Software Quality Starts with the Modelling of Goal-Oriented Requirements

Emmanuelle Delor, Robert Darimont

CEDITI
Avenue Georges Lemaître, 21
B-6041 Charleroi
Belgium
Phone : +32 (0) 71 25 94 04
Fax : +32 (0) 71 37 20 64

{Emmanuelle.Delor , Robert.Darimont}@cediti.be

André Rifaut

CETIC
rue Clément Ader, 8
B-6041 Gosselies
Belgium
Phone : +32 (0) 71 91 98 25
Fax : +32 (0) 71 91 98 02

Andre.Rifaut@cetic.be

Abstract

Developing high quality requirements specifications is mandatory for a number of critical industrial systems. The KAOS goal-driven methodology has been designed to elicit and validate requirements and also to prove their consistency. This methodology has been successfully implemented in two integrated tools (Objectiver and FAUST) and has been validated in many industrial projects.

All of these have shown how the quality of the requirements can be improved with KAOS/Objectiver due to the following factors :

- a rigorous reasoning and decision making on the requirements,
- the constructive nature of the goal-oriented method,
- the tracing facilities automatically generated between properties and specifications, and
- the automatic generation of reports based on the goal- oriented structure of the requirements.

Another contribution to the quality of requirements is the tight integration between informal requirements and formalised requirements. One the one hand, the KAOS methodology allows the analyst to give a complete informal description of the specification. On the other hand, the FAUST extensions of the Objectiver environment allows to describe and analyse the formal aspects of the specification by making use of some of the best formal tools hidden behind the stakeholder view on the requirements.

Key-words: goal-directed approaches, formal methods, V&V, safety-critical system modelling

1. Introduction

A recent report from the Standish Group estimates that a staggering 40 percent of all software projects fails. To a large degree, these failures are linked to requirements - either they are incorrectly defined from the beginning or are poorly managed as they evolve throughout the project life cycle. In safety-critical applications, those errors can have life-threatening consequences on the software environment or can induce financial disasters.

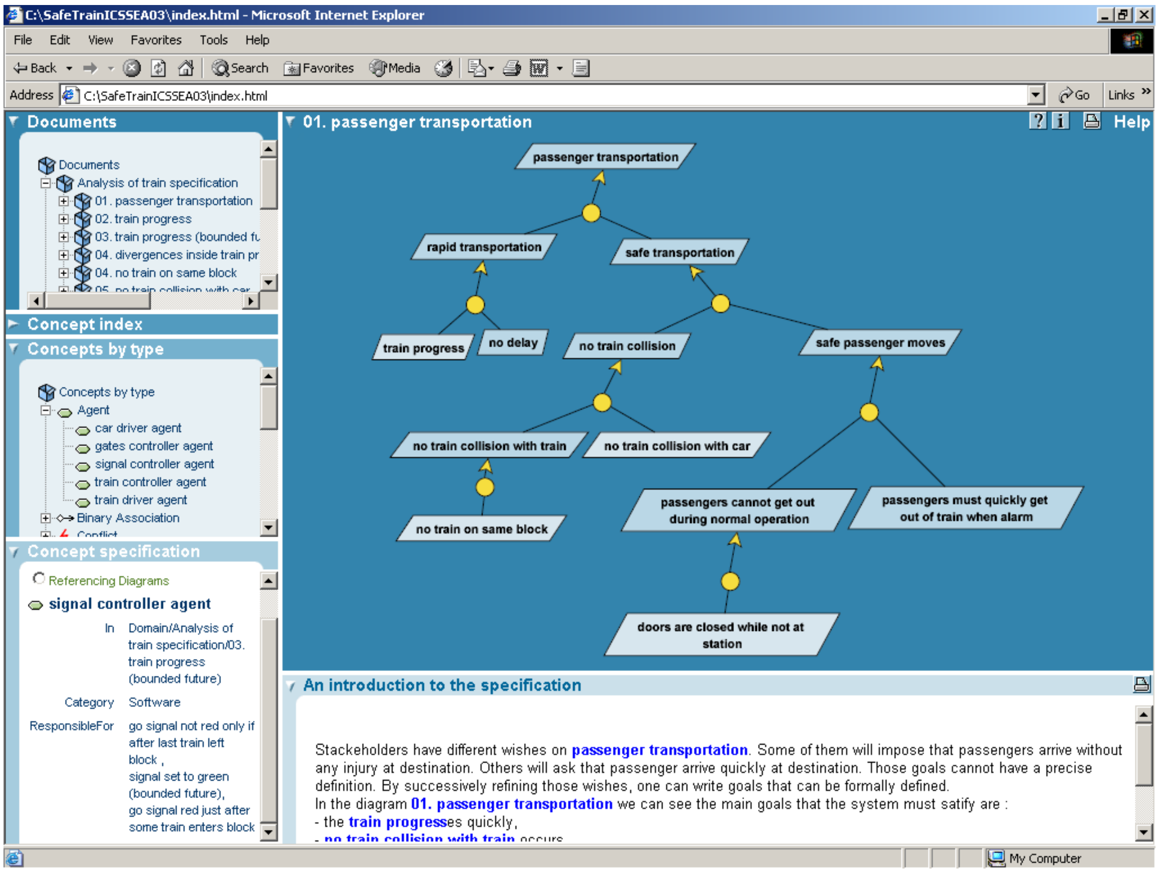
In Section 2, the KAOS goal-driven methodology will be succinctly presented. Next, in Section 3, presents how the research results on the methodology have been implemented into a tool called **Objectiver**. Two specific extensions to **Objectiver**, supporting the formal aspects of the method, are presented in Section 4. The last section presents the benefits obtained by the use of the tools and the methodology in commercial industrial projects.

2. The KAOS goal-driven methodology

The KAOS goal-driven methodology [1] is based on a rich framework for requirements elicitation, analysis and management.

The aim of the KAOS method is to provide a constructive assistance during the requirements engineering activity, starting from the elicitation of the objectives of the system and its integration into the environment, and ending with the formal definition of the specifications of the most critical parts of the system. To keep the method as close as possible to the way the stakeholders communicate and analyse their needs, KAOS is based on a goal-oriented process: goals (called objectives hereafter) are easy to understand and communicate, describe the problem instead of the solution, can be refined at will to different levels of abstraction and allow a local and incremental analysis process while the global consistency is strongly under control.

The KAOS method relies also on the tight integration of four complementary views describing not only the future system and its environment, but also the existing system and its environment. The four models are seamlessly integrated into one formal model and expressed using one formal language. The following list details the four complementary views.


C:\SafeTrainICSSEA03\index.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\SafeTrainICSSEA03\index.html
Go Links
Documents
Documents
Analysis of train specification
01. passenger transportation
02. train progress
03. train progress (bounded fu
04. divergences inside train pr
04. no train on same block
Concept index
Concepts by type
Concepts by type
Agent
car driver agent
gates controller agent
signal controller agent
train controller agent
train driver agent
Binary Association
Concept specification
01. passenger transportation
Help
passenger transportation
rapid transportation
safe transportation
train progress
no delay
no train collision
safe passenger moves
no train collision with train
no train collision with car
no train on same block
passengers cannot get out during normal operation
passengers must quickly get out of train when alarm
doors are closed while not at station
An introduction to the specification
My Computer

Referencing Diagrams

signal controller agent

In: Domain/Analysis of train specification/03. train progress (bounded future)

Category: Software

ResponsibleFor: go signal not red only if after last train left block , signal set to green (bounded future), go signal red just after some train enters block

Stackeholders have different wishes on **passenger transportation**. Some of them will impose that passengers arrive without any injury at destination. Others will ask that passenger arrive quickly at destination. Those goals cannot have a precise definition. By successively refining those wishes, one can write goals that can be formally defined.
In the diagram **01. passenger transportation** we can see the main goals that the system must satify are :
- the **train progress**es quickly,
- **no train collision with train** occurs

FIGURE 1. Objectiver web-based documentation of a goal refinement analysis.

1. *The view on the objectives.* The objectives can be refined into sub-objectives, allowing one to go from an abstract description to a more concrete one. The parent objective is explicitly linked to its children objectives through a "refinement" link (Figure 1). In this view links existing between objectives and obstacles that can occur against objectives can also be displayed.

2. *The view on the application domain objects.* This view describes the objects, relationships, and events of the system and of the environment. It is compatible with UML object diagrams in many points. A difference is that the application domain properties are expressed in the temporal logic instead of OCL. (OCL will be used when its formal semantics will be introduced in the UML reference documents.)

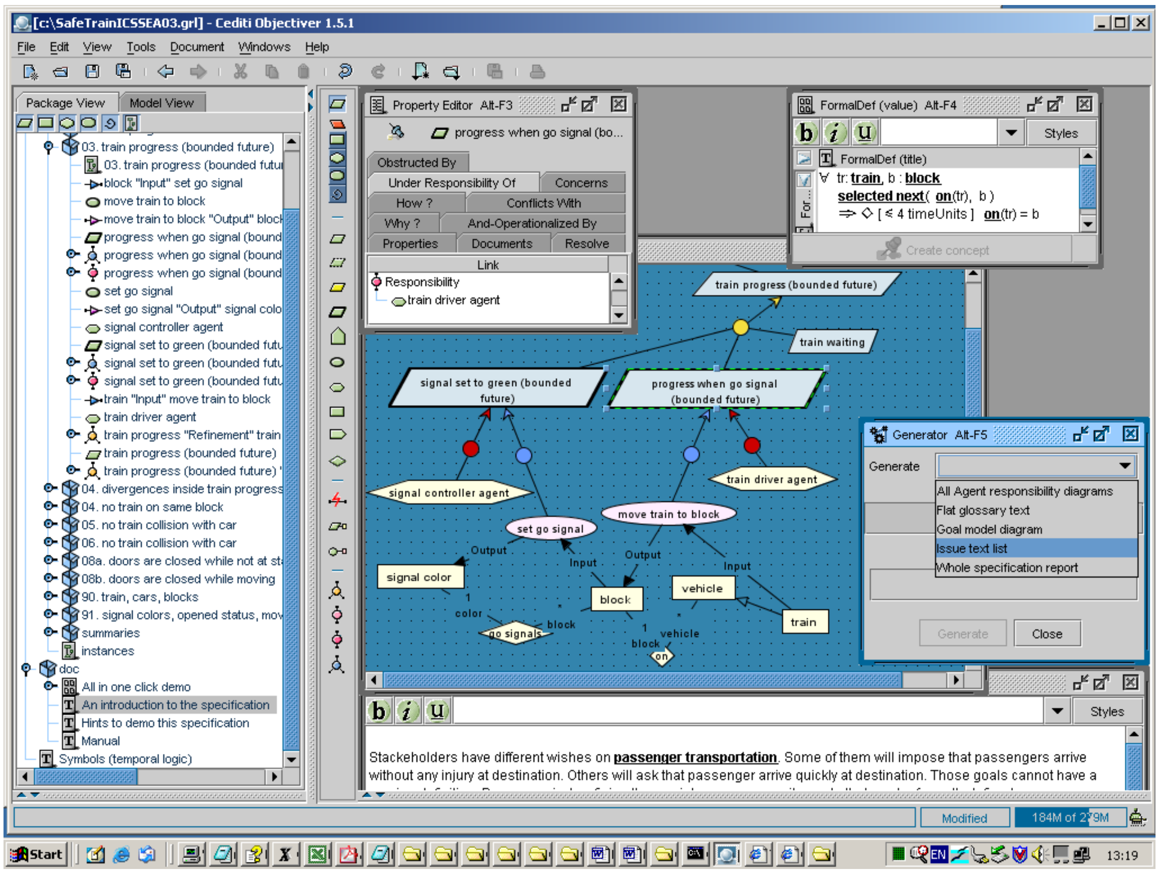
3. *The view on the agents.* This view describes software and human agents of the system and of its environment. The responsibilities and capabilities of each agent are modelled through the use of the "responsibility" link between an agent and a requirement that must be made fulfilled by this agent, and the "capability" link between an agent and operations it can perform. Each requirement must be under the responsibility of exactly one agent.

4. *The view on the operations.* Objectives are eventually refined into operational software requirements. The formalism used to describe the operations is similar to a pre/post-conditions based formalism. In this view the specific requirements which are fulfilled thanks to some pre/post conditions are explicitly linked together through an "operationalization" link.

3. The OBJECTIVER environment

To address the increasing interest in the methodology by industrial partners, a complete environment supporting the methodology[2], is currently being developed. It contains the same components found in most IDE tools :

- a graphical editor to represent the concepts and their relationships,
- a text editor allowing the analyst to record interview summaries or to associate descriptive texts to diagrams,
- an attribute editor to specify predefined attribute values or user-defined attribute-value pairs
- an explorer to retrieve diagrams, text documents, and concepts by names, types or occurrences
- an instant cross-reference navigator to go back and forth through all traceability and reference links existing between concepts or documents



03. train progress (bounded future)
03. train progress (bounded futu
block "Input" set go signal
move train to block
move train to block "Output" block
progress when go signal (bound
progress when go signal (bound
progress when go signal (bound
set go signal
set go signal "Output" signal colo
signal controller agent
signal set to green (bounded futu
signal set to green (bounded futu
signal set to green (bounded futu
train "Input" move train to block
train driver agent
train progress "Refinement" train
train progress (bounded future)
train progress (bounded future) '
04. divergences inside train progress
04. no train on same block
05. no train collision with car
06. no train collision with car
08a. doors are closed while not at st
08b. doors are closed while moving
90. train, cars, blocks
91. signal colors, opened status, mov
summaries
instances
doc
All in one click demo
An introduction to the specification
Hints to demo this specification
Manual
Symbols (temporal logic)

Stackeholders have different wishes on **passenger transportation**. Some of them will impose that passengers arrive without any injury at destination. Others will ask that passenger arrive quickly at destination. Those goals cannot have a

The most interesting functionalities of the tool are based on the specific characteristics of the goal-oriented methodology :

- a hypertext documentation generator; the produced hypertext documents allow one to inspect the entire model with a Internet browser tool in a very user-friendly way based on the rich traceability links that exist between goals, constraints, domain properties, object, agents and operations
- a script language used to define checks about the deviation from company specific quality standards and to customize automatically generated textual or graphical views on those deviations
- a powerful report generator that allows you to automatically produce reports satisfying standards about requirements documents (e.g. IEEE-830), or company specific standards by simply defining different templates, and based on the rich traceability links derived from the goal-oriented structure of the requirements.

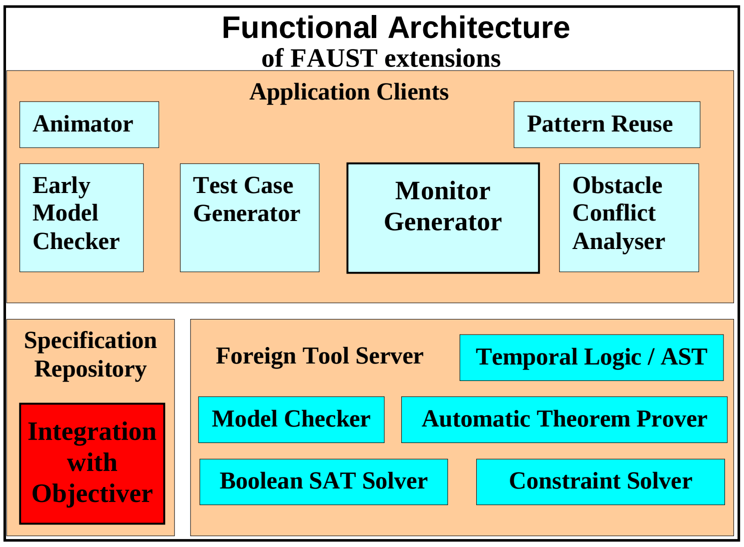
The **Objectiv*er*** tool is a meta case tool: the KAOS methodology is not hard-coded in the tool. Evolution with existing standards or customisations to company specific methodologies of the tool are easy to implement. Specifications are saved in XML format, including diagrams (W3C SVG format) and reports (W3C FOP format). In particular, the rich traceability links between requirements can be easily feed to other CASE tools.

The **Objectiv*er*** tool has been successfully used on major platforms (Windows, Linux) as it is implemented in Java. Foreign components can be tightly integrated through the meta-model based Open API. In particular, the integration with world-wide known CASE tools has been successfully tested. In summary, extensions and integration can be achieved in three ways:

- by exchanging data in XML format ;
- by interfacing with the meta-model based Open API ;
- by querying the tool with OQL (ODMG standard of object-oriented query language).

4. The FAUST extensions for formal analysis

In order to address the high quality assurance of safety-critical applications, extensions are provided to the **Objectiv*er*** tool allowing the analyst to formally analyse the requirements.The goal-oriented methodology produces not only the specifications of the operational model that must be implemented, but also all high-level properties (mostly goals) that must be verified by the operationalimplementation. With this extension the properties, the specification and the operational model are formally described with a first-order linear real-time temporal logic. Moreover, the formal aspects can be exporter to third parties formal tools with the model-based Open API of **Objectiv*er***. This makes **Objectiv*er*** an excellent platform for structuring formal models in a user-readable way and exchanging the formal models with the most powerful third parties formal tools.



The benefits of the tight integration with **Objectiv*er*** are :

- a seamless integration of formal extensions with the informal descriptions made with **Objectiv*er***, leaving to the analyst the decision to formalize only the parts of the requirements which are identified as critical.
- the splitting of the formal requirements into small user-friendly manageable parts so that existing formal technology such as automatic theorem provers, model-checkers, SAT-engines, constraint-solvers, etc. can be used automatically without user interaction.
- minimize the formal modelling activity through the implementation of the constructive aspects of the goal-oriented method, e.g. by reusing pattern libraries or by generating conflicts or obstacles that may exist, or by the generation of state-machines and their animations based on domain-level representations, for easier understanding and user validation.

The toolbox [4] will be composed of different formal modules interacting with each other. The “Pattern Reuse” module aims at helping the analyst to reuse formal decomposition patterns. An extensible library of formal patterns will be provided. Formulas will be automatically generated and checked when a pattern has been selected. The “Test Case Generator” will generate automatically tests suites covering all properties identified during the formal analysis. In particular, all goals and obstacles will be covered by the tests suites. The “Obstacle/conflict Analyser” will try to generate incrementally all possible formulas representing obstacles or conflicts that can exists in the goal model. From the goal and the operational model , the “Monitor Generator” will produce automatically the code of a system monitoring the possible violation of goals made by the target system implementing the specification and whose code is not formally verified. The last two modules are the ones currently under development :

- *The early requirements validation and verification module*.

The purpose of this tool is twofold. First, to validate the formulas in order to make sure that they represent accurately the informal description of the concepts, and second, to verify the correctness of the model.

The validation is done through the automatic creation of scenarios showing conflicts, obstacles, or simply expected good behaviors of the system and its environment in a user-friendly way, hiding the formulas. This helps to elicit the boundaries of the environment that must be represented and to validate the boundaries between the sub-systems and the environment. For the verification phase, this tool can automatically show inconsistencies between parts of the specification, e.g. between the operational model and the goal model. The errors are also presented by traces using the vocabulary of the stakeholders and are easily understood because they only concern a few properties at a time. All verifications are compositional in nature: they can be made incrementally in the background during the modelling activity and without the need of user interaction. With this tools, the analyst will obtain the formal description of high level properties (e.g. safety or liveness properties) that must be satisfied by the specification, the formal specification and its correctness wrt. the properties, and the formal description of an operational model implementing the specification and its correctness wrt. the specification.

- *The animator of operational specification*. The animation allows stakeholders to manually create traces and navigates through them easily thanks to the customized domain dependent 2D rendering of the traces of the system. The animator produces automatically the finite state machine, and automatically checks if the properties are satisfied in the traces.

5. Industrial experience report

The experience acquired from numerous industrial studies performed by CEDITI [3] at the semi-formal level has shown that the KAOS framework is highly efficient when carrying out requirements analyses, devising IT master plans or producing strategic analyses.

The following table summarizes the kinds of projects already realized.

Publishing	Reqs for a complex copyright management system, for Media Sales, Distribution & Advertising Management
Aeronautics	Requirements traceability and safety-critical analysis for Air Traffic Control Procedures
Drugs Industry & Distribution	Strategic analysis; Reqs for an e-learning system
Telecommunications	Requirements re-engineering of a cable telephone system
Language Industry	Requirements for Web-based professionnal and on-the-fly translation tools
Hospitals	IT plan, Requirements for standard clinical reporting
Bulldozer Factory	Finite scheduling optimisation

Requirements documents (typically 100 to 150 pages long) produced by the method and its associated tools are IEEE 830 standard compliant.

6. Conclusions

Formal methods are often advocated for drastically improving software quality because errors are pruned when specifications are proved to fulfil the desired properties of the future system. The KAOS goal-driven methodology is an answer to the difficulties found in the application of formal methods : it provides constructive guidance for finding the properties of the system and incrementally deriving the specification by local analysis. The **Objectiv*er*** environment can be used to collect, structure the description of the system and the properties it must satisfy. Moreover the tracing facilities between properties and specifications, and the automatic generation of specific reports help the analyst to rigorously reason and make decision on the requirements. Finally, on the one hand, the KAOS methodology allows the analyst to give a complete formalisation of the specification. On the other hand, the FAUST extensions of the **Objectiv*er*** environment use some of the best formal tools and allow formal aspects to be completely hidden behind the stakeholder view on the requirements.

Acknowledgment. The Kaos approach results from several research projects led by Prof. A. van Lamsweerde (University of Louvain) and funded by the European Union, the Belgian and the Walloon governments. The **Objectiv*er*** commercial tool is developped by Jean-Luc Roussel, Denis Ballant, Christophe Belpaire, Denis Genard, Philippe Legrain, Cédric Nève and Alain Vanbrabant, with the significant contribution of André Rifaut for providing requirements, feedback, and prototypes for the tool. The work on FAUST extensions is supported by the EU (FEDER and FSE) and the Walloon region (DGTRE). Kaos analyses that have proved the benefits of the approach have been led by the authors of this paper. Several PhD theses have contributed to explore advanced features eventually deployed in the tool ; ongoing PhD theses already give promising results.

7. References

[1] A. van Lamsweerde: Building Formal Requirements Models for Reliable Software. Invited Paper Ada-Europe 2001, Leuven, May 14-18, 2001 LNCS, Vol. 2043, Springer-Verlag

A full bibliography of publications about the method can be found at URL http://www.info.ucl.ac.be/research/projects/AVL/ReqEng.html.

[2] This tool can be downloaded from http://www.objectiver.com

[3] Industrial experience report with Objectiv*er* at Cediti can be found at URL: http://www.cediti.be/download/DossierGrailKaosAnglais.pdf

[4] FAUST formal modules screen snapshots and videos : http://www.cetic.be/~faust/toolbox/FME03-snapshots.html.