MODEL DRIVEN ARCHITECTURE WITH CODEGENIE

MDA IN ACTION: TRANSFORMING ANALYSIS TO IMPLEMENTATION USING A PLATFORM MODEL

Software development has traditionally been labour intensive and error prone as software is hand-coded from descriptions embodied in design models. Changing technologies means that obsolescence is an increasing concern. The Object Management Group's (OMG) Model Driven Architecture (MDA) approach addresses these problems by separating the application and technology domains and translating models into software. AMS Radar Systems are using Domain Solutions' novel CodeGenie MDA Generator approach to develop a new range of radar control systems. This paper explains how Domain Solutions' CodeGenie MDA Generator operated when AMS developed an abstract UML Platform Model (PM) representing the system architecture and an associated model translator. It describes how AMS translate implementation-free application models, known as Platform Independent Models (PIM), into this Platform Model forming Platform Specific Models (PSM), from which the software is generated. The paper covers the development process and presents the measurements of productivity improvement.

Company Introductions

AMS is an equal shares joint venture between BAE SYSTEMS of the UK and Finmeccanica of Italy. The company is a major force in European defence and electronics with a turnover of over 1.2 Billion Euros and an established internal customer base in over 100 countries.

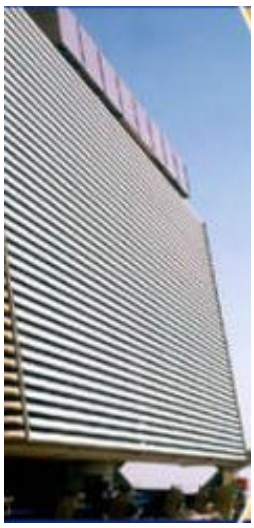
Domain Solutions specialise in the application of Executable UML software development and the use of the CodeGenie MDA code generation tool.

The Problem and Goals

The previous AMS Radar Controller project had defined an industrial strength software architecture that implemented design patterns that supported full object distribution and adhered to strict availability constraints.

The previous system was realised by software engineers repeatedly hand-coding these patterns for the application classes. Apart from being complex and labour-intensive, this introduced problems in terms of non-standard usage, misplaced focus on the architectural and software implementation ('the how') and not on the analysis of the business application ('the what'), and design-level UML models with often unreliable cost/time project metrics.

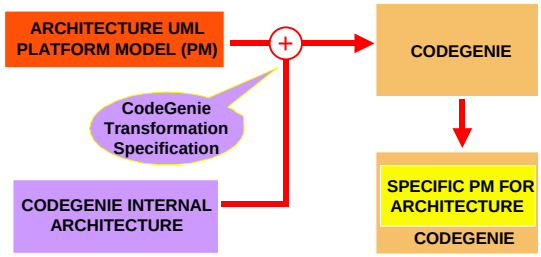
The project's goal was for the team (two architects and nine analysts) to address these problems and build a common, customisable Radar controller upon a separate architecture Platform Model. This became a necessity due to ageing hardware and COTS middleware.



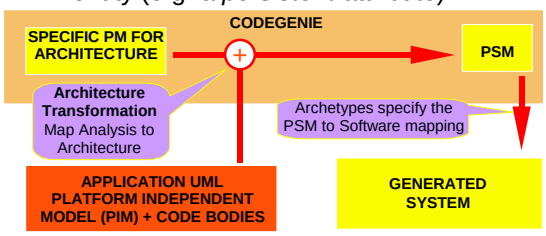
Description of the Solution Approach

The solution is based on separation of the problem from the implementation technology. This is solved using the OMG's MDA approach with the Executable UML process.

AMS selected Domain Solutions' CodeGenie product due to the innovative approach of treating the software architecture (in terms of persistence, distribution, comms etc) as a separate UML project. The diagram below shows how the architecture model is combined with the generic CodeGenie to produce a specialised CodeGenie translator:



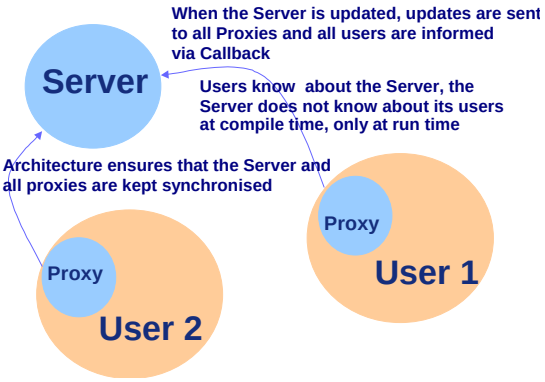
The UML Radar Platform Independent Models (PIMs) are mapped to this specialised CodeGenie Platform Model creating a Platform Specific Model (PSM). The software for the Radar Controller system is generated according to a number of archetype templates that specify the software generated for each PM entity (e.g. a *persistent* attribute).



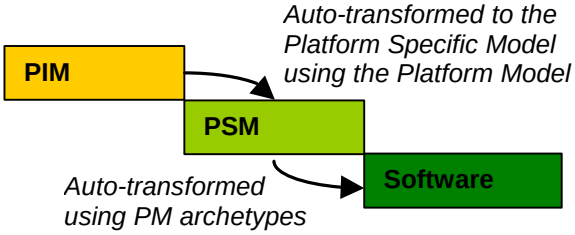
Technological Implementation
The UML Radar models were specified in the executable UML profile with the IBM-Rational Rose CASE tool. Markup tags, e.g. *persistent,* are specified as hints to the architecture mapping. Models are exported in XMI and read into CodeGenie's UML meta-model.

The architecture was also specified in the executable UML profile using Rose. The model size and complexity of the architecture project is comparable to the Radar Controller but is an additional reusable company asset. This approach means that the underlying architecture can change without significant changes to the radar controller models.

The architecture was split into a number of UML packages representing the concerns: these packages included Deployment, Object Management, Communication & Persistence. A UML class diagram for each package represented the architectural solution required e.g. object instances were managed by a factory, distribution was achieved by proxy and observer patterns. Vendor independent CORBA communication concepts were modelled for the middleware.

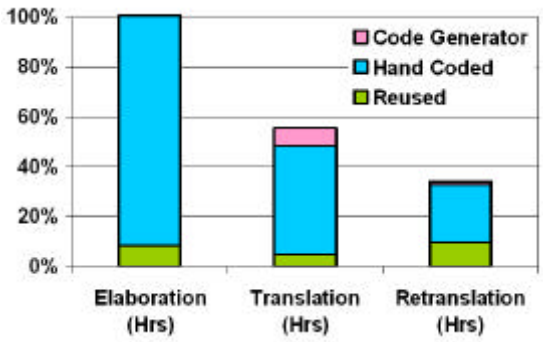


CodeGenie offers transformation between the Radar model PIM and the architecture Platform Model; this is performed via a number of standard maps. The architecture Platform Model is characterised by archetype files, written in interpreted Java known as JARC. These specify the target C++ software.



Benefits and Return On Investment (ROI)
The MDA approach aids understanding by raising the abstraction level of system description. It is beneficial to developers as it is leading edge and removes tedious coding. It is beneficial to managers as it is formal, predictive and productive. Customers can understand and specify the system due to the higher level of model abstraction. The benefits of analysis and architecture asset reuse, solution commonality and technology-shift protection provide rewards in the mid-term. The separation of concerns solves many long-term development and maintenance issues.

The following chart shows an effort comparison between hand coding (elaboration), translation and reusing the architecture and translator (retranslation).



Initial metrics show that the effort required producing a MDA project were roughly half of that of handcrafting. Reusing the architecture for future projects reduces this effort further!

MDA Considerations
The MDA approach is innovative and therefore to adopt it required a change of traditional attitudes. Developers and managers need to think differently and expect benefits in the mid-term.

Future Goals
The future goals are to deliver all Radar controllers using this development method. The project awaits an OMG standard action language before it adopts a full simulation environment. It currently uses an architecture API to insulate the behaviour from the architecture.

Summary
Modelling both the analysis and the architecture as separate concerns provides quantifiable benefits and future proofs assets.

Domain Solutions wishes to thank AMS Limited for permission to use this case study to illustrate Domain Solutions' approach to problems and goals.

For more information about Domain Solutions' CodeGenie product, contact David Pilfold:

Domain Solutions Ltd, Buckingham House, 35 Graham Road, Malvern, Worcs. WR14 2HU, UK
[t] +44 (0) 1684 578875 [w] www.ooagenerator.com [e] mda@ooagenerator.com