UMR IRISA

Project-Team PASS

Processes for Adaptive Systems

Brest

Activity Report

2016

1 Team

Team leader
Antoine Beugnard, Professor

Administrative assistant
Armelle Lannuzel

Permanent staff (Telecom Bretagne)
Jean-Christophe Bach, Associate Professor
Fabien Dagnat, Associate Professor, HDR
Julien Mallet, Associate Professor
Siegfried Rouvrais, Associate Professor
Maria-Teresa Segarra, Associate Professor

Postdoc
Fahad R. Golra, Postdoc since december 2015

PhD students
Ngoc Tho Huynh, Vietnam grant, since december 2013
Bastien Sultan, ARED & Chaire Cybersécurité des SystÃ¨mes Navals grant, since october 2015
Nicolas Szlifierski, DGA grant (Pôle Excellence Cybersécurité), since october 2016
Étienne Louboutin, Chaire Cybersécurité des SystÃ¨mes Navals grant, since november 2016

Associate members
JÃ©rÃ©my Buisson, Assistant Professor, UBS
Sylvain Guérin, Software Engineer, Openflexo
Christophe Guychard, Software Engineer, Openflexo
Vincent Leildé, Software Engineer, Openflexo

2 Overall Objectives

2.1 Overview

Adaptive software aims at adjusting various artifacts or attributes in response to changes in the system operating environment. By operating environment we mean everything that may affect the software system behavior and its properties, for example changing requirements, execution platform, and resources. Adjustment can be performed at design time and leads to the disruption of software operation. However, the higher demand for ubiquitous, pervasive, embedded, and mobile environments, has led to the need of runtime adaptation, i.e., without software disruption. Runtime adaptation increases software complexity as the software itself has to cope with variability. Complexity is even increased if the software system is distributed. Indeed, adjusting operations may involve artifacts running on different execution platforms and adaptation mechanisms available on them may vary.

On the other hand, software complexity is traditionally tackled by focusing on the software product itself or its related design process. The first one mainly proposes artifacts such as languages, software architectures, frameworks, and middleware that help on mastering complexity at a manageable degree for designers and maintainers. The second one is interested in providing a methodology (process) to design a software system that satisfies users requirements. It guides developers in taking into account important functional and non functional concerns and may be reused for each development. Both areas offer different but interesting approaches that help in managing software complexity.

The PASS team aims at combining both approaches to master adaptive software complexity. More specifically, it aims at defining system and process models and their relationships in order to ease the development and the adaptability of software systems. The mottos of the PASS research are:

1. *Adaptation requires to reify the software design process and trace its enactment.*

   In order to build adaptive software, it is necessary to reify its design process, rationales behind it, and to consider adaptivity as a first class property. Simply knowing the current variant of the software and the targeted one, as in traditional reflection-based approaches or more recently in DSPLs (Dynamic Software Product Lines), is not enough. Indeed, they do not consider relevant information for adaptation such as when it is possible to change, what can be changed, and how building blocs state should be managed.

2. *Processes are software.*

   The description, specification, and design of processes should follow the same process as software [Ost87]. As a consequence, current standard process models such as BPEL or BPMN are inappropriate since, they poorly distinguish specification from design and enactment.

3. *Components have distributed access points.*

   Classical software component models are bound to their physical execution targets (devices) and are connected through connectors (bindings) that ensure the assembly. We propose a paradigm shift releasing the software component model dependency from its underlying infrastructure. As a consequence, the access points of a single component can be distributed among various devices and (remote) communications are managed inside components.

[Ost87] L. Osterweil, "Software processes are software too", *in: Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, p. 2–13, 1987.

4. *Safety and security rely on controlled processes and specifications*

   We advocate the safety of software and their security rely on the quality of their specification (contracts) and on the mastering of their construction process. Specific activities and tools may be necessary during the development process to ensure security of the built software.

Our research is intended to be *vertical* in the sense that all aspects of systems are of interest: specification, design and implementation.

Solutions to enact these mottos are at the intersection of model driven engineering, software architecture, process modeling, verification and validation, and distributed systems. We intend to scientifically contribute to the three first domains (model driven engineering, software architecture, process modeling).

In this scientific context, we are interested in three kinds of systems: large scale, highly dynamic (mobile, P2P) and strongly constrained (real-time, embedded) where non functional properties such as safety, performance and reliability are important. For instance, ambient assisted living, satellites, cloud, mobile games, pervasive learning, etc.

On the application side, the PASS team explores the application to software systems such as satellite, home-automation, multi-users games, etc but also to non-software systems such as higher educational systems.

2.2 Key Issues

In its current state, PASS studies the following key issues:

- Systems are described with many models. Models of the system itself, but also models of the process development.

  How models can be federated? What are the nature of models connections?

- Among models connections, the trace between a requirement and its translation into a specification is the foundation of requirement engineering.

  How model federation can be used to improve the security of refinement?

- Changing multiple components requires the identification of the scope in space (what components?) and time (when?) of the adaptation.

  How can we be sure the components to be changed reached a state allowing the adaptation?

- Hiding information (especially specification) is a way to protect systems against wrong uses.

  What kind of obfuscation operations and process can be define to control the level of hiding?

- Non-functional properties (e.g., security, performance, dependability, scalability) and other quality attributes can be considered as drivers for software and enterprise architecture design.

We started a focus on localization. Dealing with the placement of pieces of software is almost always relegated to the configuration and deployment stage. We believe that localization issues have to be considered earlier in order to propose dedicated design process and tools for verification and automatic deployment and adaptation.

How methods, tools, models and processes could support non-functional (and especially localization) specifications for analysis?

- Software deployment relies on packet distribution with coarse-grain updating operators.

  Are there other levels of deployment? Are there other units of adaptation? What are the proper control primitives for updating software?

- Educational Systems are manageable from a systemic perspective. Models may provide a framework for strategic planning, continuous and adaptive quality enhancement, and be the pillars of renewal processes to meet the European 2020 strategic Agenda for higher education.

  How to qualify and quantify the quality attributes of educational systems as a whole? How to architect educational systems and with which models? What kind of adaptation and rationale decisions are required? How to continuously enhance and flexibly renew such systems at system-level?

- Graphical models and methodologies used in different engineering tasks constrain the users to a defined language, thus taking away the expressiveness that might help in the development stages of a software or a system.

  How to improve user experience in software and system modeling? What methods can be used to add more flexibility for the modeler to manipulate with the model and its language at the same time? How to separate the graphical representations from the models themselves?

3 Scientific Foundations

3.1 Modeling and Metamodeling

Modeling is a central activity of science and engineering. Computer science brought the idea to think about models as objects of interest. The reification of models lead to study the ways models are described. In the context of software design, with the importance of the UML modeling approach, the Object Management Groups came up with the Model Driven Architecture (MDA) initiative. This approach is generalized to system modeling and system of system modeling. Earlier, computer scientists had defined a theory of languages and grammar, that can be compared to the model and metamodel approach of the OMG. In fact, both approaches are very close; formalisms and theory have changed but the underlying principles are close, if not identical.

In [Mor67], W. Morris has shown that learning of models is not learning of modeling and had made suggestions to enhance the process of developing the intuitive skills associated with

[Mor67] W. T. Morris, “On the art of modeling”, *Management Science 13*, 12, 1967, p. B707–B717.

model building. Almost 50 years later, the understanding of models has improved thanks to its formalization and the development of tools. However, current approaches (UML, DSL, MOF, EMF, XML, OWL, etc.) are still not sufficient and users (scientists, engineers, developers, etc.) need *flexibility*, *merging* and *preciseness*. Flexibility, since existing metamodels are not offering all required concepts; as an example, UML proposes a partial solution with its profiles. Merging, since different points of view need to be combined; the proliferation of architecture framework[Web14] is another example. And Preciseness, since the interest of tools is not only drawing or writing, but its also checking, proving, testing, generating code or documentation with a controlled semantics.

Glossary :

Metamodel A set of rules that a model has to conform to. The definition of the set of rules can be words, drawings, formal languages, etc. Some descriptions are more easily computed than others.

Model A description of a System under Study. The description can be done with words, drawings, formal languages, programs, etc. Some descriptions are more easily computed than others.

3.2 Process Modeling

The description of processes is central to Software Engineering. From early life cycles that were coarse grain descriptions to the fine-grain description like BPMN, the reification of activities, of who is doing something and of what is produced, has always been an issue in (software) development.

The current process modeling languages appear to have two distinctive problems. First, they seem to ignore the importance of a consistent approach that handles process in all the stages of its lifecycle. Either a single process model seems to represent a process in all phases (e.g. at specification phase and the implementation phase) or it has to be transformed to a completely different approach for enactment. For example BPMN models processes in all stages of development through a single notation, however it does not offer the possibility to enact them. Consequently, process developers are bound to transform the models to BPEL for enactment. Second, most of the approaches focus on the flow of activities defining the order of (presumable) execution. Some approaches use event based mechanisms to induce reactivity, but still they their focus remains on the flow of activities. Approaches like Event-driven Process Chains (EPCs) use both types of inputs for the activities (events and artifacts) together, which clutters the process model. For instance, mixing at the same time data flow and control flow in a process, makes it hard to conceptualize the interactions of an activity with its context.

Software processes just like software systems are based on the notion of lifecycle, where each stage of development has different concepts to frame. Each phase of a software process organizes different factors and issues related to the degree of its maturity in terms of completeness. Fuggetta defined a software process as, the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product [Fug00]. But we tend to think more like the Osterweil's

[Web14] I. . Website, "Survey of Architecture Frameworks", `http://www.iso-architecture.org/42010/afs/frameworks-table.html`, visited 31/1/2014.

[Fug00] A. Fuggetta, "Software process: a roadmap", *in: Proceedings of the Conference on The Future* analogy of a software process, where he advocates that processes share the same nature and complexity as the software system and should be treated the same way [Ost87]. Thus we are of the view that software processes themselves need to be conceived (specification), developed (implementation), deployed (enactment) and maintained (monitoring). As the life-cycle phase of a software process advances, the focus on issues related to the process also changes. For example, in specification phase the focus remains on which artifacts would be handed over by the activity. On the other hand, for enacting a process the focus shifts to when should an artifact be handed over by this activity. During the enactment phase, the reasoning about the activity is not targeted towards the choice of its inputs and outputs, rather it is directed towards the related hows and whens.

Glossary :

Activity A time period dedicated to the production of Artifacts by persons playing their Roles in the Activity. An activity - as a program - has a specification, several implementations resulting from design choices, and many enactments.

Artifact Any human-made product resulting from an Activity. Documentation, source code, review, test plan are some examples of artifacts.

Process An organized set of activities, roles and artifacts.

Role An abstraction of a position denoting a responsibility in an Activity.

3.3 Software Components

Software components is an old idea. The first reference is usually attributed to M.D. McIlroy [McI69] in 1968. Since then, many component models were proposed and many surveys compare them. Components may be classified according to their lifecycle, to their interface specification, to their computational model, their implementation targets and many other dimensions.

Software components are units of composition, units of specification, units of management. As such they have their own development lifecycle/process. In [CSVC11], Crnkovic *et al* classify more than 20 components models with respect to their component lifecycle. It is worth noticing that these models do not propose any progression in their description. A component model is usually described at a selected level - often implementation, sometimes specification - with no references to its various forms from high level specification to low level implementation.

Software components as units are defined by their interface. It is now well admitted that a good way to describe these interfaces is through contracts [2]. However, if low level contracts (syntactic and semantics) are well established, higher levels (behavior and QoS) are seldom used.

Glossary :

Abstraction a mechanism and practice to reduce and factor out details so that one can

of Software Engineering, *ICSE'00*, ACM, p. 25–34, New York, NY, USA, 2000, `http://doi.acm.org/10.1145/336512.336521`.

[Ost87] L. Osterweil, "Software processes are software too", *in: Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, p. 2–13, 1987.

[McI69] M. McIlroy, "Mass produced software components", *Software Engineering Concepts and Techniques*, 1969.

[CSVC11] I. Crnkovic, S. Sentilles, A. Vulgarakis, M. Chaudron, "A classification framework for software component models", *IEEE Transactions on Software Engineering 37*, 5, 2011, p. 593–615.

focus on few concepts at a time.

Component a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.

Composition a system design principle that deals with the inter-relationships of components. A highly composable system provides recombinant components that can be selected and assembled in various combinations to satisfy specific user requirements.

3.4 Adaptation and reconfiguration

Adaptation can be understood in, at least, two ways. First, as a problem of plugging two (or more) parts together, even if they were not initially designed for that. Second, as the problem of changing something (a computation, a structure) to something else (another computation or another structure). Both are not totally independent. In his survey[Kel08], S. Kell proposed a large transdisciplinary survey of software adaptation techniques (first meaning). He discussed three principles that are advocated: component models, first-class connection and loose coupling. It is worth noticing this survey covers a spectrum close to the one presented in this report. For the second interpretation, the first abstract proposal comes from autonomic computing and is modeled by the MAPE-K loop[map05]. This model states that adaptation requires to share Knowledge and to process a loop of four activities: Monitor (the system and its environment), Analyze (compute and decide to trigger an adaptation), Plan (compute how to change) and Execute (actually do the changes). This model is largely adopted.

In a distributed component-based software, a set of components located in different nodes, cooperate in order to provide services. Making such a system adaptive requires including adaptation mechanisms that are themselves distributed. Several works exist that propose distributed algorithms for either deciding or planning changes [FDP+12]. Although those approaches contribute to simplify the task of building an adaptive software, they fail to provide an architectural view of the adaptation mechanisms. Such a view would allow designers to choose different distribution strategies according to their needs, i.e., some components may be managed in a distributed manner while others may not.

Glossary :

Adaptive software a software that has the ability to adapt at runtime to handle such things as changing user needs, system intrusions or faults, changing operational environment, and resource variability.

Dynamic or runtime adaptation a change on a software that takes place after deploying it. It includes deciding about the change to perform, planing the necessary operations on the software, and executing them.

[Kel08] S. Kell, "A Survey of Practical Software Adaptation Techniques", *Journal of Universal Computer Science 14*, 13, 2008, p. 2110–2157.

[map05] "An Architectural Blueprint for Autonomic Computing", *research report*, IBM, June 2005.

[FDP+12] F. Fouquet, E. Daubert, N. Plouzeau, O. Barais, J. Bourcier, J.-M. Jézéquel, "Dissemination of reconfiguration policies on mesh networks", *in: DAIS 2012*, Stockholm, Sweden, June 2012, `http://hal.inria.fr/hal-00688707`.

MAPE-K Adaptation process loop model consisting in Monitoring, Analyzing, Planing and Executing around a shared Knowledge.

3.5 Analysis and Verification

Verification covers a wide range of software engineering activities (code review, tests, simulation, model-checking, proof, static analysis, etc.). It aims to improve quality, safety and security of systems; and it is a core activity in critical software development.

An approach consists in analysing produced code or binaries (static and dynamic analyses), however due to the nature of the manipulated artifacts it is usually difficult to have formal verification at this level. Therefore, another approach consists in doing beforehead formal verifications such as model-checking and proof. It has also drawbacks: those verifications are less realistic because of the fact they focus on abstractions of the product. In practice, when safety and security are a real concern, a combination of both approaches with many techniques is used.

Another point of view is to focus on the development chain instead of the product itself. This approach advocates that an unreliable development process cannot produce reliable software. It can be done at different levels, from technical and low levels (code generation, certified compilers, qualified tools) to more abstract levels (process certification, norms). In this context, the notion of traceability is essential to be able to follow the whole development process, to link any artefact (code, documentation, tests, etc.) to tasks or requirements. When focusing on the construction process, traces are continuously produced, inspected and verified. Properties that have to be ensured during the whole process are expressed and checked in order to detect flaws.

Glossary :

Traceability is the ability to keep the relations between *objects*. It can be done at different levels: between source code and binaries during compilation time; between requirements and specification; between people and ressources within the security process (clearance managment).

4 Application Domains

4.1 Component Software Adaptation

Participants: Antoine Beugnard, Fabien Dagnat, Ngoc Tho Huynh, Julien Mallet, Maria-Teresa Segarra.

Apply the mottos 1, 2, 3 and 4.

Domains : Home-automation, HPC, Large scale applications.

In this research area we are interested in providing developers with methods and tools to easily build (distributed) adaptive software. To achieve this goal, we are currently working on the definition of 1) a generic, customizable model for distributed dynamic adaptation mechanisms at the middleware level, and 2) a set of tools that may be used by adaptation designers in order to ease the customization of our model for their needs.

Models for distributed dynamic adaptation mechanisms. To tackle current limitations we have been working on a model for distributed and coordinated dynamic adaptation [15]. We have proposed a model that constrains designers considering what are the dynamic adaptation services they need for their application and how many instances (and location) of the selected services should be created. According to the number of instances of the different services, coordination components may be mandatory and different coordination strategies are available to be used by designers, whether for decision, planning or execution services. In order to validate this work, we considered the distributed data management application domain and we intend to dynamically change data replication policies depending on the execution environment.

Development process for adaptive distributed software. The second research area is related to the automatic generation of dynamic adaptation mechanisms: decision making, planning, and execution. This work is based on the main result of Eric Cariou and Chantal Kaboré PhD work, the Cloud Component approach, a communication abstraction (Cloud Component) used to model distributed services and a refinement process that allows for automatic generation of architectural variants. It is included in a current research trend that proposes DSPLs (Dynamic Software Product Lines) and MDE (Model-Driven Engineering) to automatically compute the actions to be performed to build the target variant when an adaptation is needed, i.e., to compute the "adaptation plan" or "reconfiguration plan". In current approaches, these actions correspond to operations on the application architecture (add component, remove component, etc...). These works propose general enough approaches that can be applied to all model-based applications, but they fail to provide a systemic approach for automatic generation of adaptation plans.

Compared to these approaches, in this research area we are interested in investigating how to generate not only architectural modification actions but also (all the) actions of an adaptation plan including application-dependent actions. We claim that during the development process relevant information may be collected then exploited to compute adaptation plans.

We have been working on one type of application-dependent actions: those related to the data managed by the components affected by an adaptation. Indeed, when a component is modified or replaced its data should also taken into account. These data are ignored by most of current approaches that assume stateless components. To tackle this issue, we have been working on two areas :

- The first one aims at helping a developer to explicitly add information related to the data managed by the application. This information is related to the entities that manage data and the operations used to managed them. Such information is then used to compute data-related actions to be added to adaptation plans. More specifically, data transfers between components can be computed and added to the adaptation plan.
- The second one is related to the concept of variation point in DSPLs and its reification at runtime. Indeed, current approaches assume that all variation points in a DSPL of an application should be available at runtime: each time an optional feature is present in the DSPL, the application should be able to change at runtime to include or remove the feature. In this work, we claim that each variation point identified in the DSPL may

be reified at runtime and we investigate a development process that allows to reify such variation points [13].

4.2 Process Design

Participants: Jean-Christophe Bach, Antoine Beugnard, Fabien Dagnat, Fahad Golra.

Apply the mottos 1, 2 and 4.

There is an increasing trend to consider the processes of an organization as one of its highly valuable assets. Processes are the reusable assets of an organization which define the procedures of routine working for accomplishing its goals. The software industry has the potential to become one of the most internationally dispersed high-tech industry. With growing importance of software and services sector, standardization of processes is also becoming crucial to maintain credibility in the market. The design of a software development process follows a lifecycle that is very similar to the software development lifecycle. The process has to be designed abstractly then refined until it become an executable asset. Moreover, the multiple phases of a process development lifecycle follow an iterative/incremental approach that leads to continuous process improvement. This incremental approach calls for a refinement based strategy to develop, execute and maintain software development processes.

To be fully reusable and analyzable, a process should be modeled. Current standard models like BPMN or SPEM represent processes as artifact flows among a set of activities. To fully ensure that a defined process is applied, the process should be executable. Here executable means that the process should be enacted with the help of software. There is a gap between being executable and the abstract representation of languages such as BPMN. We propose to use refinement and model oriented technology to reduce this gap. First, a non-executable abstract model that is highly reusable is designed (or reused) and then it is refined to a concrete executable model of the same process that is fully instantiated. Furthermore, we use a concrete model relying on event rather than connected flow to be able to support the reconfiguration of a process at runtime.

4.3 Complex Systems

Participants: Jean-Christophe Bach, Antoine Beugnard, Fabien Dagnat, Julien Mallet.

Apply the mottos 1, 2 and 4.

Complex systems are systems composed of many components which may interact with each other. We are interested in such systems because of the need of many points of view to describe them. Each point of view are itself described by models. Today, the main issue is to be able to ensure consistency among all these models. In order to tackle this issue, we develop an approach called “models federation” where models are kinds of components that can be interconnected. Hence, entities of a model can be linked to entities of other models with computable relationships. Thanks to theses connections, constraints or dependencies can be propagated from entities to entities.

We apply our approaches - explicit process, model federation - to systems like naval units (vessels), VHF Omni Directional Radio Range (VOR) or automated urban metro subway systems. Note that we do not cover all stages or the full systems with all points of view but only a part of them.

4.4 Educational Systems

Participants: Siegfried Rouvrais, Antoine Beugnard.

Apply the mottos 1, 2.

An educational system provides services to students and society. In a global context and for strategic alignment, the management of educational transformation is of decisive importance to Higher Educational Institutions (HEI) and their regulation bodies. During the last decade, various models of quality management have emerged in the form of quality assurance standards (e.g., accreditation bodies, European or international frameworks) that guide educational program leaders, designers or deans of academics to evaluate and improve educational systems including syllabus, curricula, workforce, workspaces, support services, or administrative processes, or educational program engineering concerns. However, to date, educational system complexity is traditionally tackled by focusing restrictively on the course or curricula contents. There is no standard commonly accepted for conceptually describing and evaluating the overall complexity of educational systems, restricting the modernization agenda of HEI nationally, in Europe, and internationally.

The actual complexity of educational systems and the dynamic of their quality requirements call today for sound engineering principles. The aim of this research track, domain specific, is thus to define and provide methods, processes and tools to design and evaluateeducational systems that satisfy quality requirements with efficient resources. Based on modeling, metamodeling and business process modeling foundations, well defined models and processes for continuous improvement are to stimulate a program reform, transformation or renewal to better align with requirements (among other learning outcome and competency EU requirements). Educational frameworks can be defined, with varying degrees of rigor, to prepare and organize various views and models for their stakeholders, management processes, and systematize analysis and evaluation.

From this perspective, applying the mottos 1, 2 and based on a convergence of system engineering, enterprise architecture, and sound educational design principles, this PASS research track expects to provide in-depth and usable models, methods, and tools to address issues of educational system design, transformation, and adaptability.

5 Software

5.1 Pymoult

Participants: Sébastien Martinez [contact point, previous PhD], Fabien Dagnat.

When willing to update software at runtime, one choses a platform that will give mechanisms to modify the software during its execution. For each problematic of dynamic software updating, a given platform would offer one unique answer. This means that the choice of the update mechanisms to use for modifying a running application does not belong to the developer of the application or of its updates.

Pymoult is a Python library that enables application and updates developer to choose the mechanisms they want to use when designing dynamically updatable applications or dynamic updates. For that purpose, Pymoult provides the mechanisms implemented in several platforms from the literature. Pymoult also proposes a generic API for designing updates independently from any platform used.

Pymoult uses a modified version of Pypy: a Python interpreter written in Python. This modified version of Pypy allows specific interpreter operations such as intercepting object creation or capturing and modifying the runtime of threads. Pymoult was tested on several test cases from other platforms, proving itself to be a good platform for testing and prototyping dynamic software updates.

Pymoult is a free software available online at `https://bitbucket.org/smartinezgd/pymoult`.

5.2 Openflexo framework

Participants: Antoine Beugnard, Fabien Dagnat, Fahad Golra, Christophe Guychard [contact point], Sylvain Guérin [SCIC[1] openflexo].

Openflexo is an open-source (GPL3) business architecture platform that supports collaborative agile missions by seamlessly transforming multifaceted enterprise models into business oriented deliverables. Openflexo relies on the reification of models and their componentification. Hence, a model can make references to other models and then to parts of them. This allows models federation. This approach permits to federate many kinds of models such as source code, structured textual documents, databases, spreadsheets or more classical model like EMF...

FORMOD (next section) is developed thanks to Openflexo.

5.3 FORMOD

Participants: Antoine Beugnard, Fabien Dagnat, Fahad Golra, Christophe Guychard [contact point], Sylvain Guérin [SCIC[1] openflexo].

Formose Requirements Modeler (FORMOD) is a requirements elicitation, modeling and management tool. It aims to model software project requirements of critical and complex systems in a formal way. It allows the elicitation of requirements from different software project artifacts like project proposals, feasibility reports, standards, etc. The identified requirements from these artifacts are then specified in a central *requirements management view*. Then, it uses SysML-KAOS approach for the development of goal models, corresponding to the identified high-level requirements. It also helps in refining these requirements to a coarse grain level, where they become unambiguous and verifiable.

[1]Société Coopérative d'Intérêt Collectif: a social and solidarity economy company dedicated to cooperation. `http://www.les-scic.coop/sites/fr/les-scic/`

FORMOD uses model federation at its core, which helps in creating and maintaining dynamic links between several models of different paradigms *e.g.* requirements specification (textual), goal models, project artifacts, system design, *etc.* It is based on the requirements engineering methodology for critical and complex systems, developed under ANR Project FORMOSE. This methodology connects the artifacts, models and views of different paradigms used in the process, so as to keep them synchronized all along the project development life cycle. Synchronization of models is exploited to develop traceability links that can be operationalized. Operationalization of traceability links is ensured through the behavioral definitions of the mappings between model elements. These mappings with behavior are defined using model federation approach. Once such mappings are developed between different models used in the requirements engineering process, new instances of the artifacts can be generated and the old instances can be kept up to date.

6 New Results

6.1 B. Sultan's PhD

Participants: Bastien Sultan, Fabien Dagnat [15].

Ships are complex systems operated to perform a given set of missions. Such systems can be affected by vulnerabilities which, when exploited, can impact their missions. Thus it is necessary to apply patches to mitigate these vulnerabilities, and fundamental to ensure that deployed patches does not negatively affect the system's dependability. Our work aims to design a patch management process applied to naval systems. Designing such a process raises two issues : complex systems modelling and impact assessment. We decided to gather several models (functionnal decomposition, network topology, deck plans, . . .) through models federation. We then deduce a comportmental modelling of the system and safety and security properties. Indeed, in order to assess patches and vulnerabilities impacts on the system's safety, we have to model the system's behaviour and the system's missions. We can rely on timed automata for this purpose, since missions and subsystem's behaviour can be seen as state sequences. Thus we can model our system and its missions through two classes of automata : a System Automata, modelling the ship's behaviour and a Mission Automata, describing the steps sequence of the mission. The System Automata are timed, in order to model the operating times of the system's assets. Then we introduce a set of properties the system must verify to fulfill each of its missions. When a vulnerability is discovered, a cyber-attack occurs or a countermeasure is deployed, the changes on the "real" system are modelled by mutating the System Automata. Applying the mutations on the System Automata, we can model the comportemental effect of any vulnerability, attack or countermeasure at the subsystem level. Then, by checking the model through the properties, we can compute the overall impact on each mission.

6.2 Ngoc Tho Huynh's PhD

Participants: Ngoc Tho Huynh, Maria-Teresa Segarra , Antoine Beugnard [10][11].

The aim of this thesis is to propose a development process to build adaptive software architectures based on variability modeling. This process consists of a set of activities in which variability modeling plays an important role. In our approach, the variability is modeled by using Common Variability Language (CVL), and used all along the development process. In CVL, a base model is used to represent the software architecture of a product line. When the variability model is configured, a particular product architecture is deduced. In order to build adaptive software architectures, existing approaches include all the elements of the architecture of the product line even if they are not necessary for a particular product. Therefore, some elements will be available at runtime and never be active which limits deployment targets. Identifying the elements that will be used for adaptation is necessary. We extend the CVL metamodel to specify the necessary information for this purpose.

On the other hand, a reconfigurator needs to be added into the architecture to realize adaptation. However, determining the best moment to adapt is a difficult task. The adaptation process should ensure not only the validity of the new version but also should preserve the correct completion of ongoing activities and minimize disruption of the software. Therefore, an adaptation mechanism is proposed based on the concept of transaction. In our approach, transactions scope is specified at design time in the variability model and exploited at runtime to find the best moment for adaptation.

6.3 Continuous requirements engineering

Participants: Fahad Golra, Fabien Dagnat [8].

Current research in software engineering revolves around the concepts of iterative development, reducing time to market and product quality. Agile manifesto is built around the notions of small iterations, where a complete phase of software development is not dedicated to requirements engineering. The idea is to gather the principle requirements, develop them in small iterations and then refine the requirements alongside development. These refinements are used to update the product backlog, which consequently is implemented in the later development cycles. With this shift of perspective, requirements engineering is not restricted to the initial phases of software development and has become a continuous process. Putting this continuity into practice needs tools and techniques that are tightly integrated with other software development phases. Such requirements engineering methods need to consider different concerns of all the stakeholders involved in the development process. Process modeling approaches like KAOS suggest the use of multiple views as goal, responsibility, object and operation.

A multi-view approach, maintaining a dynamic link between the requirements and other artifacts of the system can offer the desired continuity. We propose using model federation for modeling the requirements and connecting them to other model (artifacts). This means that all these models used in the requirements engineering process remain connected to the requirements model, hence updating of the specified requirements as soon as the information resources are modified. The support for linking information resources to the requirements and maintaining them through synchronization allows a continuous requirements engineering process all along the software development lifecycle, regardless of its phase and iteration.

Different views/models used in the system might not adhere to a single paradigm. KAOS models are defined using MOF, but other domain models used in software development may not belong to the same paradigm *e.g.* legacy systems, databases, reports, spreadsheets and other specific domain models like building architecture models, electric circuit diagrams, CAD models, *etc.* Many of these models might be needed as information resources to update software requirements. For such cases, model federation approach can integrate models from different paradigms to create new concepts using virtual models.

6.4 Modularity of modeling

Participants: Antoine Beugnard, Fabien Dagnat, Fahad Golra, Sylvain Guérin, Christophe Guychard [6][7].

In our research and teaching activities, we are used to model a lot. We produce formal models for statical analysis, semi-formal models with UML for instance (see 3.1) or even simple drawings. This led us to identify and describe modeling situations [7].

We can summarize this situations in five categories:

from the concept to the instance It is the most frequently *equipped* approach. The toolkit brings its set of concepts that can be used for modeling. The metamodel do exists before all, the modeler refers to it. Some tools allow when the realization is impossible to enlarge the set of concepts and then realized. The instance does not exist without the concept.

from the instance to the concept This is the most commonly *used* approach before the passage to the previous case. We draw on a board, a piece of paper or a drawing tool. We identify the concepts then we change and configure tools to be able to derive concepts into instances.

recognition of an interpretation Intermediate approach, it allows the independent existence of concepts and instances, and finally recognize their interpretation link.

composition Independent from the interpretation link, composition applies to both instances and concepts. It allows to introduce the concept of composition that is used to represent associations, relationships, content, etc.

evolution The evolutions highlight the dynamic of construction of the two involved levels: instances and concepts. They can be independent of the interpretation or not.

This approach is equipped in the Openflexo modeling environment (see 5.3) as a Free Modeling module. This tool allows the modeler to draw independently instances or concepts, derive an instance from a concept or identify a concept from instances, and recognize an interpretation between already existing instance and concept.

6.5 Adapting Educational Systems for Continuous Change to meet Quality Criteria

Participants: Siegfried Rouvrais [5][12][13][14].

Thanks to the European Erasmus Plus 2014 – 2016 KA2 Strategic Partnership QAEMP project with seven other institutions in Europe (Quality Assurance and Enhancement MarketPlace for HEI), PASS members have formalized (i) a generic model of reference including 28 quality criteria and a collaborative process model for higher educational institution quality enhancement. The criteria draws upon an international super-set of criteria from engineering accreditation systems like ABET, CTI, EUR-ACE, CEAB or Engineers Australia, and is extensible. They are scored on process maturity levels as found in the most recent ISO/IEC 33020:2015 series (Information technology – Process assessment – Process measurement framework for assessment of process capability, JTC1/SC7, stage 60:60), and complemented by contextual parameters such as the size of the study program, disciplinary main focus or geography. The cross-sparring formalized process [5] focuses on the roles and responsibilities of the involved stakeholders, the inputs and the outputs. The assessment process is supported by an assessment model. The assessment model is based on a reference model that defines a set of best practices [13] (or standards) related to the domain that needs to be assessed. The measurement framework provides the maturity levels to be considered and contains a set of assessment indicators which support the ratings against the various standards. A potential request as domain specific proposal for Higher Educational Institutions, aka an ISO Publicly Available Standard (PAS) is now to be investigated.

6.6 Other results

Publications: [1, 2, 3, 4, 9]

6.7 Assessment of achievements

Participants: All participants.

The results achieved by the PASS team must be compared with the key issues presented in the objective section. Not all key issues have deserved attention yet. However, a few of them have been sufficiently well explored to start and draw conclusions.

The first key issue is “How models can be federated? What are the nature of models connections?” The experiments realized with Openflexo (see 5.2) give us confidence that we are exploring interesting paths. We have started a formalization of the approach.

The second key issue is “How model federation can be used to improve the security of refinement?” Making explicit the processes, traces, decisions is the approach we try. It relies on the possibility to combine (federate) informations from many sources. The formose project (see 7.1) should bring us better undestanding of requirements (for requirements management).

The third key issue “How can we be sure the components to be changed reached a state allowing the adaptation?” lead to the proposal by of Ngoc Tho Huynh of a process where the components to be changed are isolated thanks to software clamps that allows to reach a stable state before reconfiguration.

7 Contracts and Grants with Industry

7.1 Projet ANR Formose

The Formose project is an industrial research project that aims to provide a formally-grounded, model-based requirements engineering method for critical complex systems, supported by an open-source environment. Formose is a 48 months project proposed by a consortium made up of 2 academic partners (LACL - project leader - and Institut Mines-Telecom) and 2 companies (THALES and ClearSy).

It is well-known that requirements engineering (RE) is critical in software and system design. Indeed, a major part of the cost of software and system development is known to be traceable to the understanding of the problem domain and requirements. Today's current industrial practices and tools are not sufficiently efficient. They mainly consist of in-house processes, lessons learnt documents, and requirements management tools (word processor macros, traceability tools, requirements database). Even if, in the last decade, much research on this topic reached maturity, recent studies have pointed out that issues remain open. The Formose project addresses several challenges raised by these issues, the most crucial in the domain of RE for critical complex systems. They concern the need of taking into account the high complexity of such systems, the need of a better integration of RE with verification and validation techniques to ensure a better quality of requirements, and more generally the need of method guidance and tool support during the process of elaborating high quality requirements models.

First, we defined a requirements modeling language integrating basic concepts of existing languages, such as KAOS or Tropos/i*. To this classical concepts we added new ones to take into account the specific characteristics of critical complex systems:

- their abstract architecture is considered by allowing requirements to be defined at different abstraction layers and verifying their consistency;

- the language allows to specify not only non-functional requirements related to safety and performance but also specific requirements related to the presence of different operational modes and reconfigurations in such systems.

The language uses multiple views (natural language, graphical notations, formal notations) of requirements to be understandable and easily manipulated by all the stakeholders. For verification purpose, we are working on the B language support and started to look at the support of the timed model-checker UPPAAL. A first version of a highly customizable process for requirements engineering of complex systems is currently being validated. To support the enactment of the process we will use the OpenFlexo platform that will integrate not only the ad-hoc tools developed during the project but also the existing verification tools (Atelier B and UPPAAL model-checker). We also envision to ensure a two way exchange: OpenFlexo will be able to send proof obligations to a prover and will also be able to get the answer from the prover and interpret it to present it to engineers, using adequate representations.

7.2 Erasmus+ KA2 EU project: Quality Assurance and Enhancement Marketplace for Higher Education Institutions

Improving European education and training system quality has been set as a key target in Europe's strategy to become a smart, sustainable and inclusive economy by 2020 (Council of the European Union, 2010). These objectives are more specifically defined in the so called Modernization Agenda (EC 2011). More specifically it sets a goal to improve the quality and relevance of higher education. In this process, external evaluation and self-assessment are seen in a key role.

There have been many European projects on Quality Assurance, but they do not cover still existing problems with accreditation such as required resources, complexity, delays between the evaluation rounds, poor feedback, poor quality loop and distance from continuous education development. There is a need for more flexible evaluation models and processes with peers to reduce the inertia of heavy accreditations/evaluations in HEIs.

QAEMP-project (2014-2016, grant $2014-IS01-KA203-000172$) proposed a flexible and collaborative methods, processes and tools for degree program / higher education evaluation. The inventiveness of the project lies in the collaborative model of quality assurance that can complement accreditations and existing QA systems. The project promotes and strengthens the European cooperation in quality assurance while designing and piloting new kind of continuous, accessible, cooperation based model, tools and a virtual platform (the Market Place) supporting so called cross-sparring between institutions. Cross-sparring is to be understood as a process to make feedback more collaborative, concrete and objective, thanks to critical, but discreet brainstorming sessions, where strategies can be discussed, repeatedly contributing to the quality assurance with a critical external view. The Market Place serves as a tool for finding the best possible sparring partners as well as a forum for networking, sharing experiences, information and best practices.

The QAEMP project consortium consists of eight European higher education institutions: Reykjavik University, Iceland; Turku University of Applied Sciences, Finland ; Aarhus University, Denmark; Helsinki Metropolia University of Applied Sciences, Finland; Umeå University, Sweden; IMT Atlantique (Telecom Bretagne School of Engineering), France; Aston University and Queens University Belfast, United Kingdom.

7.3 DCNS and French Naval Academy collaborations

For the educational system track field, PASS and the TREE research Group at IMT Atlantique (Telecom Bretagne School of Engineering) initiated in 2016 a collaboration with the French naval Academy on decision making rationales and skills for engineers. Professional environments are more than ever volatile, uncertain, complex, and ambiguous. The notion of VUCA (Volatility, Uncertainty, Complexity, Ambiguity) was introduced by the U.S. Army War College to describe the multilateral world. It has been subsequently used in emerging ideas in strategic leadership that apply in a wide range of organizations. We now face major risks (e.g. earthquake, nuclear accident, toxic gas, intrusion, terrorist attacks, etc.). For Weick (The Collapse of Sensemaking in Organizations: The Mann Gulch Disaster), the organizational reliability depends on the ability of the actors to organize and reorganize, in order to anticipate and cope with unexpected and crisis situations. The analysis of risk and decisions is increasingly being considered as a critical part of contemporary management practices and to be the focus in software and system engineering also. Increased understanding of risk and uncertainty plays a major role in improving general management and engineering disciplines leading to safer practices, increased strategic, tactical and operational efficiency, cost reduction and improved forecasting and planning accuracy.

The French Naval Academy and a PASS member proposed a qualitative analysis to study decision making skills among engineers in real situations. This starter work resulted in an international publication [14], to be integrated in a Springer Book chapter in 2017. This collaboration may in the medium term find echoes in the models and processes for software and system engineering, requiring flexibility and adaptability with also a human dimension, and under reliability, safety, and security quality constraints.

7.4 Quarkslab

We collaborate with Quarkslab, a high-end cybersecurity company, highly skilled in vulnerability research and design of security solutions, in the context of a PhD thesis tackling the key issue "What kind of obfuscation operations and process can be define to control the level of hiding?"

7.5 Openflexo

We participate to the creation of the Openflexo, a social and solidarity economy company dedicated to cooperation for the development and use of the Openflexo modeling toolkit (see 5.2).

This company was created with 3 persons and runs a network of 10 cooperators (other companies or individuals).

8 Other Grants and Activities

8.1 International Collaborations

- Initial meetings were carried out with Jörg Kienzle's team at McGill School of Computer Science, Canada for probable collaborations. A Demo on *model federation* approach and *free modeling* was presented in a video conference. They showed interest in using model federation techniques and tools for the course work in their graduate program.

- A collaborative project for an undergraduate project was carried out with the team of Valentino Vranic at the Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies of Slovak University of Technology, Bratislava. The student that carried out the project was Jacub Jasan. The project concerned the modeling UML diagrams using model federation approach.

- Vincent Englebert, from Université Catholique de Namur. (Metamodeling)

- Siegfried Rouvrais is the French correspondent of the 2014-2018 Erasmus Mundus Action2 INSPIRE Project (INternational Science Promoting Innovation and entREpreneurship) with South Africa. He was visiting scholar for one month at University of Cape Town in March 2016.

- The PhD thesis of Ngoc Tho Huynh is a grant from the Vietnam-French cooperation.

8.2 National Collaborations

- We have informal project with David Chemouil and Julien Brunel from Onera, Toulouse, on an Alloy-like DSL for specifying components.
- We hosted Aymerick Savary (ATER at LACL) for two weeks at our campus in Brest for a collaborative project. He worked with us for the development of a technology connector for B-method for requirements formalization.
- Maria-Teresa Segarra is actively involved in the MERITE Project which aims at helping teachers of elementary and middle schools on introducing technical and scientific concepts to students. She is involved in three types of actions: teaching elementary and middle school teachers scientific concepts related to basic programming and robotics, design pedagogic resources for teachers and students, and helping teachers in their own classes.

9 Dissemination

9.1 Involvement in the Scientific Community

- Antoine Beugnard has been member of the programming committee of CAL'2016, INFORSID'2016 and WETICE'2016. He was also member in several doctoral committees: Fabien Dagnat (HDR) on "Du génie logiciel pour déployer, gérer et reconfigurer les logiciels", Christelle Urtado (HDR) on "Contributions to component-based software engineering - composition, reuse and evolution", Amine Benalallam on "Model transformation on distributed platforms, decentrilized persistence and distributed processing", and Mu'ath Al-Shaikh on "Protection des contenus des images médicales par camouflage d'informations secrètes pour l'aide à la télémédecine.". He is member of the "scientific board" of the school Saint-Cyr Coetquidan.
- Siegfried Rouvrais is involved in the following communities:
  - is the French representative for CDIO (a framework which provides students with an education stressing engineering fundamentals set in the context of Conceiving & Designing & Implementing & Operating real-world systems and products). He has been attending international meetings since several years (e.g. TUAS in Finland, ISEP Porto, and Trinity College Dublin in 2016). He has been elected in 2013, during the MIT/Harvard Conference, as council member for a one year term. The CDIO Initiative (acronym for Conceive – Design – Implement – Operate) is a structured educational framework for preparing the next generation of engineers, including more than 100 prestigious Universities for STEM and engineering education worldwide. Dr. Rouvrais works closely and on a regular basis with the European Region.

  - was upgraded as IEEE Senior Member in 2016 (CS and Education chapters).
  - is member of the French AFNOR working group IQLS (*Ingenierie et qualite du logiciel et des systemes*), with a focus on Processes and methods for software development and software engineering, so as IT service governance and Capacity Maturity Models (aka ISO 330xx).
  - is reviewer and TPC member for the following conferences: IEEE Frontiers in Education, IEEE International Conference on Learning and Teaching in Computing and Engineering, international CDIO Conference.

- Jean-Christophe Bach is reviewer for SLE2016 (9th ACM SIGPLAN International Conference on Software Language Engineering), IWST2016 (International Workshop on Smalltalk Technologies) and INFORSID 2016.

- Fahad Golra reviewed an article for the Journal of Systems and Software. The title of the article is "A tool to support the definition and enactment of model-driven migration processes". After two rounds of review, the paper is accepted.

9.2 Teaching

- Antoine Beugnard teaches software engineering, modeling and object programming. He teaches these courses at License and at Master level. He is the coordinator of the brittany research master (Master Recherche en Informatique de l'Université de Rennes 1) at Telecom Bretagne.

- Fabien Dagnat teaches advanced programming (programming languages, functional programming, compilation), formal modeling (for concurrency mainly) and object oriented programming to master students of Telecom Bretagne. He teaches also a formal modeling course in the core of the brittany research master.

- Julien Mallet teaches software engineering, modeling, object oriented programming and computer security at License and at Master level. He is also involved in project-based learning.

- Siegfried Rouvrais is involved in a broad range of courses relating to computer science and software engineering both at B.Sc. and M.Sc. levels, as well as for adult professionals and higher VET (e.g., Requirements, Software Architecture, Programming Languages, UML, Business Process Design and Modelling, Workflow, Enterprise Information Systems). In connection with these theoretical and practical courses, he investigates problem- and project-based learning styles aiming at favoring students' autonomy with a focus on process and reflectivity. Since 2003, as educational program designer, he has been particularly involved in integrating such models into Telecom Bretagne curricula and vocational trainings, with a clear emphasis on student's competency development, multidisciplinarity, and industry connections. Since several years, he trains faculty members from UBL and offers consultations for improving teaching and curriculum design (PjBL, Active Pedagogy, Reflectivity, Constructive Alignment, Program Design, etc.). He co-leads the

TREE research Group at IMT Atlantique (Transdisciplinary Research in Engineering Education).

- Maria-Teresa Segarra is the head of the computer science domain at Telecom Bretagne. She teaches advanced programming paradigms (component and service oriented), object-oriented programming and design and databases at License and Master levels.
- Jean-Christophe Bach teaches basic programming (C, Java), software engineering, object-oriented design and programming, advanced programming (programming languages, functional programming, compilation), formal modeling (for concurrency mainly) and logic at Telecom Bretagne. He is also involved in scientific mediation by proposing several activities to demonstrate the *algorithmic thinking* at the core of the Computer Science without requiring any computer or even electric devices. These activities are a first part of the CSIRL (Computer Science In Real Life) project (renamed InfoSansOrdi) which aims to popularize computer science and to initiate children, school students and non-scientists into this domain.

10 Bibliography

Major publications by the team in recent years

[1] A. Beugnard, A. Hassan, “A New Component Model for Highly Distributed Environements”, *in: FACS 2011 : 8th International Symposium on Formal Aspects of Component Software*, Springer (editor), 2011.

[2] A. Beugnard, J.-M. Jezéquel, N. Plouzeau, D. Watkins, “Making components contract aware”, *Computer 32*, 7, july 1999, p. 38 – 45.

[3] A. Beugnard, S. Sadou, “Method overloading and overriding cause distribution transparency and encapsulation flaws”, *Journal of Object Technology, Special Issue OOPS Track at SAC 2006 6*, 2, february 2007, p. 31 – 45.

[4] A. Beugnard, “OO languages late-binding signature”, *in: The Ninth International Workshop on Foundations of Object-Oriented languages, FOOL 9*, p. 1 – 6, 2002.

[5] J. Buisson, F. Dagnat, E. Leroux, S. Martinez, “Safe reconfiguration of Coqcots and Pycots components”, *Journal of systems and software*, 2016.

[6] J. Buisson, F. Dagnat, “ReCaml: execution state as the cornerstone of reconfigurations”, *in: ACM SIGPLAN: 15th International Conference on Functional Programming*, ACM (editor), p. 27 – 38, New York, NY, USA, 2010.

[7] V. Chiprianov, Y. Kermarrec, S. Rouvrais, J. Simonin, “Extending Enterprise Architecture Modeling Languages for Domain Specificity and Collaboration: Application to Telecommunications Service Design”, *Software and systems modeling 13*, 3, july 2014, p. 963 – 974.

[8] A. Cortier, L. Besnard, J.-P. Bodeveix, F. Dagnat, G. Garcia, M. Pantel, M. Strecker, J.-P. Talpin, A.-E. Rugina, J. Ouy, M. Filali, J. Buisson, *Synoptic: a domain-specific modeling language for space on-board application software*, *Engineering*, Springer, 2010, ch. Synthesis of embedded software, frameworks and methodologies for correctness by construction, p. 79 – 119.

[9] J.-M. Gilliot, A. P. Khac, A. Beugnard, M. T. Segarra, “L’ingénierie dirigée par les modèles pour la conception d’applications à architectures réparties adaptables”, *Technique et science informatiques 30*, 1, janvier 2011.

[10] F. R. Golra, A. Beugnard, F. Dagnat, C. Guychard, S. Guerin, “Continuous Requirements Engineering Using Model Federation”, *in: 24th IEEE International Requirements Engineering Conference (RE)*, p. 347–352, 2016.

[11] F. R. Golra, A. Beugnard, F. Dagnat, C. Guychard, S. Guerin, “Using free modeling as an agile method for developing domain specific modeling languages”, *in: MODELS 2016 : ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, p. 24–34, 2016.

[12] F. R. Golra, F. Dagnat, “Generation of Dynamic Process Models for Multi-metamodel Applications”, *in: ICSSP 2012: International Conference on Software and System Process*, 2012.

[13] J.-B. Lezoray, M. T. Segarra, A. Beugnard, J.-M. Gilliot, “Modeling Dynamic Adaptations using Augmented Feature Models”, *in: SAC 2013 : 28th Symposium On Applied Computing*, 2013.

[14] J. Mallet, S. Rouvrais, “Style-Based Model Transformation for Early Extrafunctional Analysis of Distributed Systems”, *in: QoSA ’08 : international conference on quality of software architecture, october, Karlsruhe, Germany*, S. verlag (editor), p. 55 – 70, 2008.

[15] M. Zouari, M. T. Segarra, F. André, A. Thépaut, “An Architectural Model for Building Distributed Adaptation Systems”, *in: 5th International Symposium on Intelligent Distributed Systems*, 2011.

Doctoral dissertations and “Habilitation” theses

[1] F. Dagnat, *Du génie logiciel pour déployer, gérer et reconfigurer les logiciels*, Habilitation à Diriger des Recherches, Université de Rennes I, janvier 2016.

[2] S. Martinez, *Plates-formes et mises à jour dynamiques configurables*, PhD Thesis, INFO - Dépt. Informatique (Institut Mines-Télécom-Télécom Bretagne-UEB), Mar 2016, Th. doct. : Informatique, Institut Mines-Télécom-Télécom Bretagne-UEB, mars 2016.

Articles in referred journals and book chapters

[3] J. Buisson, F. Dagnat, E. Leroux, S. Martinez, “Safe reconfiguration of Coqcots and Pycots components”, *Journal of systems and software*, 2016.

Publications in Conferences and Workshops

[4] M.-P. Adam, M. Arzel, A. Beugnard, M. Le Goff-Pronost, P. Tremenbert, D. Baux, B. Vinouze, M. Morvan, J.-P. Coupez, “Boosting advanced skills in project management thanks to complex human and technical situations”, *in: SEFI 2016 : European Society for Engineering Education annual conference*, 2016.

[5] J. Bennedsen, S. Rouvrais, “Finding Good Friends to Learn from and to Inspire”, *in: FIE 2016 : 46th IEEE Annual Frontiers in Education Conference The Crossroads of Engineering and Business*, 2016. http://fie2016.org/.

[6] M. El Hamlaoui, B. Coulette, S. Ebersold, M. Nassar, A. Beugnard, Y. Jamoussi, H. N. Tran, J.-C. Bach, A. Anwar, S. Bennani, “Alignment of viewpoint heterogeneous design models: "Emergency Department" Case Study”, *in: GEMOC workshop 2016 - International Workshop on The Globalization of Modeling Languages*, C. W. Proceedings (editor), *1731*, p. 18–27, CEUR-WS.org, 2016.

[7] F. R. Golra, A. Beugnard, F. Dagnat, C. Guychard, S. Guerin, “Addressing Modularity for Heterogeneous Multi-model Systems using Model Federation”, *in: MODULARITY 2016 : 15th International Conference on Modularity*, p. 206–211, 2016.

[8] F. R. Golra, A. Beugnard, F. Dagnat, C. Guychard, S. Guerin, “Continuous Requirements Engineering Using Model Federation”, *in: 24th IEEE International Requirements Engineering Conference (RE)*, p. 347–352, 2016.

[9] F. R. Golra, A. Beugnard, F. Dagnat, C. Guychard, S. Guerin, “Using free modeling as an agile method for developing domain specific modeling languages”, *in: MODELS 2016 : ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, p. 24–34, 2016.

[10] N. T. Huynh, M. T. Segarra, A. Beugnard, “A Development Process Based on Variability Modeling for Building Adaptive Software Architectures”, *in: FedCSIS 2016 : 36th IEEE Software Engineering Workshop, Federated Conference On Computer Science And Information Systems*, 2016.

[11] N. T. Huynh, M. T. Segarra, A. Beugnard, “Ensuring consistent dynamic adaptation: an approach from design to runtime”, *in: AICCSA 2016 : 13th ACS/IEEE International Conference on Computer Systems and Applications*, 2016.

[12] T. Lucke, L. Brodie, I. Brodie, S. Rouvrais, “Is it possible to adapt CDIO for distance and online education?”, *in: CDIO 2016 : 12th International Conference: Enhancing Innovation Competencies through advances in engineering education*, *45 - Research Reports (Turku University of Applied Sciences)*, Turku University of Applied Sciences, 2016.

[13] S. Rouvrais, H. Andunsson, I. Soemundsdottir, C. Lassudrie, G. Landrac, “Pairwise Collaborative Quality Enhancement: Experience of Two Engineering Programmes in Iceland and France”, *in: CDIO 2016 : 12th International Conference: Enhancing Innovation Competencies through advances in engineering education*, T. U. of Applied Sciences (editor), *45 - Research Reports (Turku University of Applied Sciences)*, p. 186–195, 2016.

[14] S. Rouvrais, S. Gaultier Le Bris, “Breadth Experiential Courses to Meet Programme Outcomes for Engineers”, *in: WEEF & GEDC 2016 : 6th World Engineering Education Forum & The Global Engineering Deans Council*, Springer (editor), 2016.

[15] B. Sultan, F. Dagnat, C. Fontaine, “Maîtrise des Correctifs de Sécurité pour les Systèmes Navals”, *in: CIEL 2016 : 5ème Conférence en Ingénierie du Logiciel*, p. 1–6, 2016.