

# Applying MDA in the avionics: A practical approach

Teodora Bozheva, Terry Bailey, Julia Reznik, Tom Ritter

{Teodora.Bozheva, Terry.Bailey}@esi.es  
{reznik, ritter}@fokus.fraunhofer.de

## Introduction

Increasing the efficiency and the quality of the results of the software developers is an objective for all organizations and projects. There are different approaches to achieve this goal. Depending on the context of an organization or a project, one or another approach is more suitable.

Model-Driven Architecture™ (MDA) is usually portrayed as a means to perform the same amount of work with less people, ensuring in addition, improved quality of the software developed, decreased time to make changes to existing systems, and organizational know-how kept within the core of the products, namely through their models.

Some of these aspects are extremely important for specific types of systems. An example is an air-traffic management system, which must be free of defects, straightforward to modify and comprehensively documented. A typical and logical question posed by decision-making people, however, is how MDA could be successfully introduced in such projects.

We find that the three important factors for successful application of MDA are (1) basing the MDA-application on an existing product, (2) definition of an appropriate life cycle, and (3) develop/use a tool chain, by means of which to implement the software system. Applying MDA for a first time is much easier when there is available good domain knowledge and a clear idea about what should be modeled and what outputs are expected from the model transformations. In addition, it could be estimated the extent to which a tool chain improves the entire development process, although at the time of writing this paper such data are not yet available to the authors. With respect to the life cycle, an important prerequisite is to ensure effective and efficient learning from experience, and on-time delivery of quality results despite of the potential difficulties with adopting the new technology.

This paper describes our experience when applying MDA to the development of an air-traffic management system. We discuss the complementarities of these factors using this particular case as an example.

## The project and its predecessor

Aircraft coordination in increasingly crowded airspace is becoming a major concern for air traffic management authorities around the whole world. Conventional management schemes are being replaced by extensively computer-integrated Air Traffic Management (ATM) Systems to maintain safety levels and increase throughput of congested airways. One of the primary goals of introducing ATM systems is to provide the controllers as much information as they need to effectively manage the air traffic, and in a comprehensive form while taking care not to overload controllers with unnecessary information.

Research and development activities in this area have been established with the D<sup>3</sup> system, developed by NEXT S.p.A. for an Italian national project. D<sup>3</sup> System provides a 3D System for geo-referenced data representation and visualisation.

The AD4 project [16] extends D<sup>3</sup> and develops an innovative Virtual Air-Space representation for ATM Systems to provide the controllers with the ability to use 3D data about the air traffic/airport space in real time.

To develop and validate the requirements for the AD4 system, observations and in-depth analysis of the work practices and strategies used by the air traffic controllers have been carried out and a number of operational scenarios have been defined. The resulting system requirements together with the technical solution of the D<sup>3</sup> system are the basis for the architectural design of the AD4 System.

From the point of view of the MDA technology, the principal benefit of having the D<sup>3</sup> system available is that it significantly facilitates the understanding of what the resulting product has to accomplish and how it is expected to work. Having a correct idea about the operational aspects of a software product is a crucial point in the MDA-development. This allows making a formal specification, i.e. by means of models, leaving aside (for the moment) the technological and engineering details which are irrelevant to the fundamental functionality of the software system. At the same time, knowing the background system guarantees to a great extent the correctness of the architectural solution for AD4. In particular, it ensures that the AD4 components and their interfaces are adequately defined.

As a successor of D<sup>3</sup>, a number of specific requirements exist with respect to AD4. Namely, it has to be a distributed component based system, reusing existing D3 components and providing integration with external, pre-selected platforms.

Additionally, the AD4 development has to be model-driven. In order to minimize the risk of applying the new MDA technology and to ensure rapid development of the AD4 system, agile approaches to software development also need to be put in place.

## AD4 life cycle

A System Development Life Cycle is the overall process of developing a software system through a multistep process from investigation of initial requirements through analysis, design, implementation and maintenance. Defining an appropriate life cycle for a project helps to achieve predictable results and to coordinate resources. This is an essential success factor for a project such as AD4, developing big and complex systems, involving specialists from different organizations and countries.

Taking into consideration that the AD4 system (i) is based on the D<sup>3</sup> software, which includes a number of components, to be reused, (ii) is developed by a distributed team of experts in the ATM domain and in software development, and (iii) involves exploration of the new MDA technology, a number of constraints to the life cycle are identified. More precisely, it has to:

- Support component-based development
- Support model-based development
- Be iterative
- Support collaboration of distributed teams
- Support learning from experience.

A number of established life cycle models have been investigated with respect to their suitability to the AD4 project. Among these are the Component-based software development life cycle model [9], Spiral [8], Rapid Application development. Also, life cycles derived from the best known agile methods, Agile Modelling [10], eXtreme Programming [11], Feature Driven Development [13] and Adaptive Software Development [14] have been analyzed.

A key feature of the component-based software development life cycle model is the emphasis on reusability during software creation, and the production of components that are meant to be used in future projects.

Reusability implies the use of composition techniques during software development, which is achieved by initially selecting reusable components and assembling them or by adapting the software to a point where it is possible to pick out components from a library.

In general, this life cycle model supports the requirement to build the AD4 system based on components and to reuse components from D<sup>3</sup>. However, since D<sup>3</sup> was developed within another R&D project, the system components have not been fully packaged as to be easily reused in future similar applications. This implies performing additional activities related to the completion and the preparation of these components for usage in AD4. Moreover, this has to be aligned with the scope and the effort

planned for the AD4 project, i.e. it requires decision making for each D<sup>3</sup> component to be included in AD4.

Another problem is that the AD4 system is foreseen to be integrated in different platforms. Additionally, specific security aspects have to be considered and implemented in it. Since both, the platforms and the definition of the security aspects to be addressed in AD4 are a subject to investigate, the life cycle should provide the necessary flexibility with respect to changing requirements. This is not that easy in the component-based life cycle due to the extent of rework related to completing the components.

The agile methodologies provide features like iterativity, team collaboration and learning from experience. These methods are also suitable for developing products which requirements are rather in a process of investigation. However, the agile methods are not quite appropriate when the product development is carried out by teams distributed in several countries, with different types of expertise and different levels of experience with the technologies to be used. Therefore, only some aspects of theirs are selected to be included in the AD4 life cycle.

With respect to the MDA technology, it is important that the life cycle reflects the development of a tool chain, which will be used to implement the software system. The process of tool chain development includes the following steps (1) identifying the Platform Independent Model (PIM) and the Platform Specific Model (PSM) metamodels (concept spaces), (2) creating model repositories and (3) creating model transformation (PIM -> PSM) and code generation (PSM -> Platform).

Based on the considerations about the life cycles related to the AD4 one, two alternative life cycles have been defined and analysed according to specified criteria. The one which has been selected for the AD4 project is illustrated and explained in the next section.

## **Phases of the AD4 development life cycle**

The overall AD4 life cycle is organized in phases (iterations) with each phase being characterized by a series of goals and activities to be performed in order to achieve these goals. Within each iteration a small subset of requirements is selected to be developed. At the beginning of an iteration the system requirements are revised: existing requirements can be updated (on the basis of the previous iteration review) and new ones added.

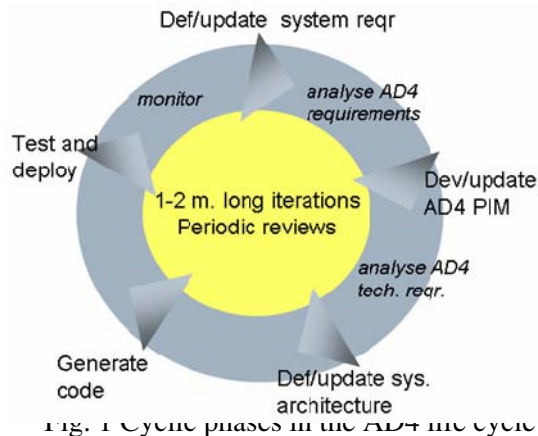


Fig. 1 Cyclic phases in the AD4 lifecycle

Changes are clearly propagated to subsequent activities such as platform independent modelling, architectural design, implementation, test and deployment. Each iteration terminates with a review and a retrospective of all the activities performed within it with the objective to assess development results achieved in the period.

As a result each development phase allows us to incorporate lessons learned into the next iteration. In the following sections we outline the content for each phase and describe what we hope to achieve:

### **Phase 0: Preparation**

This phase aims to prepare the “environment” for later phases and lay the foundations for all of the iterations throughout the project’s duration. Key activities of this phase are centred on requirements gathering and planning of infrastructural concerns such as the modelling and development infrastructure (AD4 Tool Chain) and the actual physical and logical platform (preliminary architectural design) on which the system will be built. Later a subset of requirements for next phase are selected and a preliminary PIM model is produced.

### **Phase 1: First release**

The aim is to provide the first release of system infrastructure and core components. The development focuses on the implementation of requirements selected in the previous phase. Key activities concern the identification and enhancement of D<sup>3</sup> components to be reused in AD4, the identification of simulation platforms components to interoperate with and the design of integration strategies. System integration and test activities are

part of this phase as well. The first PIM to PSM transformation is produced and then, after refining the PSM model, the code for the identified components is generated. To check the validity of these new components we need to prepare the system integration and test environment. Another key part of this phase is the updating of existing assets and the identification of new components that will need to be developed. In order to include improvements in future phases, a “retrospective” workshop is held, where we decide on any corrective action that needs to be included in the next phases and identify and problematic areas.

### ***Phase 2: Final release***

The aim is to provide the final release of system infrastructure and core components. A key is the inclusion of the lessons learned in the previous phase. The activities to be carried out are those already defined in Phase 0 modified according to the conclusions of the retrospective workshop. At the end of this phase we will hold another workshop to provide another feedback loop making sure we continually adapt our process by applying best practices thus mitigating risk throughout the projects development.

### ***Phase 3: Demonstrator***

This phase aims to define a scenario, to construct the demonstrator and to integrate it with ATC simulation platforms in order to validate the proposed airspace/airdrome 3D representation. In this phase we will include the components developed up to now and define the test cases for the demonstrator. The lessons learned from this phase will be used for future development work and the whole life cycle will be stabilized.

## **AD4 tool chain**

The model-driven development process in the AD4 project consists of two parts: first, designing and building of the AD4 tool chain; second, designing and building the AD4 system by using the AD4 tool chain following the AD4 development lifecycle. The idea is to integrate the most suitable existing development tools for AD4 in one open integrated environment, and implement just model transformers and profiles to make it work. The artefacts that are needed for the tool chain construction are shown in Fig. 2 and explained in the following sections.

## Platforms

In order to automate mappings between application models we have to identify and define an executable technology or platform for AD4 system. Indeed, the input to and output from AD4 tool chain steps is directly dependent on the way platforms are used and described.

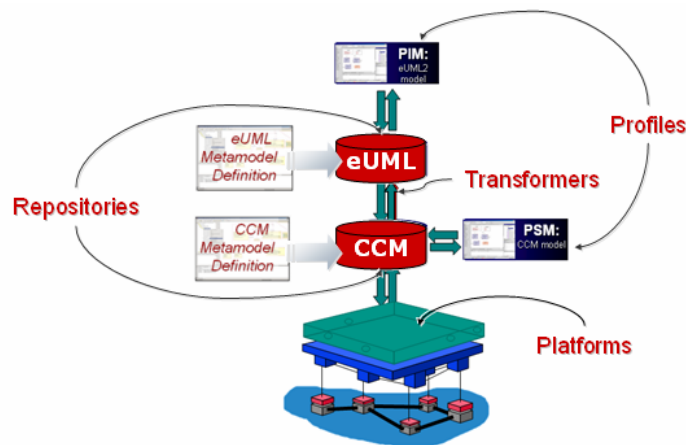


Fig. 2: AD4 Tool Chain building artifacts

The CORBA Component Model (CCM) [1] defines a component model based on CORBA which supports interactions of complex distributed objects written in different languages for different operating systems. All components of the AD4 system are developed as CORBA Components to gain from benefits a component based platform provides with respect to the composition and the deployment of applications. In the AD4 project we use Qedo [2] which is an open source implementation of CCM. Qedo provides code generators, development support, runtime environment, and deployment infrastructure for CORBA Components.

## Metamodels and repositories

Metamodels play an important role in the AD4 tool chain building process because the metamodels provide means to manage models. Out of metamodels we can create repositories where models are stored and managed. For the AD4 project we need both a PIM and a PSM metamodel. For PIM modelling we use UML2 [5,6]. However, in order to avoid huge repository size, premature design and to facilitate the comprehensibility of the modelling techniques to involved user with varying UML2 background

a specialized subset of UML2 language, namely eUML (essential UML) has been tailored. eUML includes only required UML2 metamodel elements which formally define modelling elements, their semantic and relations. The PSM metamodel is the standardised CCM metamodel as defined by the OMG [1].

## **Tools and Profiles**

Selecting the right tools is essential for building an effective tool chain. Since we use UML2 for metamodel specification and for system design we selected Enterprise Architect (EA) from Sparx Systems [7] as a host UML modelling tool and realised the eUMLModeller, which is a Plug-In implementation of the eUML Profile for EA and used for PIM modelling. eUMLModeller is synchronized with the AD4 repository in both directions (load and store of eUML models), including the synchronization of graphical information of the models stored in the repository.

As a modelling front-end for CCM an Eclipse Plug-In, the CCM Modeller, has been developed based on EMF and GEF. The Eclipse Plug-In is a profile implementation for CCM. The Plug-In is also synchronized with the AD4 repository in both directions.

## **Transformers**

To achieve the integration of different modelling techniques and for different modelling layers, the repositories (eUML and CCM) are interconnected together by specific model transformers, which map models to other models or to a programming language code.

Two transformers have been built and integrated into the tool chain: eUML2CCM and CCM2IFR. eUML2CCM transforms eUML models into CCM models. Since Qedo's code generators are based on the Interface Repository (IFR), the CCM2IFR transformer has been developed, which integrates Qedo into the AD4 tool chain. CCM2IFR implicitly generates C++ code for CCM components by triggering the Qedo code generators after transforming the CCM model into an IFR Model.

## **AD4 Tool Chain Architecture**

The AD4 Tool Chain architecture is represented on Fig. 3:

The heart of the AD4 tool chain is a generic control application component, implemented for AD4 and used to manage and control the various

components of the tool chain. The AD4 Control Application completely manages the loading of repositories, transformers and models.

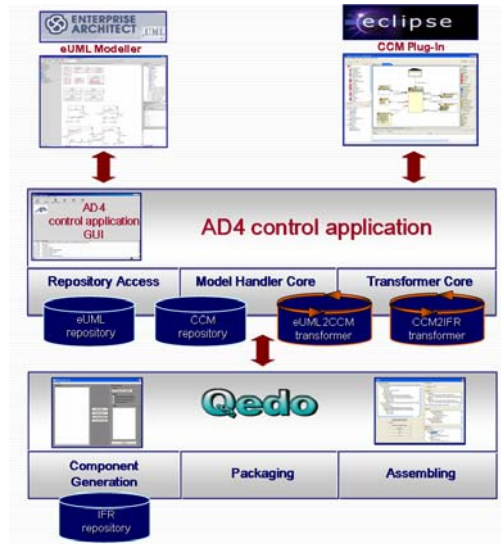


Fig. 3: AD4 Tool Chain architecture

In the standard configuration of the Control Application loads the eUML and the CCM repositories, and two transformers: the eUML2CCM and CCM2IFR.

Fig. 4 shows how to apply the AD4 tool chain for system development:

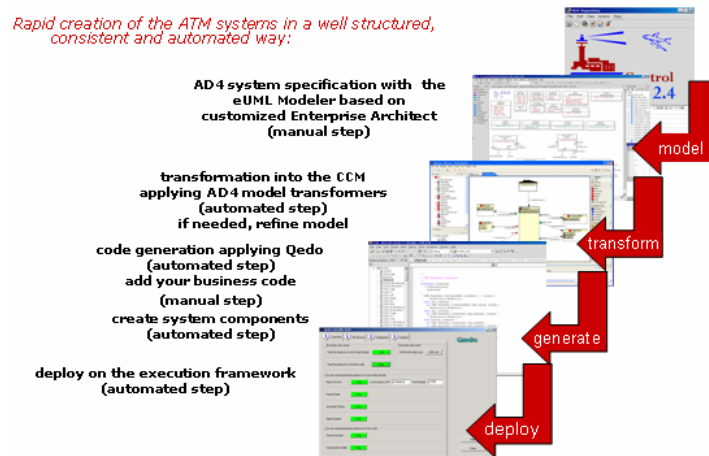


Fig. 4: System development steps using the AD4 tool chain.

The development of the new AD4 platform starts with requirements specification and analysis. Afterwards, it has to be decided how the existing D<sup>3</sup> platform will be modified, in particular what components will be created or updated and what technologies and techniques will be used for this. In the next step new components are designed using the eUMLModeller. The created eUML model is then ready to be transformed into the CCM model applying the eUML2CCM transformer. The transformed CCM models can be either refined, by using the CCM Modeller, or directly transformed into the Qedo IFR repository which implicitly invokes the Qedo code generators and produces the source code of the CCM components.

## **Tying it all together**

MDA is a new technology allowing the organizations to benefit from the model-centric approach of software development. However, applying MDA requires performing specific practices related to preparing and handling the needed modeling environment, as well as to developing software based on models. These requirements imply specific adjustments to the life cycles which would be used in code-centric approaches to development.

In the AD4 project, the preparation of the modeling environment started together with the investigation of the system requirements and the development of the operational scenarios, in Phase 0. Normally, the requirements should be traceable to the PIM and PSM constructs that implement them, and vice versa. However, requirements specification in the case of the AD4 project involves considerations of human, psychological and 4D Human Machine Interaction factors, which substantially increases the complexity of this task. Moreover, defining the requirements in a format, which allows integrating them in the tool chain, proved to be an issue that still requires deep investigation. Therefore, the AD4 tool chain begins with a PIM, developed by means of the eUMLModeller, and the system requirements are modeled independently from the AD4 tool chain.

Designing and building the AD4 system by using the AD4 tool chain is iterated through Phases 1 and 2. The approach is to prioritise the requirements for the AD4 system and to start with modeling and creating new or updating D<sup>3</sup> components addressed by the highest priority requirements and proceed like this with requirements having lower priority. The PIM and PSM models are continually maintained synchronized.

A key activity at the end of each iteration is the retrospective workshop, which gathers together all involved in the iteration to discuss which practices were particularly useful for the joint work and how to improve other

practices as to achieve better development results. This activity fosters the adoption of the new MDA technology and the learning from experience.

The proposed tool chain provides the developers the possibility to decide later on which platform to run the product. This opportunity is realized by relevant transformers that should be integrated into the tool chain once the target platform is selected. Additionally, since other specific technologies or platforms (e.g. J2EE) can be supported and the eUML models can be transformed to new platform specific models, the list of loadable tool chain components can be arbitrarily extended.

## Conclusion

Our experience in the AD4 project shows three main conclusions:

A practical approach to adopting MDA, especially in complex software development project is to apply the new technology to extend an existing application in order to have a clear idea about the expected inputs and outputs from the modeling activities as well as to benefit from the knowledge to develop software products in the domain.

Defining a life cycle, which is based on short iterations, active feedback back and forth between the designed system and the actual output, and retrospective workshops at the end of each iteration, facilitated the adoption of the new technology and learning from experience.

Developing an adequate tool chain not only speeds up the development process and guarantees the quality of the results, but also increases the flexibility with respect to future developments. For the time being these conclusions are only based on qualitative data since quantitative ones from the previous and the current project are not available yet.

Using MDA provides a number of benefits to projects like AD4:

- Easier implementations on different platforms while conforming to the system structure and behavior as described in the PIM.
- Integration of different applications by explicitly relating their models, enabling integration, interoperability and system evolution aligned to platform technologies' one.
- Easier validation of models uncluttered by platform-specific semantics, since the PIM does not include unnecessary information.
- Clear separation of concerns throughout development process
- Improved system quality due to better and earlier fault detection

An open issue remains the integration of the requirements specification in the tool chain and this is the focus of our future work.

## References

1. Object Management Group. CORBA Component Model. OMG document number formal/02-06-65
2. Qedo Team. Qedo (Quality of Service Enabled Distributed Objects) CCM Implementation Web Page, <http://www.qedo.org/>, March 2006
3. Object Management Group. Meta Object Facility Core Specification 2.0, OMG document number, formal/2006-01-01
4. AD4 Consortium. EU FP6 R&D project AD4 - 4D Virtual Airspace Management System, <http://www.ad4-project.com/>
5. Object Management Group: OMG ptc/04-10-02: UML 2.0 Superstructure Revised Final Adopted Specification
6. Object Management Group: OMG ptc/03-09-15:UML 2.0 Infrastructure Final Adopted Specification
7. Sparx Systems: <http://sparxsystems.com.au/>
8. Barry Boehm, Spiral Development: Experience, Principles, and Refinements, CMU/SEI-2000-SR-008 (2000)
9. Luiz Fernando Capretz, Y: A New Component-Based Software Life Cycle Model, Journal of Computer Science 1 (1): 76-82, 2005
- 10.Scott W. Ambler, Agile Modelling, John Wiley&Sons, Inc., 2002
- 11.Kent Beck , Extreme Programming Explained: Embrace Change, Addison-Wesley (2004)
- 12.Schwaber K., M. Beedle, Agile Software Development with Scrum, Prentice Hall (2001)
- 13.Palmer St., J. Felsing, A Practical Guide to Feature Driven Development, Prentice Hall (2002)
- 14.Highsmith J., Adaptive Software Development, Dorset House Publishing, (2000)
- 15.McConnell St., Rapid Application Development, Microsoft Press (1996)
- 16.[www.ad4-project.com](http://www.ad4-project.com)