering

For the formalization of OO models derivation process, the ETVX (Entry, Task, Verification, and eXit) [RROC85] approach can be used.

5 Proposed Approach

The proposed approach (Fig. 1) consists of three main stages, each of them have several steps inside:

- Preparation and derivation of high level software design:
  - Defining goals. On this step software practitioner (SP) analyses the purpose of the system and presents software features as goal diagram;
  - Defining system border. SP analyses software features and business processes of organization and defines IS boundaries;
  - Derivation of the software design with layers. SP uses NFR graph to support decision over what type of software layering to use;
  - Ordering by priority BPDs. SP orders the BPDs that will be supported by IS by priority. Business process can be core, support or administrative;
  - Ordering by priority NFRs. SP orders the NFRs that IS should met by priority.
- Refinement of the software design with business processes:
  - Popping out the BPD. SP considers the next most important BPD. SP models the detail BPD with specification of what activities will be supported by IS.
  - Refinement of the software design from only successful path of BPD;
  - Implementation, testing and getting feedback;
  - Refinement of the software design from feedbacks and alternative paths;
  - Implementation, testing and getting feedback. Software team implements and tests the software. The feedback from users are considered and proper changes in software design are made;
- Refinement of the software design with non-functional requirements:

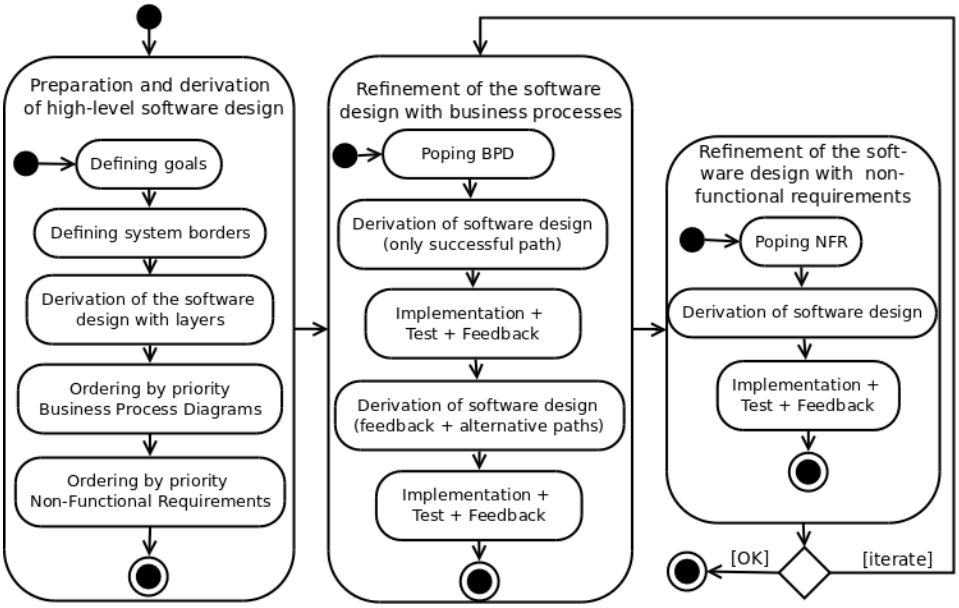

Figure 1: Proposed Approach

- Popping out the NFR. The next most important NFR is considered. The NFR graph for decision over the software design pattern to use is designed;
- Refinement of the software design. The software design is refined according to chosen design pattern;
- Implementation, testing and getting feedback;

The second and the third stages are repeated until all BPDs and NFRs will be considered in the software design.

6 Conclusions

The paper introduced the literature review on the topic of bridging the gap between requirements and OO models. The literature review shows that there are still open questions that need to be investigated. Empirical methods in research methodology will help develop an approach that will be close to software practice. The paper also presents the proposed approach.

References

[CdPLdMSN01] L.M. Cysneiros, J.C.S. do Prado Leite, and J. de Melo Sabat Neto. A framework for integrating non-functional requirements into conceptual models. *Requirements Engineering*, 6(2):97–115, 2001.

[CKM02] Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the Tropos project. *Information Systems*, 27(6):365–389, September 2002.

[CL99] L. L. Constantine and L. A. Lockwood. *Software for use: a practical guide to the models and methods of usage-centered design*. Addison-Wesley, 1999.

[dlVG11] J. L. de la Vara González. *Business process-based requirements specification and object-oriented conceptual modelling of information systems*. PhD thesis, Polytechnic University of Valencia, 2011.

[DMMM06] Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

[DvLF93] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1â2):3–50, April 1993.

[EP00] Hans-Erik Eriksson and Magnus Penker. *Business modeling with UML*. Wiley Chichester, 2000.

[Eri09] S Yu Eric. Social Modeling and i*. In *Conceptual Modeling: Foundations and Applications*, pages 99–121. Springer, 2009.

[ERPM06] Hugo Estrada, Alicia Martnez Rebollar, Oscar Pastor, and John Mylopoulos. An Empirical Evaluation of the i* Framework in a Model-Based Software Generation Environment. In Eric Dubois and Klaus Pohl, editors, *Advanced Information Systems Engineering*, number 4001 in Lecture Notes in Computer Science, pages 513–527. Springer Berlin Heidelberg, January 2006.

[ESSD08] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering*, pages 285–311, 2008.

[FWB08] M. H. Fortuna, C. M. Werner, and M. R. Borges. Info Cases: integrating use cases and domain models. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 81–84, September 2008.

[GEM04] P. Grünbacher, A. Egyed, and N. Medvidovic. Reconciling software requirements and architectures with intermediate models. *Software and Systems Modeling*, 3(3):235–253, 2004.

[GY01] Daniel Gross and Eric Yu. From non-functional requirements to design through patterns. In *Requirements Engineering*, 2001.

[IRRG09] Marta Indulska, Jan Recker, Michael Rosemann, and Peter Green. Business process modeling: Current issues and future challenges. In *Advanced Information Systems Engineering*, pages 501–514. Springer, 2009.

[ISO01] ISO. International Standard ISO/IEC 9126-1: Software engineering â Product quality â Part 1: Quality Model. ISO, International Organization for Standardization, 2001.

[LA98] P. Loos and T. Allweyer. Process Orientation and Object-Orientation-An Approach for Integrating UML and Event-Driven Process Chains (EPC). *Publication of the Institut für Wirtschaftsinformatik, Paper*, 144, 1998.

[Lau02] S. Lauesen. *Software requirements: styles and techniques*. Addison-Wesley Professional, 2002.

[LF+93] J.C.S. Leite, A.P.M. Franco, et al. A strategy for conceptual model acquisition. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 243–246. IEEE, 1993.

[MCPE03] Alicia Martnez, Jaelson Castro, Oscar Pastor, and Hugo Estrada. Closing the gap between organizational modeling and information system modeling. In *WER03-VI Workshop em Engenharia de Requisitos*, 2003.

[Mon99] J. Montilva. An object-oriented approach to business modeling in information system development. In *Proceedings of the III World Multiconference on Systemics, Cybernetics and Informatics - SCI'97*, volume 2, pages 358–364, International Institute of Informatics and Systemics, Orlando, Florida, USA, August 1999.

[OMG11] OMG. Business Process Model and Notation (BPMN). Version 2.0. OMG, Object Management group, 2011.

[RDtHI07] G. Redding, M. Dumas, A.H.M. ter Hofstede, and A. Iordachescu. Reconciling Object-oriented and Process-oriented Approaches to Information Systems Engineering. In *Proceedings of the 3rd International Workshop on Business Process Design (BPDâ07)*, 2007.

[Rol07] Colette Rolland. Capturing system intentionality with maps. *Conceptual modelling in Information Systems engineering*, page 141â158, 2007.

[RROC85] R.A. Radice, N.K. Roth, AC O'Hara, and W.A. Ciarfella. A programming process architecture. *IBM Systems Journal*, 24(2):79–90, 1985.

[Sch00] August Wilhelm Scheer. *Aris: Business Process Modeling*. Springer, 2000.

[VDATH05] W.M.P. Van Der Aalst and A.H.M. Ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[VKJ07] Vijay K Vaishnavi and William Kuechler Jr. *Design science research methods and patterns: innovating information and communication technology*. Auerbach Publications, 2007.

[VL01] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.

[vLDDD91] Axel van Lamsweerde, Anne Dardenne, B Delcourt, and F Dubisy. The KAOS project: Knowledge acquisition in automated specification of software. In *Proceedings AAAI Spring Symposium Series*, pages 59–62, 1991.

[YBL09] T. Yue, L. Briand, and Y. Labiche. A use case modeling approach to facilitate the transition towards analysis models: Concepts and empirical evaluation. *Model Driven Engineering Languages and Systems*, pages 484–498, 2009.

[YBL10] T. Yue, L. Briand, and Y. Labiche. *Automatically Deriving a UML Analysis Model from a Use Case Model*. Carleton University, 2010.