least one cycle (in fact, in the last cycle of sequences built by GATeL) are stated by reach directives. To build a sequence reaching the test objective according to the Lustre model of the program and its environment, these three elements are automatically translated into a constraint system. A resolution procedure then solves this system through alternate propagation and labelling phases. Propagation checks the local coherence of the system, while labelling aims at incrementally eliminating the constraints by the choice of a variable and a value within its authorised domain.

The random aspect of this resolution procedure implies that the input domains are not fairly covered, meaning quite distinct sequences may be generated for the same objective (for instance, different ways to raise an alarm). A second step in the definition of a selection strategy is to help GATeL to distinguish these sequences. This can be achieved by splitting the constraint system so that each sub-system characterises a particular class of behaviour reaching the objective. This splitting can be processed interactively either by applying predefined decompositions of boolean/integer/temporal operators in the Lustre expressions corresponding to the current constraint system, or by declaratively stating through a dedicated split directive the various behaviours one wants to observe. Notice that a systematic unfolding of different classes of operators would lead to the usual structural coverage criteria.

Finally, test submission consists in reading input sequences generated with GATeL, computing program outputs, and then comparing these values to the expected ones evaluated during the generation procedure. When the program has not been automatically generated from the Lustre model, this gives an automatic oracle. For the alternative case, the truth value of the test objective can play the role of a partial oracle. Methodological and efficiency aspects of GATeL are still under development. However, it has been successfully applied implemented in on industrial case studies, and has generated sequences of a thousand cycles, after the resolution of several thousand constraints over more than forty input flows.

Please contact:
Bruno Marre, Benjamin Blanc
Commisariat à l'Energie Atomique, France
Tel: +33 1 6908 5416
E-mail: benjamin.blanc@cea.fr

Cow_Suite: A UML-Based Tool for Test-Suite Planning and Derivation

by Francesca Basanieri, Antonia Bertolino, Gaetano Lombardi, Giovanni Nucera, Eda Marchetti and Alberto Ribolini

The Cow_Suite tool provides an integrated, practical and tool-supported approach to the strategic generation and planning of Unified Modeling Language (UML)-based test suites for large industrial software systems. The tool can be adopted from the early stages of system analysis and modelling, and uses the UML diagrams developed for analysis and design without requiring any additional formalism or specific ad-hoc effort for testing purposes.

The Cow_Suite (COWtest pluS UIT Environment) approach supports the early generation of high-level strategic test plans, which can be outlined during the early phases of software life cycles and continuously refined and updated throughout development as the design evolves. Cow_Suite combines two original components: a method to derive test cases, called UIT (Use Interaction Test), and a strategy for test prioritization and selection, called Cowtest (Cost-Weighted Test Strategy). These two components work in conjunction as Cowtest helps decide which and how many test cases should be planned within the universe of test cases that UIT could derive for the entire system under consideration.

The tool was developed at ISTI-CNR in collaboration with Ericsson Lab Italy (ERI, Rome), within the framework of the PISATEL inititative.

Cow_Suite Features

With respect to the many UML-based test approaches already proposed, the main innovative features of Cow_Suite can be summarised as:

- usability: there is no need to augment the UML specifications with specific annotations to facilitate test derivation, nor to translate the UML diagrams into an intermediate notation that the testing methods can process. Cow_Suite adapts to the modelling notations and procedures in use, and not vice versa.
- timeliness: while other methods require a complete and fairly detailed set of UML diagrams, Cow_Suite can begin to outline a test plan from the early stages of software development. Naturally the plan will be as abstract as the diagrams being processed.
- incrementality: Cow_Suite has been conceived for system and integration testing, typically conducted in an incremental fashion, considering progressively larger parts of the system and addressing, at each step, the relevant functionalities and interactions for that level.
- scale: Cow_Suite trades thoroughness for comprehensiveness: it addresses UML-based testing of real-world systems in a practical, efficient way.

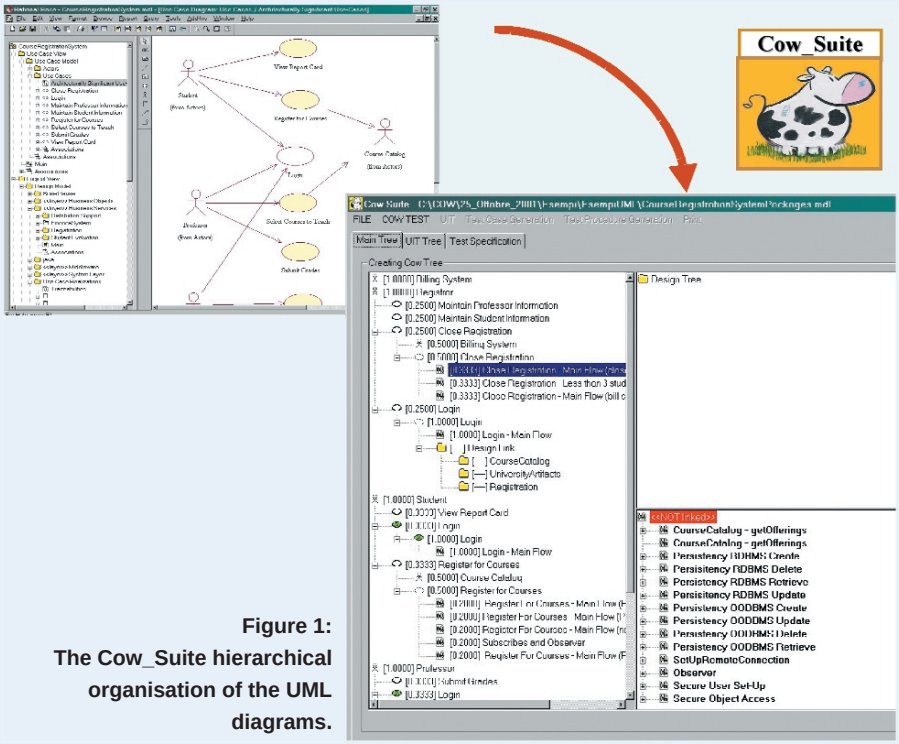

Figure 1: The Cow_Suite hierarchical organisation of the UML diagrams.

The combined usage of Cowtest and UIT makes it possible to derive a feasible number of test cases while keeping the coverage of functional areas as wide as possible.

Cow_Suite Usage

The Cow_Suite tool has been designed to be compatible with the Rational Rose tool from which it retrieves the required information. In particular, it employs the Use Case, Sequence, Communication and Class diagrams.

The tool execution starts by importing information on the UML design elements and organising it in a sort of hierarchy, whose root is represented by an Actor and leaves by Sequence or Communication Diagrams (see Figure 1). This hierarchy provides the user with a complete view of the status of the functionality specification and up-to-date documentation on Use Cases and their Realisation, Sequence and Communication Diagrams associated with each specification level, the reused nodes and those elements not linked with the other parts of the design.

At this point, users can annotate each node with a specific weight, representing the relative 'importance' of this node with respect to the other nodes at the same level, and choose between two supported test strategies: a fixed number of test cases, or fixed functional coverage.

Users can then decide the integration level at which the test suite should be derived (or which of the elements of the UML model should be tested), by simply highlighting a portion of the hierarchy.

The UIT component will then automatically derive a list of test cases on the basis of the UML diagrams corresponding to the chosen integration level. These will be specified with a granularity corresponding to the degree of detail at which the considered diagrams are modelled. In Figure 2 we show a schema of the process adopted by the tool.

Case Study

The Cow_Suite tool has already been applied to several case studies. In one industrial case study we compared the UIT-derived test plan with an existing test plan (called ERI). The ERI test plan had been developed manually, following the standard in-house procedures at Ericsson, based mainly on the testers' skill and their knowledge of the system. The UIT test plan was instead derived automatically at ISTI-CNR, using only the available UML design diagrams. The purpose of the comparison was to evaluate the main advantages of the UIT test plan in terms of cost, schedule and test strategy selection. The following advantages emerged from experimentation with the tool:

- derivation of a detailed test plan in advance with respect to the testing stage; this plan can be used as a baseline for deciding the most appropriate test selection strategy
- a realistic evaluation of the requirements and functional coverage that can be reached (timely implementation of

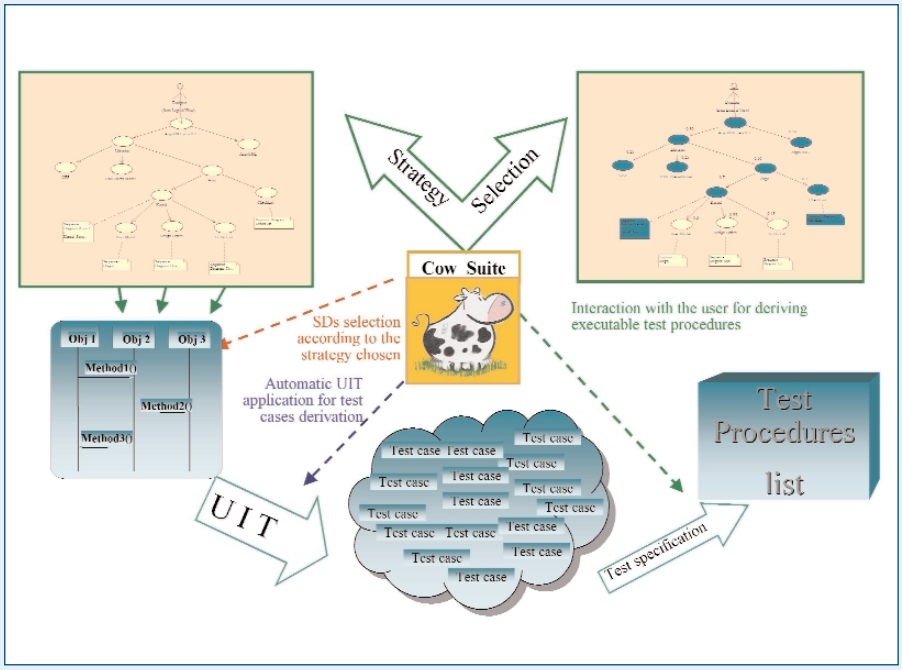

Figure 2: The Cow_Suite usage schema.

corrective actions or different choices for the test strategy are possible)
- reduction of the time necessary for test plan derivation (the same level of requirement coverage was obtained in a quarter of the time).

On the negative side we observed that the automatic derivation of test cases failed to include exceptional test cases, that is, test cases to handle abnormal system behaviour. This is reasonable and suggests that as good practice an expert should check the automated test plan before deployment to cover special situations.

Further experimentation is of course desirable, and Cow_Suite is freely available in a prototype version for public usage.

Link:
http://www.isti.cnr.it/ResearchUnits/Labs/se-lab/software-tools.html

Please contact:
Eda Marchetti, ISTI-CNR, Italy
Tel: +39 050 315 3467
E-mail: eda.marchetti@isti.cnr.it

Three Countries' Offensive towards Testing with Advanced Languages

by Wan Fokkink, Matti Kärki, Jaco van de Pol, Axel Rennoch, Ina Schieferdecker and Markus Sihvonen

The need for appropriate means to test systems and software is still alive! Even though a huge number of test tools are on the market, series of conferences on testing continue to be held, and an international standard on testing methodology has existed for more than ten years. Most industries are still looking for ways to make their testing process more effective, efficient and understandable, to strengthen confidence in their products and services. Three European countries, Finland, Germany and the Netherlands, have started a common offensive to establish a basis for the industrial application of tests and testing methodologies with advanced languages. Independent national funding sources have been brought together for the TT-Medal project under the supervision of the Information Technology for European Advancement (ITEA) association.

The goal of TT-Medal is to develop a test architecture that covers the whole life cycle of a product, starting from the initial specifications and ending up with regression testing during the maintenance phase. Obviously this goal can only be reached by a testing methodology that addresses tests at both an abstract and an executable level.

In TT-Medal key roles are assigned to international standards, the Testing and Test Control Notation (TTCN-3) by ETSI, the Universal Modeling Language (UML2.0) and its testing profile by the OMG. The TT-Medal strategy covers a systematic testing methodology including the production of TTCN-3 based tests from UML models and the development of a generic test infrastructure and test architectures for TTCN-3 with open interfaces for test execution and management. Special emphasis is given to the reuse of test cases between testing phases and for different products in the same domain. Three ERCIM research institutes, CWI, Fraunhofer FOKUS and VTT Electronics, are the driving force behind these innovations.

An important aim of TT-Medal is to develop a test platform based on TTCN-3 for major European industries such as automotive, railways, telecommunication and embedded systems. Four industrial partners in the project, DaimlerChrysler (Ge), ProRail (NL), Nokia (Fi,Ge) and LogicaCMG (NL), will apply the project results with the objective of unifying the various test environments in use at the different companies, and making seamless the transition from one environment to another.

Furthermore, tool support will be developed for automatic test-case generation, validation and deployment, and to support the reuse of test cases. Three tool developers, Conformiq (Fi), NetHawk (Fi) and Testing Technologies (Ge), are responsible for this part of the project. Finally, a complete training package will be developed by Improve QS (NL), and will enable European industries to integrate the new improved test practices into their development processes.

The strong application of the standardised and formalised language constructs of TTCN-3 and UML2.0 in mainstream industry provides a special challenge to the customers and providers of test products and services. Coupling the development of system models and tests helps to optimise the rapid implementation of test solutions. Combining the potential of UML2.0, the maturity of TTCN-3 and the advice of industrial manufacturers is an exciting process.

The project work is organised into five work packages, which address issues on methodology, test development tools, test execution tools, industrial case studies, and dissemination and standardisation work. There are strong relationships such as major requirements from