



Philippe Mils: *Thales Resear & Technology*
Usine Logicielle Project Coordinator
Contact : *philippe.mils@thalesgroup.com*

Usine Logicielle

Position paper

Abstract

Usine Logicielle is a project operated in the context of the French System@tic Cluster. Its objective is to create an integration platform of tools allowing the engineering of complex software dominant systems. The efforts of 19 partners part of private and public research centres, as well as small to multinational companies focus on the integration in that platform of up to date tools and on the adoption by large communities of users of these new technologies. The Eclipse architecture provides the foundation technologies that enable the building of that integration platform.

In that paper we present the project's scope in term of technologies and studied engineering common problems. A focus on how Eclipse plays a central role as a technology provider is made.

Introduction

Complex systems developed and deployed in high technology companies are characterized by a strong heterogeneous environment (assembling software and hardware components), large size (large number of complex calculations, organisation in term of systems of systems), a dynamic environment (sensitivity to time, functional evolution regarding the context), a capacity to deal with the close interactions with their human and physical environment and lastly the ability to take in charge critical missions in terms of security for the people and goods.

Finding new solutions in order to increase the productivity of software development, increasing the quality of the delivered systems thus reinforcing the qualities of Ile de France facing international competition and lastly generating a strong increase on the market, these are Usine Logicielle challenges.

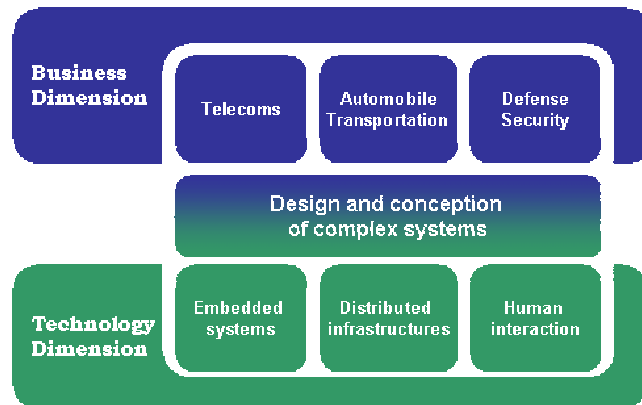
The implementation of the model driven engineering and the provision of flexible and optimal infrastructures of execution are the main means allowing to provide a beginning of answer to these challenges. In addition, the separation of the business concerns from implementation constraints allows building systems which are simultaneously globally stable in time and evolutionary while leaving the components of infrastructures living their own rapid rhythm of evolution. The results have shown that there is no unique existing tooling solution to deal with the design of all the types of complex systems.

- Proprietary tool chains are already existing and address particularly well specific engineering issues. However they do not provide any unified solution for integrating these tool chains into a complete whole engineering process wide solution.
- Numerous specialisations of processes and tools must be made in order to answer to the needs of the different technological domains and markets.

At the heart of the pole, the Usine Logicielle project is positioned as a supplier of engineering technologies



that can be used in a generic manner and on a small number of technical application fields: for the moment Real time, Embedded and Technical Information Systems. This project plays a central role with regards to the other projects of the pole System@tic in the sense that these technologies can be composed and specialised to deal with vertical application domains needs required by other projects of the pole: The Automobile Market, Super computers, High scale simulations etc.



The Market position of Usine Logicielle

Project's Approach

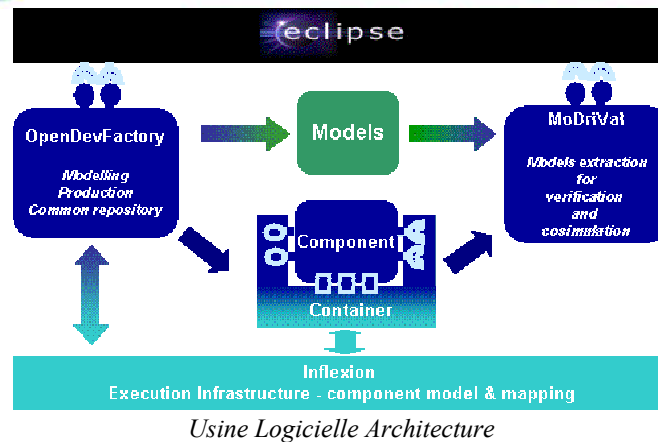
In order to reach the project's objectives the project is build among the following principals:

- The scientific and technical research community elaborates new solutions.
- These solutions are submitted for experimentation to large companies who are the potential users of engineering solutions and are confronted on a daily basis with the problem of mastering the complexity. They are the ones who vouch for the pertinence of the problems to resolve and for their proposed solutions.
- The utilisation of the solutions is generalized in the French industrial web. Only the small and medium size software publishers or consulting companies play this role of dissemination vector.

Work Program

Usine Logicielle places *modelling* at the central place of the development of complex software systems where the model becomes the reference for the application from which codes, documentations and tests can be generated. Usine Logicielle focuses essentially on automation, techniques of generation, full scale deployment of techniques of validation and verification but also on the capitalisation of the know how and the sharing of reusable software assets between domains. The Work Program splits into three distinct and complementary axis:

- The study and development of a support for model driven engineering and more particularly for the modelling and production of embedded systems: **OpenDevFactory** sub project .
- The study and development of a support for the test and validation of models and of software components issued from the model driven engineering: **MoDrival** sub project.
- The definition of a flexible and optimised component based execution platform dedicated to real time embedded systems and capable to support the execution of components issued from the MDE: sub project **Inflexion**.



OpenDevFactory

The objective of the sub project OpenDevFactory is to supply a standard platform for integrating technological developments for modelling software tools. This sub project produces technological components on top of which domain tools (automobile, security, telecommunication, aeronautical...) can be derived at a lesser effort. That platform will be Eclipse centric supporting interoperability with third party tools.

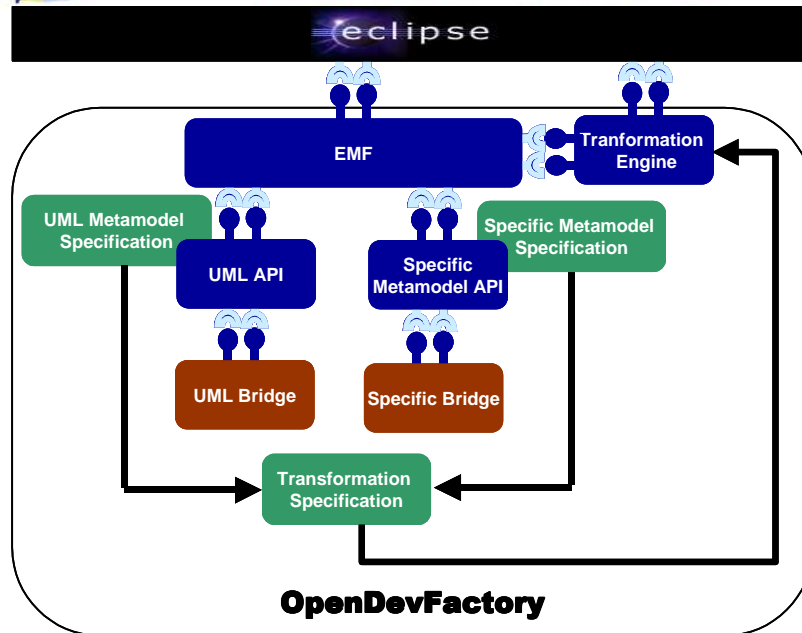
The technological bricks are organized as follows:

- Technological infrastructure bricks for MDE such as providing support for model transformation, behaviour modelling as well as orchestration of engineering activities. These components are provided as Eclipse plug-ins.
- Domain extension bricks supporting fault tolerance modelling, Real time embedded systems modelling, platforms modelling, requirements modelling or UML simulations. These components are as well provided as Eclipse plug-ins.
- Integration technologies of MDE design environments with other engineering environments such as design environments for design of automatism or critical embedded software. These technologies are build above the core Eclipse components such as EMF and ATL. This is detailed in the next section.

Eclipse architectural integration pattern

The integration structure of **OpenDevFactory** is build on top of the Eclipse framework which is de facto the integration hub of the Usine Logicielle platform. On this structure, the generic and specific engineering components are case by case integrated depending on the needs of the industrial project's case studies.

The Eclipse integration architectural pattern that is heavily used to perform the integrations is represented here after:



Eclipse architectural integration pattern

- EMF provides the meta-model management facilities:
 - The service to specify new metamodels.
 - The service to generate the corresponding model management APIs.
- Case study specific bridges are implemented when necessary to import and export conformant models into the Eclipse workbench from various sources and syntaxes.
- The transformation engine is used to transform one source model conformant to a source metamodel to a destination model conformant another corresponding metamodel. The transformation specification is written by the integrator according to the integration needs.

Engineering Case studies

There is a significant difference between providing an environment to support case studies and providing it in an industrial operational context. Numerous questions, issues and opportunities are raised by realising true model driven engineering. They need a deep analysis which can be realised with the help of industrial experimentations. Thanks to **OpenDevFactory**, the emerging themes that we can investigate are numerous. Among these, the project will focus on the following:

Engineering heterogeneity:

The objective is to integrate in the same engineering process various types of competing engineering disciplines. The technical domain chosen for this experimentation is real time embedded. In order to design such systems and to satisfy the constraints of volume or of resources consummation (time, memory and energy...), it is necessary to make use of several specific engineering disciplines in the context of MDE such as performance engineering, test engineering and embarcability engineering.

Discontinuity of engineering processes :

Some system classes need to integrate in a same design process different interdependent engineering disciplines: mechanic, physic, electronic, computers etc. These kinds of complex systems are not only software predominant because each solution element brought to the system architecture by exercising an engineering



discipline impacts the nature of the problem to deal with by the other engineering disciplines.

The issue which is discussed in the project is to make the iterative process resulting of this situation seamless by bringing a formal support to the continuity of the engineering processes.

Models heterogeneity:

The support for obsolescence management does not only concern the electronic components and computers, it also includes engineering tools used by companies in their system and software IDEs. For software embedded in long life time products such as nuclear centrals or aircrafts, the durability of these tools producing these software can be less important than the products inside of which they are integrated. We are then confronted to a problem in MDE which is to transform in the most efficient way possible the model of the embedded software from the obsolescent tool into a model for the replacement tool without modifying the behaviour of the software defined by the model to be translated. Just like the issue of interoperability between engineering disciplines is sorted out, the model transformation tools of the **OpenDevFactory** platform eases the development of translators thus allowing the migration of the software models.

City planning of technical information systems:

The technical information system “carries” the virtual product throughout the life-cycle, crossing its different temporal states (before projects are completed up to ready for retirement products) and its different views for each discipline and this, in a distributed heterogeneous and multi partners environment. The objective here is to put into place solutions allowing 1) to assure the interoperability of the different technical information systems partners and 2) to allow the transparent evolution of the technological implementation tools (E.g. The replacement of one application with another.) This objective requires the capacity to formalize the management and evolution model of the technical information and to master the conversion procedures (link with heterogeneous engineering and platform heterogeneity.)

Platforms heterogeneity:

The objective is to integrate in the same engineering process design aspects and multi platforms distributed deployment aspects. The problem is to be able to state at a lesser cost the design of a real time system on different execution platforms not necessarily offering the same quality of service nor the same programming model. In order to design systems according to this approach, it is necessary to use different engineering disciplines stated in the MDE world such as execution platform modelling, real time aspects modelling, model transformation and completion by non functional elements, failure management; this in order to satisfy the targets performances constraints.

Multi formalisms semantic correspondence:

The design of components based system on type “models of numeric algorithms” is characterised by an heterogeneity of the formalisms used for describing their behaviour. In order to study and finally validate the global consistency of a GALS system (Generally Asynchronous Locally Synchronous) built with the help of these components, it is necessary to lean on a common point of description of their properties, in particular for their real-time properties. The models built on this meta model (or semantic pivot) must be consistent and meet by construction the desired properties for the considered system. Thus for the same functional architecture, different ordering for the model execution can be defined but only a few are consistent from the real time and behavioural point of view.

The objective will be to evaluate a prototype of semantic pivot meta model, consistent with the formalisms used for this class of systems and allowing to separate the functional design from the model of execution.

MoDriVal

The **MoDrival** sub-project is centred on the verification of complex systems at different stages of their development from their specification and their modelling up to their acceptance. The notion of complex sys-



tem is restricted here to those which complexity results in the simultaneous presence of software and hardware components whether these materials be existing processors, systems on Chip (SoCs) or Network on Chip (NoCs).

Several themes are approached in the sub project:

- The testability through diverse techniques of automated test generation especially by statistic methods or starting from formalised requirements.
- The automatic verification of behavioural properties of software models and of programs.
- The simulation of hardware architectures supporting the execution of software to estimate performance or energy consumption.

The first family of investigated techniques specifically concerns tests synthesis which means to calculate the input test data that allow verifying particular software or system properties. Different techniques are studied such as generating tests cases from formal requirements or from a distribution of a targeted test coverage for the software's structure to be tested.

The second family of techniques studied in **MoDrival** is the static analysis by abstract interpretation. Instead of executing the testing program for a set of numeric values, a mathematic model replaces the execution of the program using symbolic values, meaning any kind of value for the input data, taking into consideration in the algorithm only what is pertinent for the class of properties to be established. Two types of properties are considered: a majoring approximation of rounded errors for numeric calculations and the verification of inexistent memory overflow during program execution.

Finally the third family of validation techniques studied in **MoDrival** concerns the performance evaluation of a software running on a hardware simulated by software in order to predict precisely the execution time and the spent energy. These predictions are essential when embedding software in small objects such as cellular phones or chip cards during the development phases where there are only software models of processors available.

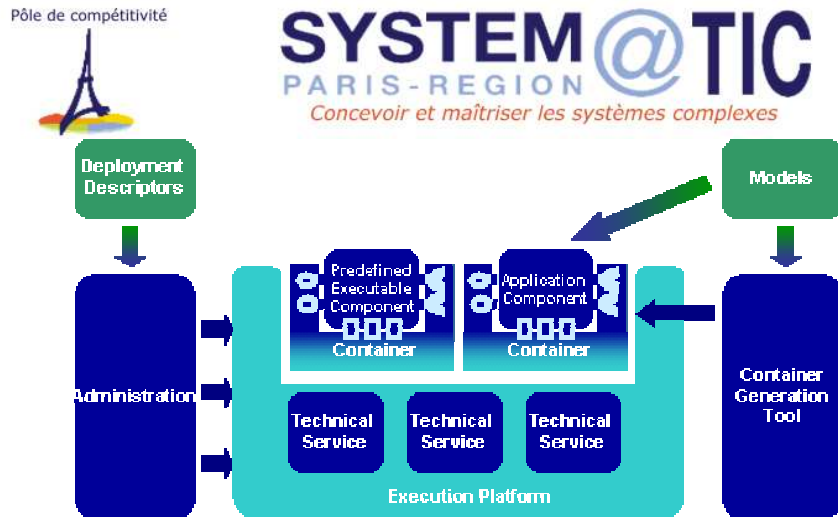
Inflexion

The **Inflexion**'s sub-project objective is to define a flexible and optimised execution platform dedicated to embedded real-time systems. This platform will be compliant with the component/container architecture principles which allows a true separation of functional concerns (or business) and non functional concerns (or technical services). The **inflexion** platform will facilitate the design of systems by assembly of components and will facilitate the reuse of existing ones.

Inflexion extends the tasks already started on the theme of components based infrastructures for RTE systems by bringing the following innovations.

- Support for Fault tolerance management.
- System dynamic reconfiguration capability.

Furthermore, the design and assembly of a system running on top of Inflexion will be realised using the MDE IDE supplied by **OpenDevFactory**.



Inflexion's Architecture

The container :

The container is the key architectural element of the component/container organisation. It isolates the components it protects from the underlying platform and provides the non functional properties which are required (temporal requirements, fault tolerance). These non functional properties are specified by the means of descriptors which allow attaching quality of Service requirements to the functional connectors of the components. As indicated on the right hand side of the above diagram, a tool for automating the generation of containers (parameterized by the descriptors of the components that they will host) is an inherent part of the environment provided services.

The administration service:

This environment service figuring on the left hand side of the hereunder diagram allows the display and configuration of the components and their containers then the supervision of the system and its reconfiguration in case of drifting. As a consequence, this component is highly impacted by the project's innovations (fault tolerance and dynamic reconfiguration.)

In fact it must then support the starting, the stopping and the modification of one part of the system (and not of its entirety) while keeping a whole consistent execution. It must cope also with the partial failure of re-configuration operations and allow their roll back. Finally, it must keep the system in a "fault tolerant" state while supervising the components state (active and passive) in collaboration with the containers that supervise them continuously.

Fault tolerance:

Fault tolerance is an extremely vast field. The first step is to define the taxonomy of faults and their management strategies that the project will address. On this basis, the model of fault tolerance for an application must be established. This model must in particular specify the criticality which applies to certain parts of the application (in most cases, dealing with the whole system at the highest degree of criticality is not realistic.) and the fault tolerance strategies (types of faults to detect, hardware faults, deadlines failure, time drifting, numerical exceptions, management strategies, redundancy, reconfiguration...) that we want to put in place.

This model is then taken into account (for use) by the administration service in collaboration with the dedicated support integrated in the containers:

- Detection of the components' faults according to the specifications of the model.
- Application (in the most transparent way possible) of the selected correction strategies (for example, in case of active redundancy, coordinated emission of requests towards diverse replicas then selection of the right answer-the application code seeing only one query call to a unique emitter.)

It is important to notice that the existing container architecture already allows the possibility of its own ex-



tion and defines the concept of integrated behavioural service.

Reconfiguration :

Dynamic reconfiguration capability is more and more requested to embedded systems and will become essential as soon as we will want to integrate them in “systems of systems” which composition is by essence much more dynamic than that of the classical systems.

Similarly to fault tolerance, reconfigurability has an impact both on the containers that implement the low level mechanisms and on the administration service that coordinates their activation (this last point is very important in order to minimize the risks for a system that a reconfiguration makes it unavailable in particular when the aim is to make systems tolerant to faults.) We should notice that these two new improvements (fault tolerance and reconfiguration) reinforce each others harmoniously because some fault management strategies will make use of the reconfiguration capability.

Integration to OpenDevFactory

In order to benefit from this type of architecture in terms of productivity improvement and to facilitate its adoption, it is important that the generation of the execution’s environment (the containers) and the various descriptors that are necessary to its operation (descriptors of components, descriptors of assemblies) be integrated in the tool chain developed in **OpenDevFactory**.

This integration requires that there exists an application model that correspond to the application’s execution platform view. One of the project’s objectives will be to define this execution platform’s meta-model. Then specific plug-ins will be designed in order to allow the generation of descriptors from the application model and the triggering in a very transparent way of the generation tools environment.

Perspective

Using Eclipse technology has significantly improved the way Usine Logicielle has addressed the integration problem:

- Eclipse framework provides the extensibility mechanisms that are used to extend the workbench with specific features.
- In addition, Eclipse provides an integration facility which is made from the EMF development environment and the ATL transformation engine.

These two technologies have radically changed the way tool chains can be assembled: Specific components can be developed directly as Eclipse plug-ins while existing tool chains can be gracefully integrated at a reasonable cost within the Eclipse Environment. As a result, tool chains can be cleanly build from existing ones using one single technology class and avoids expensive ad’hoc integrations. This opens a new field of application for Eclipse which is tool chains integration.

Project’s numbers

Partners: 19.
 Effort: 3,2 M€.
 Duration: 30 months.
 Sponsor: [Sustem@tic](#) Cluster.

Pôle de compétitivité



SYSTEM@TIC
PARIS-REGION
Concevoir et maîtriser les systèmes complexes

Usine Logicielle Partners



THALES



TRIALOG

