

# FAUST : Formal Analysis Using Specification Tools

A. Rifaut, P. Massonet, J-F Molderez, C. Ponsard, P. Stadnik  
CETIC research center  
{ari,phm,jfm,cp,ps}@cetic.be

Axel van Lamsweerde, Tran Van Hung  
Université catholique de Louvain  
{avl, tvh}@info.ucl.ac.be

## Abstract

*Developing high quality requirements specifications is a necessity for a number of critical industrial systems. An integrated toolset, called FAUST, is proposed to assist in the production of such specifications based on the KAOS goal-driven methodology.*

*The tool suite is designed to naturally extend the existing semi-formal modeling framework and to allow formalizing only the relevant critical parts. Two tools from the toolset are presented. The requirements checker performs KAOS goal-level checks using existing model checking technology. The requirements animator produces domain-level animations hiding formality even further.*

## 1. Introduction

Several studies of current software practice have shown that a large number of software bugs can be traced back to the requirements phase. Moreover those errors are difficult to discover, costly to fix and can have dangerous -sometimes life-threatening- consequences on the software environment.

The KAOS goal-driven methodology [1] has already proved a rich framework for requirements elicitation and management. High assurance systems with reactive, real-time, embedded or critical components can benefit from a goal-oriented approach. Until recently, the KAOS CASE tool support provided by the GRAIL platform was restricted to the semi-formal level of the language.

The FAUST toolbox [2] is designed to fill this gap with the following goals in mind :

- Seamlessly integrate formal and semi-formal levels allowing the analyst to formalize only the part of the requirements identified as critical.
- Provide goal-oriented formal tools by transparently mapping them onto existing formal technology such as theorem provers, model-checkers, SAT-engines, constraint-solvers, etc.
- Hide the formulas as much as possible, e.g. by reusing pattern libraries or by generating animations based on domain-level representations, allowing easier understanding and user validation.

The toolbox is composed of a number of interacting tools: the requirement checker, the animator, the pattern manager, the obstacle generator and the acceptance test generator. We will detail hereafter the two major and

currently prototyped tools: the requirements checker and animator.

## 2 The Requirements Checker

The purpose of this tool is to automatically analyze the consistency of various part of a KAOS specification using exploration techniques on an instantiated finite domain. For example, the goal sub-model can be checked for goal refinement correction and obstacle consistency wrt to domain properties; the operation sub-model can be checked as correctly operationalizing the addresses goals. Such verifications are compositional in nature: they only focus on a few properties at a time.

Several formal alternative tools are available for performing this state exploration, some of them outperforming others depending on the context. They can also be used conjointly in order to cope with a larger problem.

If assessed, the validity only holds for the given domain. If not, the produced violation trace can be used to understand the design flaw. Feeding this trace in the animator tool can help understand it more easily.

## 3. The Requirements Animator

The animator tool is based on the KAOS operational model. It is composed of the following components: (1) a state machine compiler producing statecharts descriptions from the KAOS operations descriptions, (2) an animator engine allowing one to instantiate and animate those statecharts, either interactively or using computed traces and (3) a rendering engine including a generic statecharts viewer and a toolbox for designing user-level visuals. 2D and 3D worlds are being investigated.

## 4. Future work

The FAUST toolbox is still in development. The checker and animator will soon be validated in critical industrial applications. Other tools like the test generator are in an early development phase.

## 5. References

[1] Reference material about the KAOS methodology:  
<http://www.info.ucl.ac.be/research/projects/AVL/ReqEng.html>.

[2] Project and toolbox status: <http://www.cetic.be/~faust>.