types that augment a virtual facility with energy consumption information along with spacial characteristics, location and status of equipment, while providing simple ways to control them. Unlike previous work that has largely relied on CAD or VRML for scene generation, this work takes advantage of recent game engine technologies for fast and real-time rendering of feature-rich representations of the facility, along with spacial information, equipment conditions and device status.

One of the above-mentioned tools has been rolled out commercially. Our vision is to create a platform for data integration and visualization that enables easy and efficient ways for unskilled users to explore energy data. We believe that a better, integrated ICT solution for handling energy data may translate to faster and more effective energy management through more accurate energy usage diagnostics and just-in-time corrective action that will eventually translate to a more rational usage of energy.

References:
[1] J. Granderson, M. A. Piette, G. Ghatikar, P. Price: “Building Energy Information Systems: State of the Technology and User Case Studies”, technical report, LBNL-2899E, Environmental Energy Technologies Division (2009)
[2] P. Fite-Georgel: “Is there a reality in Industrial Augmented Reality?”, in proc. of ISMAR’11, IEEE Computer Society, Washington, DC, USA, 201-210 (2011)
[3] Viet Toan Phan and Seung Yeon Choo: “Using AR for real-time crosscheck of ventilator ducts at worksite”, in proc. of VRCAI’10, ACM, New York, 293-298, 2010.

Link:
http://www.inesc-id.pt

Please contact:
Paulo Carreira, Alfredo Ferreira
INESC-ID Lisboa and IST/Technical University of Lisbon, Portugal
Tel: +351 214 233 287, +351 214 233 512
E-mail: paulo.carreira@ist.utl.pt, alfredo.ferreira@ist.utl.pt

VMC: A Tool for the Analysis of Variability in Software Product Lines

by Maurice ter Beek, Stefania Gnesi and Franco Mazzanti

Researchers from the Formal Methods and Tools group of ISTI-CNR have developed a tool for the computer-aided verification of behavioural variability in product families.

During the last decades, we have witnessed a paradigm shift from mass production to mass customisation in an attempt to serve as many individual customer's needs as possible. A typical example is the production of mobile phones. Software Product Line Engineering (SPLE) translates this paradigm into a software engineering approach aimed at developing, in a cost effective way, a variety of software-intensive products that share an overall reference model, ie that together form a product family. Usually, commonality and variability are defined in terms of features, and managing variability is about identifying variation points in a common family design and deciding which combinations of features are to be considered valid products.

Feature models are widely used for variability management in (S)PLE. A feature model provides a compact representation of all the products of a product family in terms of their features. Figure 1 shows an example feature model inspired by the mobile phone industry, adapted from [1]. The features are represented as nodes in a tree, with different relationships between them. We see that all mobile phones must run software to support calls and to display information on either a basic, a colour or a high resolution screen. The software may include support for a GPS and for media devices such as a camera, an MP3 player or both. Finally, mobile phones including software for a camera must also include software to support a high resolution screen, whereas software for a GPS cannot run on a basic screen.

There is a large body of literature on the computer-aided analysis of feature models to extract valid products and to detect anomalies, ie, undesirable properties such as superfluous or - worse - contradictory variability information (for instance so-called false optional and dead features). These analyses however do not take into account any behavioural variability as only the presence of the software implementing the features is considered, not their temporal ordering. Since software products are often large and complex, and many are used in safety-critical applications in the avionics, railways, or automotive industries, it is unrealistic and practically infeasible to specify - let alone verify - the behaviour of each product individually. Note that our mobile phone example would already require behavioural models for 14 largely identical products.

For this reason, we recently launched a research effort to 1) develop a formal modelling and analysis framework capable of dealing with behavioural variability and 2) pro-

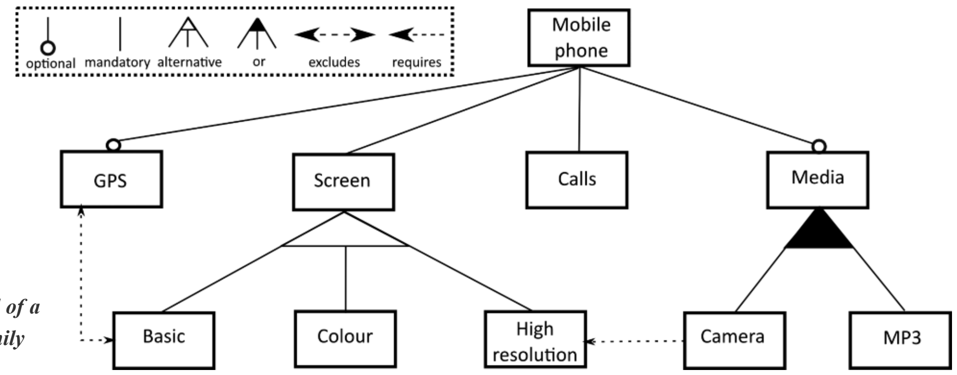


Figure 1: A feature model of a mobile phone product family

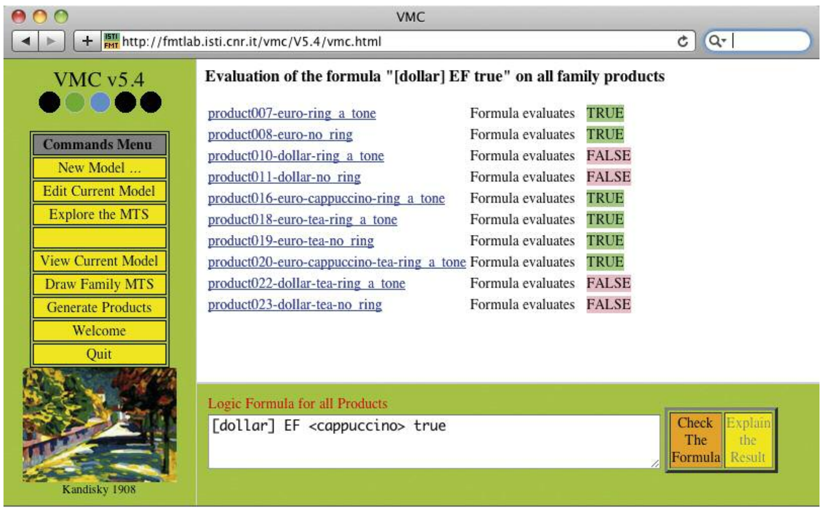
vide tools to support this framework with automated verification. This has resulted in the use of Modal Transition Systems (MTSs) as a formal method for describing the combined operational behaviour of an entire product family. An MTS is a Labelled Transition System (LTS) that distinguishes between optional (may) and mandatory (must) transitions. Since MTSs cannot model all variability, we enrich them with an additional set of variability constraints.

We have also developed a tool for the computer-aided verification of behavioural variability in product families, namely VMC (Variability Model Checker). VMC takes as input a high-level description of an MTS together with a set of textual constraints. In our mobile phone example the latter would include “GPS EXC Basic” and “Camera REQ High resolution”.

VMC allows the user to interactively explore an MTS of a product family; model check properties (branching-time temporal logic formulae) over an MTS; visualise the (interactive) explanations of a verification result; generate one, some, or all of the family's valid products (represented as LTSs); browse and explore these; model check whether or not products (one, some, or all) satisfy certain properties; and, finally, help the user to understand why a certain valid product does or does not satisfy specific verified properties by allowing such a product to be inspected individually. Figure 2 shows VMC's capability to model check a temporal logic formula over all valid products of a family.

The core of VMC consists of a command-line version of the model checker and of a product generation procedure. These programs are stand-alone executables written in Ada that can easily be compiled for the Windows, Linux, Solaris, and Mac OS X platforms. These core executables are wrapped with a set of CGI scripts handled by a web server, making it easy to build an html-oriented GUI as well as to integrate graph drawing tools. The development of VMC is ongoing, but a prototypical version is publicly usable online (see the link below) while its executables are available upon request.

Currently, VMC is not targeted at very large families. Its main limitation, however, is the generation of the model from its input language, while its on-the-fly verification engine and advanced explanation techniques are those of the highly optimized family of on-the-fly model checkers developed at ISTI-CNR over the last few decades for verifying formulae in action and state-based branching-time temporal logics derived from the CTL family of logics. The on-the-fly nature of VMC means that in general it is not necessary to generate and explore the whole state space. This feature improves its performance and allows it to deal with infinite-state systems.



Evaluation of the formula "[dollar] EF true" on all family products

product007-euro-ring a tone	Formula evaluates	TRUE
product008-euro-no ring	Formula evaluates	TRUE
product010-dollar-ring a tone	Formula evaluates	FALSE
product011-dollar-no ring	Formula evaluates	FALSE
product016-euro-cappuccino-ring a tone	Formula evaluates	TRUE
product018-euro-tea-ring a tone	Formula evaluates	TRUE
product019-euro-tea-no ring	Formula evaluates	TRUE
product020-euro-cappuccino-tea-ring a tone	Formula evaluates	TRUE
product022-dollar-tea-ring a tone	Formula evaluates	FALSE
product023-dollar-tea-no ring	Formula evaluates	FALSE

Logic Formula for all Products

```
[dollar] EF <cappuccino> true
```

Figure 2: Verification result of a property for all products of a family of coffee machines

Link:
VMC: http://fmt.isti.cnr.it/vmc/

References:

[1] D. Benavides, S. Segura, and A. Ruiz-Cortés: “Automated analysis of feature models 20 years later: A literature review.” Information Systems 35(6) pp. 615-636, 2010. http://dx.doi.org/10.1016/j.is.2010.01.001

Please contact:

Maurice ter Beek
ISTI-CNR, Italy
E-mail: maurice.terbeek@isti.cnr.it