Challenges for the model based development of distributed real time systems

Position Paper

Michael Giddings, Jan Jürjens, Pat Allen

Computing Department, Open University,
Walton Hall, Milton Keynes, United Kingdom, MK7 6AA

Abstract. Large distributed Real Time systems are time consuming to develop. They frequently cost more than estimated and frequently overrun. Model Driven Architecture has claimed to be able to address some of the issues that cause these problems. A high level of abstraction and automatic translation between models may help. Platform Independent models for the individual components of the system mixed with scheduling information may enable functional changes and real performance to be assessed early in the development. Establishing different views from the requirements repository may better fit the development engineers skill and reduce errors. This is a position paper that discusses current challenges for the model-based development of distributed real time systems and how they might be overcome.

1 Introduction

Distributed real time systems are systems with at least two processing units with associated sensors, emitters, actuators and displays. In larger systems they can consist of hundreds of devices. They are real time because timeliness, performance and schedulability are essential to correctness of operation.

Originally computers were used to generate design documentation for these systems that was printed out and used as reference documents. Each document was independently generated using a word processor or graphical tool. In the past functional representation tools, such as CoRE [6] were used to identify independent functional processes. More recently UML based object oriented modelling systems were used such as StateMate [7]. With the advent of computer networks where every developer has access to a computer terminal a central design repository was used to store and access these documents. Databases were introduced to store representations of the functional and UML diagrams. The problem with using many independent documents was that it was very difficult to ensure consistency and trap errors. Non functional requirements such as performance, failure management and functional criticality were particularly difficult to manage. Traceability was also difficult to document.

This assessment is based on the experience of a recently finished large distributed real time system and reflects some of the difficulties experienced. It focuses on the difficulties that the software developers experienced rather more than the academic challenges. It aims to identify future research challenges rather than any solutions. It is an avionics system that has been in development for nearly two decades and comprises hundreds of individual hardware equipments. A variety of programming languages are used it its processing units – the most common being ADA.

2 Current Design Approaches

Typically many individual companies are involved in the development of the sensors, processing units, displays and actuators that make up the system with all the components only coming together at the end of the development.

The requirements analysis is undertaken in two phases. In the first phase a textual document which outlines of the features that customer wants implemented is generated by the customer. This document forms the overall system baseline.

The requirements are then redefined by the system supplier in a set of documents which include a graphical representation of the requirements, a system architecture and supporting documentation to form the second phase. Traceability is redefined consisting of requirements defined in the customer documentation plus additional requirements included in the redefined requirements.

The requirements are defined using tools such as CoRE [6] or StateMate [7].

The design documentation for each subsystem is generated either by adding detail to the requirements definition or by generating a separate document. Numerous separate documents are stored in a requirements/design repository. Design management tools control the issue and applicability of these documents to the design phases of the development.

In the past independent functional processes were identified to model the requirements and design. In the implementation these independent process elements were scheduled on a rotational basis. Now UML based tools have replaced the functional modelling tools. Performance and functional criticality are defined in separate documents. Traceability is recorded using a separate tool such as DOORS [8].

Hardware is defined in Contact Specifications and Interface Control Documents. Implementation is defined separately aimed at satisfying the requirements defined in the design documentation. Software is tested on equipments rigs prior to delivery to the systems integration rigs.

Performance is calculated by manually extracting the parts of the design that contributes towards an end event using a complex mathematical model. The calculation takes into account processing errors and delays but uses assumptions about scheduling. Demonstration of full system performance prior to deployment is undertaken on a system integration rig. This is achieved by using simulators for the sensors and complex monitoring and test systems.

3 The problems associated with distributed real time systems

Models defined in UML can satisfactorily cover functionality but do not effectively cover other aspects such as scheduling, non functional requirements, traceability and system abstraction associated with cross system end events. There are a number of UML profiles which address real time systems [1][2] defined on the OMG web site. These profiles address non-functional properties modelling, time modelling, schedulability modelling and performance analysis modelling. They are complex and are not suitable for use at the requirements stage where a design does not exist.

Many of the current tools are complex in themselves. Tool complexity has two major drawbacks. Firstly without internal error checking, users of complex tools generate many errors. Secondly system developers also want to employ non specialist engineers which they can redeploy as necessary. They want to use young engineers to do most of the work cost effectively without the need for specialist training. The engineering workforce is mobile so as one engineer leaves another engineer needs to be able take over with the minimum of delay. The design detail needs to be accessible so that an engineer can work on a design element without the need to be an expert in the larger system.

Requirements/Design repositories contain disjoint definitions. This means that linkage between models and documents become prone to errors.

In some tools communication between the design repository and the developer does not match the developer's skills and the job he or she is undertaking. If the notation that is used mismatches either the developers skills or the job he is trying to do then the process becomes inefficient and susceptible to errors.

In practice the real world design decisions are never in the order that best suits the existing tool set. The requirements originated by the customer are rarely fully defined requiring the system provider to develop and document the requirements on the customer's behalf. Sometimes the developer *is* the domain expert rather than the customer. Some design decisions need to be made before the requirements and the design fully established. Very often these design decisions impact the final performance outcome.

Sometimes a functional view is required rather an object view. This particularly useful when scheduling is being assessed or the system is being viewed by domain experts.

It is important that the requirements tool can be used as the requirements emerge. The developer needs to focus his attention on a single requirement adding new requirements as he/she adds detail. For most systems the requirements definition will need to be reworked and extended as the detailed requirements emerge. The problem is compounded because very often some long lead hardware items will need to go to contract before the requirements are completely in place.

Very often the requirements and the design are stored separately so cannot be seamlessly used to host the emerging design and implementation. If the requirements and the design are stored separately then it becomes difficult to ensure that the design fulfils the requirements. In the past traceability between requirements and design has been made more difficult because the structure and detail of the design documentation does not match the requirements documentation. Usually the requirements and design teams are separate with their own management teams and objectives which compound the problem. Comparing the two sets of design documentation is time consuming and prone to errors. If a single tool can handle both then mismatch between the requirements and the design can be eliminated.

The functional elements defined in the requirements need be extended to embed non functional requirements and hence be animated to reflect the real performance of the system. Some collections of functional elements need be used to generate equipment contact specifications and Interface Control Documents to enable system equipment to be procured from another supplier as COTS or specially designed hardware.

Currently performance is calculated by manually extracting the parts of the design that contributes towards an end event and calculated using complex mathematics. The calculation takes into account processing errors and delays. Assumptions are made about how the system schedules functional elements which causes inaccuracies. Visualisation of the performance is difficult which leads to a lack of confidence in the results. Using a requirements/design repository which includes modelling of the schedulers would enable the calculations to be based on more accurate design information and would enable real time visualisations to be produced from real external scenarios.

It is not possible for most complex real time systems to be precisely defined from the start. Each stage of the development process adds new requirements. Requirements creep is common. Using requirements/design repositories to work on and investigate new requirements reliably would promote re-use.

Traceability is typically recorded separately with a new traceability definition added at each development stage. Checking that traceability documentation is correct is achieved by manual inspection which becomes tedious and error prone. Traceability between the design stages and development phases is difficult as the terminology used in each design stage and development phase does not fully match. This is compounded by the need to add extra requirements to match the added detail added in each design stage.

When the implemented software is first loaded into an element of the hardware it is common for the hardware not to work as expected. Comparison between the software implementation and the design is difficult. Reliable fully automatic code generation would establish a link between the design and the implementation. Automatic code generation would also mean that changes to the design can be mapped to the implementation automatically improving accuracy of the change control system. It would also be good if elements of the implementation could be interchangeable with its design counterpart in the design repository on a part by part basis so that the real time (or scaled real time) simulation that had been achieved at the design stage could be repeated with a mixture of the requirements/design representation and real code.

Demonstration of full system performance prior to deployment is undertaken on a system integration rig. This is achieved by using simulators for the sensors and complex monitoring and test systems. This occurs very late in the development program and is usually delayed by the non availability of hardware. It is further delayed if a problem is discovered or a failure occurs and the hardware or the software loaded in it has to go back for rework. Complex test and monitoring systems can be used to simulate individual hardware elements so that full system performance can be assess with some elements of the system missing.

If individual hardware elements could be duplicated by the test and monitoring system and linked to the requirements/design repository then changes to the system that require rework and redesign can be assessed prior to or even before that rework is initiated. There have been occasions where a cross system issues have required a hardware element to go through the rework process several times to achieve a satisfactory solution. If each rework cycle takes several months considerable delays can be experienced.

If the system integration rig and the design repository can be reliably jointly used to represent the behaviour of the system then the customer can be involved the operation of the system at an early stage.

4 MDA suitability for real time systems

Model Driven Architecture has been claimed as a method to overcome some of these problems described above. The main feature of MDA is the ability to define a system at a high level of abstraction (Platform Independent Model or PIM), followed by automatic transformation of the PIM to one or more Platform Specific Models (PSMs). One of the benefits is that the PIM can be constructed by those who understand the problem domain. It can be used to promote abstraction and re-use of model elements. It is claimed to be faster because once the transformation rules have been established the PSM can be generated automatically. Changes in high level requirements can be incorporated very quickly. If the transformation rules are correct then traceability of functional requirements is achieved.

However the use of MDA for hard real time systems is not without its problems. An efficient design process that achieves traceability for non functional requirements has yet to be established. The platform independent design notation was originally

UML (although the most recent standard permits any MOF compliant language). Extensions to UML are required if UML is going to be used. Another problem is that it is not clear how to integrate PIMs with hardware model representations of the system.

The MDA development lifecycle comprises of the following phases. The first phase takes textual requirements, conducts requirements analysis and generates a Platform Independent Model. The design is generated from the Platform Independent Model and is represented as a Platform Specific Model. The code is generated from the Platform Specific Model.

Whilst this is ideal for the development of business systems it needs to be extended for distributed real time systems. The architecture of a distributed real time system is so fundamental to the requirements that it must be taken into account at the start. Each element of the system requires a Platform Independent Model representing diverse elements such as Processing Units, Sensors, Displays and Actuators. SysML now enables the system to be defined within the PIM. [3]. In order to assess end to end functionality all internal and external communications need to be modelled with Platform Independent Models as well. The advantage of this approach is that functionality can be animated end to end prior to specific platform decisions are made.

As most real time systems are managed by schedulers of one kind or another the schedulers need to be modelled in a Platform Independent way so that the requirements can be run in a way that reflects real time performance. To do this schedulers can be used to represent busses and communication links.

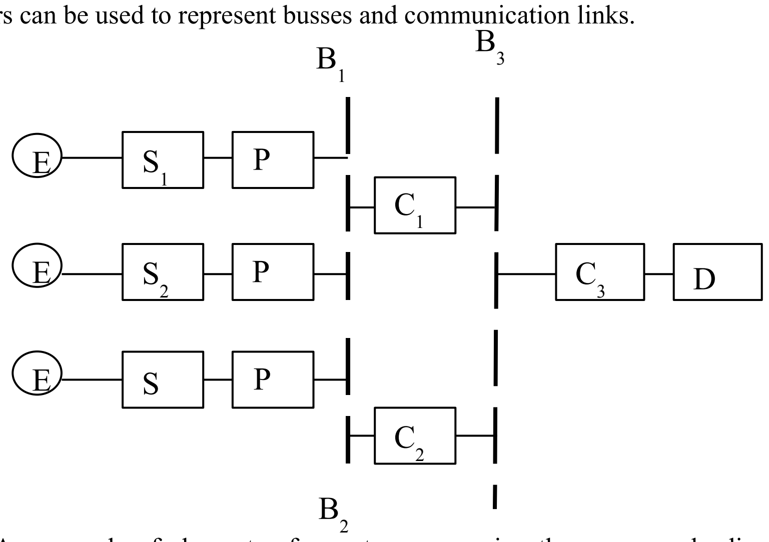

Fig. 1. An example of elements of a system processing three sensors leading to a single end event D

The Platform Independent Model needs to be capable of identifying individual model elements that contribute towards an end event so that these elements can be extracted from the model making the performance assessment simpler and easier to achieve. For example consider the system in fig 1 which has three sensors S1, S2 and S3 sensing the environment E in three different ways whose outputs are processed by processors P1, P2 and P3 respectively. Computer C1 combines the information from Sensors S1 and S2 received on Bus B1 and the information from sensor S3 received on B3. Computer C2 processes the information from Sensors S3 received on Bus B2 and transmits the results to Computer C1 on Bus B3. Computer C3 receives processed from sensors S1, S2 and S3 for display parameter D from Bus B3.

The behaviour of display D with respect to changes in the environment E depends on the accuracy of the sensors, processing errors and delays. If there exists a Model for each element (sensor, processor, bus, computer and display) comprising of independent functional elements and an associated notional scheduler then the cross system performance can be animated with changes to the environment E being represented on display D in a way that represents real cross system performance. Alternatively the relevant independent functional elements can be expressed in a way that independent calculation can be undertaken.

5 Research Challenges

The following research challenges exist.

- Can MDA Platform Independent models be extended so that real end to end performance of cross system end events can be assessed at the requirements stage? Currently this is not possible.
- Can the central repository of design be optimised to enable the user to be presented with views that suits his skill and the job he has to do? Current tools are challenging for non software engineers.
- Can the central repository of design be optimised to enable the requirements/design be used with the system integration rig? Although requirements have been animatable for some time requirements/design repositories have not been compatible with system integration rigs.
- Can UML be extended to support object and functional views of the Platform Independent and Platform Specific Models? It is currently not possible to simply extract independent functional elements associated with an end event and display them in a functional representation similar to a collection of CoRE 'threads'.
- Can the Platform Independent Model be extended so that Functional Elements associated with an end event be extracted and independently animated against specific test criteria and external scenarios? Platform independent models are animatable and therefore capable of being tested against internal and external test criteria. As far as we are aware, testing interconnected platform independent models is not currently being used.
- Can valid Platform Independent models COTS at a suitable level of abstraction and specially designed equipment that will be developed elsewhere that can be used in the functional and performance animation? Can this Platform Independent Model be transformed into a contract specification and Interface Control Document? Platform Independent Models can be generated to represent COTS but not in a simple form that can be used to integrate into multi platform independent model testing.
- Can valid Platform Independent Models express the requirements for internal and external communication elements such as busses and data links that can be used in the functional and performance animation? Platform Independent Models can be generated to model communication elements but not in a simple way that that can be used to integrate into multi platform independent model testing.
- Can the design be automatically generated from the requirements repository? Whilst this is difficult to achieve in full it may be possible to automatically update the design for requirements changes.

- Can the code for a particular processing unit be automatically generated from the requirements/design repository? Whilst this difficult to achieve in full it may be possible to automatically update the implementation for design changes.

6 Related work

There are a number of research groups undertaking research into real time systems. The most relevant work includes [4][5][6]. Whilst these papers deal with schedulability of real time systems they do not address introducing scheduling at the requirements stage enabling cross system functional and performance animation of distributed real time systems.

Lu et al [9] propose the concept of a Function Block as an output of a MDA based development process. It does not, however, describe how Function Blocks can be used in a large distributed real time systems. Cuccuru et al [10] describe a MDA methodology to bridge the gap between the abstract specification level and a heterogeneous architecture level but does not cover traceability and non functional requirements. DeAntoni et al [11] describe a MDA simulation based method for the development of real time systems but the simulation described does not cover a use case approach. In none of these three papers do the authors cover how an MDA based development method can be used in practice for large scale distributed real time systems.

Berkenkötter [15] describes how UML 2.0 overcomes some of the limitations of earlier UML versions to deal with real-time dependant weaknesses of earlier versions of UML. Selic [12] discusses large complex real time systems which have extended periods of evolving incremental requirements. Graf et al [14] discuss how the OMEGA-RT principle can be used to describe state changes and duration constraints to be expressed for the period between these state changes. Apvrille et al [13] describe how class and activity diagrams can be extended to cover real time systems. Whilst these papers and the books by Douglass [16][17], are useful in describing ways that UML can be used in real time systems they do not specifically address the application of MDA in this domain to obtain improvements in traceability and reduction of errors this approach is designed to provide.

7 Conclusions

The final cost of development and application of real time systems is rarely the same as the estimate at the start of development. The overspend is easy to calculate but the efficiency of the development process remains clouded. More and more real time systems are developed by many commercial organisations. Very often there are several customers involved all looking for a slightly different product. As the complexity of real time systems increase attention must be placed on how the development process is undertaken or there may be a limit to the complexity that it is practicable to achieve.

Various tools were developed on an ad hoc basis to identify errors and provide some linkage between the development stages. Whilst these some of these tools made startling improvements to error detection the overall process still remains somewhat disjointed. This approach still had a number of shortcomings resulting in late detection of mistakes that require rework at a late stage to overcome.

Current system development tools can satisfactory design real time systems but can still be improved by integrating the design documentation in the design repository. Even if some design stages cannot be automatically generated from the previous stage the computer hosting the design repository must be used to minimise the human part of the process to a minimum and present what elements that need human intervention in a simple and easy to use manner. In order to assess the benefits of and the improvements to MDA an evaluation technique must developed which accurately reflects the benefits to design and the developer.

Can MDA be utilised in association Platform independent models expressed in terms of its constituent independent process elements be used to overcome the difficulties identified above?

References

1. OMG profile 'UML Profile for MARTE' www.uml.org
2. OMG Profile 'UML Profile for Schedulability, Performance and Time' www.uml.org
3. OMG Profile 'OMG Systems Modeling Language (SysML)' www.uml.orgFengxiang Zhang, Alan Burns:, Schedulability Analysis for Real Time Systems with EDF Scheduling. (2008) Real-Time Systems Group, University of York
4. A. Burns, A.J. Wellings:, Delivering Real-Time Behaviour (2007) Domain Modeling and Duration Calculus Springer
5. Mullery 1979 Mullery,G (1979); "CoRE a method for controlled requirements expression"; Proceedings of the 4th International Conference on Software Engineering (ICSE-4), 126-135; IEEE Compuer Society Press.
6. StateMate available from Telelogic AB, PO Box 4128, Kungsgatan 6, SE-203 12 Malmö, Sweden. Also www.telelogic.com
7. DOORS available from Telelogic AB, PO Box 4128, Kungsgatan 6, SE-203 12 Malmö, Sweden. Also www.telelogic.com
8. Lu, Halang and Zhang:, A Component-based UML Profile to model Embedded Real-Time Systems Designed by the MDA approach. Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)
10. Cuccuru, Simone, Saunier, Seigel and Sorelo:, An innovative MDA Methodology for Embedded Real-Time System. Proceedings of the 2005 Euromicro Conference on Digital System Design (DSD'05)
11. Julien DeAntoni and Jean-Philippe Babau:, A MDA-based Approach for real time embedded systems simulation. Proceedings of the 2005 Ninth IEEE International Symposium on Distributed Simulation and Real –Time Applications (DS-RT '05)
12. Bran Selic: Using UML for Modelling Complex Real-Time Systems. Languages, Compilers, and Tools for Embedded Systems ACM SIGPLAN Workshop LCTES '98 Proceedings.
13. Apvrille, Courtiet, Lohr and Saqui-Sannes: TURTLE: A real time UML Profile Supported by a Formal Validation Toolkit. IEEE Transactions of Software Engineering Vol 30 No7 July 2004.
14. Graf, Ober and Ober: A Real Time profile for UML. International Journal on Software Tools for Technology Transfer April 2006 Vol 8 Iss 2 pages 156-166.
15. Kirsten Berkenkötter: Using UML 2.0 in Real-Time Development A Critical Review. EAI Knowledge Base 2004
16. Bruce Powell Douglass: Real Time UML ISBN: 0-321-16076-2
17. Bruce Powell Douglass: Doing Hard Time ISBN 0201498375