Software Automation meets Interactive Media Development
by Dirk Deridder, Thomas Cleenewerck, Johan Brichau and Theo D'Hondt

Developing the software component of interactive media requires an advanced set of
development tools and enforces a different view on software development in
general. This is mainly due to the specific characteristics of the environment in
which this development takes place. We propose a combination of software
automation techniques to counter this effectively. Since the iMedia domain is in
continuous flux, and these technologies are mostly designed for stable domains, the
evolvability of the approach is guaranteed by rooting it into the heart of the system.

Media production companies, in particular media broadcasters, are being challenged
by their consumers to produce media in which the consumers can actively participate
and interact with TV shows and between peers. This new form of media, called
interactive media, is a combination of traditional media and a behavioral component
(software). Examples of such components are online games, quiz software, virtual
community worlds etc.

The challenge lies in the production of the behavioral component of the new form of
media within the broadcasting environment. The production cycle is constrained by
the imposition of extremely strict broadcasting times, the extremely short time-to-
market situation caused by last-minute changes, and the extreme deployment
prerequisites.

IMedia Software Generation System
IMedia development requires more advanced development tools. The approach we
propose combines existing research from the areas of generative programming,
transformation systems and domain engineering. This results in a system that is best
described as an iMedia Software Generation System (IMSGS; see Figure 1),
specialized for each product range. In an IMSGS, more autonomy and flexibility is
given to the media producer to adapt the iMedia software product. This is achieved
by generating different tailor-made ‘instances’ of the product range, given a high-
level specification. The tailoring of particular instances is managed by the media
producer (the domain expert).

Evolution of the IMSGS
Our research focuses on the evolvability problems of a system that is based on DSL
and generative programming technology. The system and the techniques used to
build the system were designed with evolution in mind: the impact of changes to the
system is limited to individual easily identifiable modules.

An IMSGS for a specific product range is divided into a set of program generators,
each targeted at a specific concern in the product (eg the application logic, the
graphical user interface etc). For each concern a domain model with CoBro is
constructed that defines the concepts in the domain and the relationships between
them. These models are then used to construct concise domain-specific languages
(DSLs) compliant with the definitions in the domain model. The DSL compilers are
program generators implemented in Linglet Transformation System (LTS), which
translate the DSL specifications into executable code in some generic language using
generic libraries and frameworks. Finally these program generators are composed
using Generative Logic Meta-Programming (GLMP) in order to integrate each of their
generated program parts into one application.
Figure 1: Overview of the IMedia Software Generation System with the major evolution δ’s in an IMSGS

The domain knowledge is described in CoBro. CoBro follows a concept-centric approach in which we couple the domain concepts to their corresponding implementations in the quiz language. In this way, one can start at the level of the domain concepts to estimate which parts of the implementation (δ₃ in Figure 1) will be affected by the evolution (δ₂ in Figure 1). Moreover, connecting the domain knowledge to the implementation provides a valuable source of documentation of the assumptions made by the original developers.

The DSLs are constructed using the Linglet Transformation System via a composition of language components, which is expressed in a language specification. LTS modularizes the language components by specifying the necessary communication patterns among them in a separated language specification, through the customization of the language components. Hence the dependencies among the language components become explicit and are removed from their implementations. Consequently the impact of changes in the language (δ₃ and δ₄ in Figure 1) is isolated to the language specification and to individual identifiable components.

The composition of the program generators (DSL compilers) is realized with Generative Logic Meta-Programming. GLMP features a grey-box composition model of program generators that allows the specification of integration relationships among the subparts of different program generators. This mechanism is vital for adapting the generators so that they produce program parts; these can then be combined into a single application with no undesired interferences that could break their functionality (δ₅ in Figure 1).

This research is performed in the context of the Advanced Media Project, a collaboration between Vlaamse Radio- en Televisieomroep, Vrije Universiteit Brussel, Universiteit Gent and IMEC.
Links:
http://prog.vub.ac.be/
http://www.xmt.be/sake.html

Please contact:
Dirk Deridder
Vrije Universiteit Brussel, Belgium
E-mail: Dirk.Deridder@vub.ac.be