Activity Report 2011

Section Software

Edition: 2012-03-22
DISTRIBUTED SYSTEMS AND SERVICES

1. ACES Project-Team (section vide) ............................................................. 5
2. ADAM Project-Team ........................................................................ 6
3. AMAZONES Team ........................................................................ 9
4. ARLES Project-Team .................................................................. 12
5. ASAP Project-Team .................................................................. 17
6. ASCOLA Project-Team ................................................................. 19
7. ATLAMMOD Team .................................................................. 22
8. CIDRE Project-Team ................................................................. 28
9. FOCUS Project-Team ................................................................ 29
10. INDES Project-Team ................................................................ 30
11. MYRIADS Team .................................................................... 33
12. OASIS Project-Team ................................................................ 37
13. PHOENIX Project-Team .............................................................. 39
14. POPS Project-Team .................................................................. 44
15. REGAL Project-Team .................................................................. 47
16. RMOD Project-Team .................................................................. 49
17. SARDES Project-Team .............................................................. 51
18. SCORE Team ........................................................................ 54
19. TRISKELL Project-Team .............................................................. 55

DISTRIBUTED AND HIGH PERFORMANCE COMPUTING

20. ALGORILLE Project-Team ................................................................. 57
21. CEPAGE Project-Team .................................................................. 60
22. GRAAL Project-Team .................................................................. 62
23. GRAND-LARGE Project-Team .......................................................... 65
24. HIEPACS Project-Team .................................................................. 70
25. KERDATA Team ........................................................................ 74
26. MESCAL Project-Team .................................................................. 76
27. MOAIS Project-Team .................................................................. 78
28. RUNTIME Project-Team ............................................................... 80

NETWORKS AND TELECOMMUNICATIONS

29. DIONYSOS Project-Team ................................................................. 85
30. DISTRIBCOM Project-Team ............................................................. 86
31. DNET Team ........................................................................ 87
32. GANG Project-Team (section vide) ................................................... 88
33. HIPERCOM Project-Team .............................................................. 89
34. MADYNES Project-Team ............................................................... 90
35. MAESTRO Project-Team (section vide) ........................................... 92
36. MASCOTTE Project-Team ............................................................. 93
37. PLANETE Project-Team ............................................................... 98
38. RAP Project-Team (section vide) ................................................... 104
39. RESO Project-Team ................................................................. 105
40. SWING Team ................................................................. 106
41. TREC Project-Team ................................................................. 107
ACES Project-Team (section vide)
5. Software

5.1. Introduction
We intend to develop a number of software to evaluate and validate our solutions. We will complete our
development by experimentation, benchmarks and deployment in multi-paradigm platforms. We list our actual
software that we intend to continue and to extend in the ADAM project-team.

5.2. CALICO
Participants: Laurence Duchien, Antonio de Almeida Souza Neto, Anne-Françoise Le Meur.
Modern software is characterized by a need for constant and rapid evolution, such as in the mobile domain.
To facilitate the development and the rapid evolution of complex systems, software engineering approaches
have been proposed, such as software architecture and agile software development. However, current solutions
offer poor support to enable the development of a reliable system.
In this context we propose CALICO, an agile development framework for the design and evolution of safe
compontent-based and service-oriented software. The agile software development relies on an iterative and
incremental development cycle that allows the architect to iterate between the design of the architecture and
the debug of the software in its execution context. At each iteration, the architect can evolve its software
and check the consistency of its evolution through the execution of static and dynamic analysis tools. Thus,
during the design and the evolution of the system, the architect can use a set of metamodels to specify the
structure of the architecture and its various quality of services requirement. During the deployment, CALICO
instantiates the system on the target runtime platform from the models specified and keeps them synchronized
with the software during its execution. Through this means, the architect has a conceptual view which allows
him to reason on the critical software properties during its evolution. Moreover, in order to check these
evolutions, CALICO provides a unifying framework which allows reuse of many static analysis tools of
software architectures and dynamic debugging tools, that were scattered in different existing platforms. Thus,
each change can be statically analyzed on the conceptual view before being propagated to the software system.
Dynamic analysis are based on data values available during the execution only. The capture of these values is
done through automatic instrumentation of the software system.
Globally, CALICO enables reliable evolution even if the underlying platforms does not natively provide this
support. The current version handles four component-based and service-oriented platforms. Moreover, the
benchmarks that we have performed show that CALICO is usable for the design and development of safe
applications up to 10,000 components and services, which corresponds to the maximal load of most runtime
platforms. CALICO has been developed in the context of Guillaume Waignier’s PhD thesis [79].
CALICO is an open source software available at http://calico.gforge.inria.fr.

5.3. Fractal
Participants: Christophe Demarey, Philippe Merle [correspondant], Romain Rouvoy, Lionel Seinturier.
FRACTAL is a modular, extensible and programming language agnostic component model that can be used
to design, implement, deploy and reconfigure systems and applications, from operating systems, middleware
platforms to graphical user interfaces [53], [55], [67]. FRACTAL has been designed by both Inria and
France Telecom R&D.
FRACTAL is also a LGPL open source software project hosted by the OW2 international consortium and is
available at http://fractal.ow2.org [59].
Philippe Merle is the leader of the OW2 FRACTAL open source project. The ADAM project-team actively contributes to this project, and more specifically on the following modules:

- **AOKell** is an aspect-oriented implementation of the FRACTAL component model [76].
- **Fraclet** is an attribute-oriented programming model enabling the rapid development of FRACTAL components [73].
- **Fractal ADL** is the extensible architecture definition language for FRACTAL associated to an open FRACTAL component-based toolchain.
- **Fractal Distribution** is the module to produce packaged releases of the FRACTAL project.
- **Fractal Documentation** is the module to produce the whole documentation of the FRACTAL project.
- **Fractal Eclipse Plugin** is a plugin to create FRACTAL projects within the Eclipse IDE [57], [58].
- **Fractal Explorer** is a framework to build graphical consoles to introspect and manage FRACTAL components dynamically at runtime.
- **FScript** is a scripting language for both introspection and reconfiguration of FRACTAL software systems.
- **Juliac** is an extensible framework for generating and compiling the code of FRACTAL component-based systems. Juliac is registered with the APP (Agence pour la Protection des Programmes) under reference FR.001.230007.000.S.P.2009.000.10600.
- **Koch** is an implementation of the FRACTAL component model where components have a component-based control membrane.

### 5.4. FraSCAti

**Participants:** Christophe Demarey, Damien Fournier, Rémi Mélisson, Philippe Merle [correspondant], Christophe Munilla, Romain Rouvoy, Lionel Seinturier.

FraSCAti is a runtime platform for the Service Component Architecture (SCA) component framework. SCA is an initiative for unifying Service Oriented Architectures (SOA) and Component-Based Software Engineering (CBSE). SCA is supported by the Open SOA consortium, which includes partners, such as IBM, Oracle, Sun and Iona, and is standardized by the OASIS consortium (see at http://www.oasis-open.org/SCA).

FraSCAti includes Tinfi, which provides a SCA personality for the FRACTAL component model. Thanks to the openness of this latter model, the necessary code elements (so called *controllers* and *membranes*) have been designed and developed to customize FRACTAL and to end up with components owning both a FRACTAL personality and a SCA personality. As far as we know, this result, which has been presented in [75], is original and is the first one to concretely demonstrates that FRACTAL is open and flexible enough to implement different component personalities. Moreover, Tinfi reuses the aspect-oriented concepts defined in FAC [70] for component-based programming and allows integrating smoothly non functional concerns (so called intents and policy sets in SCA terms). FraSCAti and Tinfi have been implemented by reusing modules developed in the context of the FRACTAL project, and among others, the Juliac FRACTAL compiler.

The development of the FraSCAti platform is conducted in the context of some current and past funded projects (ICT FP7 SOA4All Integrated Project, ANR ARPEGE ITEmIS project, FUI EasySOA project, Inria ADT Adapt).

FraSCAti is a LGPL open source software, hosted by the OW2 consortium since November 2008 at http://frascati.ow2.org. FraSCAti is registered with the APP (Agence pour la Protection des Programmes) under reference FR.001.050017.000.S.P.2010.000.10000.

### 5.5. SPACES

**Participants:** Russel Nzekwa, Daniel Romero [correspondant], Romain Rouvoy, Lionel Seinturier.
SPACES is a context mediation middleware that follows the **REpresentational State Transfer** (REST) principles [61]. The current implementation of SPACES is based on the COSMOS context framework [54], [72] and the **COMANCHE** web server [53]. Both COSMOS and **COMANCHE** are based on the **FRACTAL** component model and use the **JULIA** implementation of the **FRACTAL** runtime environment [53].

The main features of the current SPACES implementation are presented below:

1. **Ubiquitous connectors**: SPACES defines connectors that encapsulate the distribution concern. These connectors expose the COSMOS context nodes as REST resources with logical associated URLs, and enable interactions between consumers and producers via different communication protocols and the discovery of the available context sources. The current SPACES implementation supports interaction using the HTTP and twitter [65] protocols. For discovery, the implementation uses the Service Location Protocol (SLP) [63].

2. **Context Representation**: Following the REST principles, SPACES supports multiple representations of the context information: **JSON** [56], **XML** and **Java serialization**.

3. **Quality of context (QoC) information**: The QoC properties are incorporated as service attributes in the SLP advertisements of the context information.

4. **Context selection**: The restrictions in terms of QoC of the required context information are expressed as LDAP filters [77]. SPACES benefits from the LDAP based queries of SLP to select the context providers.

We use **XStream 1.3.13** [51] and **JSON-lib 2.2.34** [50] to serialize context information as XML and JSON documents. For SLP and twitter we employ **jSLP 1.0.0** [71] and **twitter-4j 2.0.6** [80].

SPACES is registered with the **APP** (Agence pour la Protection des Programmes) under reference **IDDN 10-500002-000**.

5.6. **ApplIDE**

**Participants**: Laurence Duchien, Christophe Demarey, Clément Quinton [correspondant].

ApplIDE is directly connected to the work of Carlos Parra’s PhD and **Ubino** ADT’s work which covers the definition and implementation of a Context-Aware Dynamic Software Product Line (DSPL) named **CAPucine**. It provides a set of tools for selection of features, metamodel transformation and code generation for mobile applications [40]. The current implementation of ApplIDE addresses transformation from **CAPucine** metamodel towards **SCA** metamodel, and Spoon **EMF** metamodel. The transformations were formerly written with Acceleo tool, which is a dedicated language for transformation, enhancing the readability. **ApplIDE** meta models are based on the Eclipse Modeling Framework. Code generators are all written in Acceleo.

ApplIDE is registered with the **APP** (Agence pour la Protection des Programmes) under reference **IDDN.FR.001.500004.000.S.A.2010.000.10600**.
5. Software

5.1. Logos

**Participants:** Julien Ponge, Stéphane Frénot.

Logos is a development project linked to the LISE ANR grant. Its goal is to generate execution logs from OSGi services interactions. The main idea is to intercept every service call and generate an entry in a log file. The log file system should be used in the LISE context which is related to legal issues. Generated Logos logs should be: Complete, encoded with a cryptographic algorithm, compact and immutable.

The software is currently used as a Amazones internal test suite. It is fully tested on standard OSGi architectures.

5.2. Logminer

**Participants:** Julien Ponge, Stéphane Frénot.

LogMiner is a toolbox, written in Scala in current development. The LogMiner framework takes Logos inputs and generates service usage automata. The goal of logminer is to represent application activity in a synthetic way in order to identify behavioral changes while updating the system. When one updates its applications on its environment, the logminer framework enables observation and identifies variations in service usages.

The software is currently under development it integrates a automata generator and a data visualisations modules.

5.3. Eimc

**Participants:** Zheng Hu, Stéphane Frénot, Bernard Tourancheau [Projet Swing].

Eimc is an architecture for managing sensor dedicated to legacy equipment management. The project aims at designing a dynamic framework that integrates sensors from the surrounding environment and detects new equipments from their physical behavior. For instance, a fridge vibrates when the compressor is working. The frequency of vibrations distinguishes a fridge from washing machine. The framework designs a Complex Based Event processing architecture where we need to focus on the number of manageable equipments, the number of deployed sensors and the number of physical measurements that can be handled.

The project is a joint project with Orange Labs, and a PhD student Zheng Hu. He is co-directed by Stéphane Frénot and Bernard Tourancheau from Amazones and Swing teams.

5.4. Aoraï

**Participant:** Nicolas Stouls.

Developed at CEA-LIST, Frama-C is an extensible and collaborative platform dedicated to source-code analysis of C software. The Aoraï [49] plug-in for Frama-C [31] provides a method to automatically annotate a C program according to a behavioral property P such that, if the annotations are verified, then we ensure that the program respects P.

The computation process is divided into two steps: the specification generation from the property and the constraints propagation for static simplification. According to the classical invariant verification granularity, observable states of a program correspond to each call or return statements of an operation. Each state of the program is associated to a set of transitions in an internal representation of the property, managed as a Büchi automata. Starting from a super-set of authorized behaviors, some static simplifications can be done in order to generate sufficient pre/post-conditions on each operation.
The classical method to validate generated annotations is to use the Jessie plug-in and the Why tool, using theorem provers.

A new research report [50] has been published and some new developments have been done in order to increase consequently the efficiency of the tool.

5.5. STOP

**Participants:** François Goichon, Stéphane Frénot, Pierre Parrend.

STOP is a security-oriented program analysis toolkit developed by François Goichon as part of his masters thesis. He was supervised by Stéphane Frénot and Pierre Parrend from FZI, Karlsruhe.

The tool implements a novel static analysis technique called Service-oriented Tainted Object Propagation, described in more detail in the Results section.

5.6. IzPack

**Participant:** Julien Ponge.

IZPack [47] is a software installer creation framework for the Java platform. Its main differentiator with respect to the other installation solutions is that it generates cross-platform installers that can adapt themselves to the underlying operating system so as to still provide tight integration. It was also designed to be highly customizable and extensible.

IZPack is nearing its 10 years landmark. It is hosted at the Coddehaus [32] Foundation and released under the terms of the Apache Software License version 2.0. Its users community non-exhaustively comprise SpringSource, JBoss / RedHat, Oracle / Sun Microsystems, the Scala language, XWiki, Terracotta or Silverpeas.

The project was originally created by now INRIA Amazones team member Julien Ponge, who still leads the project. In 2010, it was presented at the Devoxx conference.

5.7. WSNet

**Participants:** Guillaume Chelius [INRIA D-NET Team, project leader], Antoine Fraboulet, Loïc Lemaître [INRIA SensTools IJD].

WSNet is a modular wireless network simulator. It incorporates the following aspects: (i) accurate simulation of the radio channel: Supports MIMO, multi-interface, multi-channel, etc. (ii) simulation environment: simulation of the interaction between sensors and their environment: measurement and control, simulation of device power consumption. Furthermore, WSNet can be interfaced with the WSim sensor node emulator to form a distributed emulation of a sensor network.

WSNet source code is registered at the Agency For The Protection Of Programs (APP IDDN 06-370013-000). Licence: CeCILL (2). See also the web page http://wsnet.gforge.inria.fr/.

5.8. WSim

**Participants:** Guillaume Chelius [INRIA D-NET Team], Antoine Fraboulet [Project leader], Loïc Lemaître [INRIA SensTools IJD], Julien Carpentier [INRIA ORSI IJD].

WSIM is a platform emulator for embedded systems allowing performance evaluation and programming assistance during the application design stages of distributed wireless sensor networks. WSIM is a simulation tool enabling a rapid and relevant feedback on features and quality of embedded software in constrained systems. Its simulation model allows to interface with other tools like WSNet to build complex simulation environments.

WSim source code is registered at the Agency For The Protection Of Programs (APP IDDN 06-370012-000). Licence: CeCILL (2). See also the web page http://wsim.gforge.inria.fr/.
5.9. Esimu

**Participant:** Antoine Fraboulet.

eSimu is a complete system energy model based on non-intrusive measurements. This model aims at being integrated in fast cycle accurate simulation tools to give energy consumption feedback for embedded systems software programming. Estimations take into account the whole system consumption including peripherals. Experiments on a complex ARM9 platform show that our model estimates are in error by less than 10% from real system consumption, which is precise enough for source code application design, while simulation speed remains fast. eSimu can be used as a standalone tool or in conjunction with WSim.

Licence: CeCILL (2). See also the web page [http://esimu.gforge.inria.fr/](http://esimu.gforge.inria.fr/).

5.10. ABR

**Participants:** Frédéric Le Mouël, Stéphane Frénot.

The Ambient Bundle Repository (ABR) is an OSGi extension, compliant with the Bundle Repository API. Instead of proposing a centralized discovery as the default bundle repository implementation, ABR abstracts different discovery protocols (UPnP, ...) and publishes/subscribes a local repository containing bundles in a device geographically-close environment. ABR implements mobility models to track mobile devices, to warn the user deploying bundles of the remaining presence time of bundles and to anticipate a possible bundle deployment non-ending.

5.11. AxSeL

**Participants:** Amira Ben Hamida, Frédéric Le Mouël, Stéphane Frénot.

While installing and executing applications on mobile devices, the issue of the limit of resources is quickly encountered.

AxSeL (A conteXtual Service Loader) is an OSGi prototype extension that modifies the bundle loading at deployment time for a context-aware service loading at run time. The approach is based on a service graph colouring process. We represent an application as a bi-dimensional dynamic graph with services and bundles dependencies. The colouring decision provides an optimal deployment configuration of the application in a given context. Context listening mechanisms capture changes and propagate recolouring and redeployment processes.

Context elements currently implemented and monitored are the hardware memory and disk sizes. Application currently implemented and tested is a service-oriented PDF viewer that is adapting its display to available device resources [6].

This prototype is a part of the PhD thesis of Amira Ben Hamida [29].

5.12. QuestMonitor

**Participants:** Stéphane Grumbach, Ahmad Ahmad-Kassem, Fuda Ma.

QuestMonitor [28] is a visualization tool that allows to visualize dynamic networks, and monitor the execution of protocols written in the data centric language Netlog. The language allows to specify protocols which in sometimes their behavior, in dynamic networks, are tricky to understand. QuestMonitor allows to monitor all the communication between the nodes, the evolution of the data stores on each node, as well as the execution of the declarative code. It also allows to color the virtual data structures, such as routes, backbones, etc. Together with the code editing facility, it constitutes a good tool for rapid prototyping.

Amazones team aimed at bridging the gap between "high-level" developed architectures that we called the *northBound* and "low-level" run-time, the *southBound*. Northbound architecture rely on virtual machines and advanced development languages, whereas southbound architectures rely on micro-kernels and drivers development. Our results are mainly initial studies since we fixed our research team on November 2010, and the end notification arrived on June 2011. Although our time frame was short we managed to gain knowledge in three areas linked to Amazones goals.
5. Software

5.1. Introduction

In order to validate our research results, our research activities encompass the development of related prototypes as surveyed below.

5.2. Emergent Middleware Enablers

Participant: Valérie Issarny [correspondent].

As part of our research work on Emergent Middleware, we have implemented Enablers (or Enabler functionalities) that make part of the overall CONNECT [30] architecture realizing Emergent Middleware in practice.

Discovery Enabler: The CONNECT Discovery Enabler is the component of the overall CONNECT architecture that handles discovery of networked systems (NSS), stores their descriptions (NSS models), and performs an initial phase of matchmaking to determine which pairs of systems are likely to be able to interoperate. Such pairs are then passed to the Synthesis Enabler so that mediators can be generated. The Discovery Enabler is written in Java and implements several legacy discovery protocols including DPWS and UPnP. Matchmaking is done on the basis of affordances contained in the system description, that is, ontological concepts describing the system’s category. Systems with the same affordance, or affordances standing in a specialization relationship, can be considered for connection. If a system does not provide its affordance, the Discovery Enabler can infer a likely one using text categorization based on the system’s interface description. The Discovery Enabler will soon be available for download under an open source license.

Synthesis Enabler: We have implemented (in Java) two approaches to mediator synthesis as part of the CONNECT Synthesis Enabler:

- **Mapping-based mediator synthesis.** This implementation focuses on networked systems that have compatible functionalities but are unable to interact successfully due to mismatching interfaces and/or behaviors. The ontology used in our implementation is encoded so as to make the reasoning more efficient at runtime while considering both subsumption and the union of classes. Based on the interface mapping, a correct-by-construction mediator is generated. In our current implementation, we are leveraging the LTSA (Labeled Transition System Analyser) model checker to generate the parallel composition of the mapping processes and verify that the overall system successfully terminates. In the near future, we will be incorporating our techniques so as to deal with ambiguous mappings, i.e., when an action from one system may semantically be mapped to different actions from the other system.

- **Goal-based abstract mediator synthesis.** This implementation considers the protocols of two networked systems and produces the mediator protocol that allow them to interact so as to satisfy user goals. More specifically, the alphabet of the two protocols are aligned using ontology matching. The aligned protocols as well as the user goal are encoded as a satisfiability problem. The Zot model checker solves this problem (if possible) and produces a possible feasible interaction satisfying user goals.

The synthesis enabler will soon be available for download under an open source license.

5.3. Service-oriented Middleware for Pervasive Computing

Participants: Nikolaos Georgantas [correspondent], Valérie Issarny [correspondent].
In the past years, we have built a strong foundation of service-oriented middleware to support the pervasive computing vision. This specifically takes the form of a family of middlewares, all of which have been released under the open source LGPL license:

- **WSAMI - A Middleware Based on Web Services for Ambient Intelligence**: WSAMI (Web Services for AMbient Intelligence) is based on the Web services architecture and allows for the deployment of services on wireless handheld devices like smartphones and PDAs. URL: [http://www-rocq.inria.fr/arles/download/ozone/index.htm](http://www-rocq.inria.fr/arles/download/ozone/index.htm)

- **Ariadne - A Protocol for Scalable Service Discovery in MANETs**: Ariadne enriches WSAMI with the Ariadne service discovery protocol, which has been designed to support decentralized Web service discovery in multi-hop mobile ad hoc networks (MANETs). Ariadne enables small and resource-constrained mobile devices to seek and find complementary, possibly mobile, Web services needed to complete specified tasks in MANETs, while minimizing the traffic generated and tolerating intermittent connectivity. URL: [http://www-rocq.inria.fr/arles/download/ariadne/index.html](http://www-rocq.inria.fr/arles/download/ariadne/index.html)

- **MUSDAC - A Middleware for Service Discovery and Access in Pervasive Networks**: The Multi-protocol Service Discovery and ACcess (MUSDAC) middleware platform enriches WSAMI so as to enable the discovery and access to services in the pervasive environment, which is viewed as a loose and dynamic composition of independent networks. MUSDAC manages the efficient dissemination of discovery requests between the different networks and relies on specific plug-ins to interact with the various middleware used by the networked services. URL: [http://www-rocq.inria.fr/arles/download/ubisec/index.html](http://www-rocq.inria.fr/arles/download/ubisec/index.html)

- **INMIDIO - An Interoperable Middleware for Ambient Intelligence**: INMIDIO (INteroperable MIddleware for service Discovery and service InteractiOn) dynamically resolves middleware mismatch. More particularly, INMIDIO identifies the interaction middleware and also the discovery protocols that execute on the network and translates the incoming/outgoing messages of one protocol into messages of another, target protocol. URL: [http://www-rocq.inria.fr/arles/download/inmidio/index.html](http://www-rocq.inria.fr/arles/download/inmidio/index.html)

- **COCOA - A Semantic Service Middleware**: COCOA is a comprehensive approach to semantic service description, discovery, composition, adaptation and execution, which enables the integration of heterogeneous services of the pervasive environment into complex user tasks based on their abstract specification. Using COCOA, abstract user tasks are realized by dynamically composing the capabilities of services that are currently available in the environment. URL: [http://gforge.inria.fr/projects/amigo/](http://gforge.inria.fr/projects/amigo/)

- **ubiSOAP - A Service Oriented Middleware for Seamless Networking**: ubiSOAP brings multi-radio, multi-network connectivity to services through a comprehensive layered architecture: (i) the multi-radio device management and networking layers together abstract multi-radio connectivity, selecting the optimal communication link to/from nodes, according to quality parameters; (ii) the communication layer allows for SOAP-based point-to-point and group-based interactions in the pervasive network; and (iii) the middleware services layer brings advanced distributed resource management functionalities customized for the pervasive networking environment. URL: [http://www.ist-plastic.org](http://www.ist-plastic.org)

### 5.4. Supporting Service Orchestrations over Heterogeneous Interaction Paradigms

**Participant**: Nikolaos Georgantas [correspondent].
Established architectural paradigms enabling open system integration, such as service oriented architecture (SOA) and enterprise service bus (ESB), have provided answers to the essential issue of interoperability in distributed systems. However, realizations of these architectural paradigms fall short when it comes to integrating systems featuring heterogeneous interaction paradigms, such as client/server (CS), publish/subscribe (PS) and tuple space (TS), due to the differing interaction semantics of the latter. Typical solutions constitute in wrapping any system behind RPC-based service interfaces, which results in partial loss of interaction semantics. This can cause suboptimal or even problematic system integration.

Aiming at enabling seamless integration of heterogeneous interaction paradigms, we introduce an interoperability solution based on abstraction and merging of their common high-level semantics, paying special attention to the preservation of semantics. To this end, we propose three abstract connector types for the CS, PS and TS interaction paradigms. We further introduce a higher-level generic application (GA) connector type, which provides an abstract union of the three models, thus preserving their interaction semantics. We express our connector types in terms of application programming interface (API) primitives and related interaction protocol semantics. We then apply our abstractions to rethink a typical SOA- and ESB-based orchestration of heterogeneous distributed systems. Our solution features:

- Extending the BPEL workflow language with GA API primitives in terms of extension activities enabled by the BPEL specification;
- Introducing XSLT transformation between the GA-extended BPEL and the standard BPEL, which consists of encapsulating GA primitives into standard BPEL primitives and enables conveying GA semantics on top of BPEL primitives and subsequently on top of the common bus protocol primitives;
- Providing Java code templates for systematic and highly facilitated building of ESB-embedded binding components;
- Proposing interface description languages in the form of XSDs for systems employing CS-, PS-, TS-, and GA-connectors; and
- Introducing XSLT transformations between native system interface descriptions and GA-based interface descriptions.

We have developed our solution on top of the PEtALS ESB, which provides inherent support for BPEL by embedding the EasyBPEL workflow engine. Our solution considerably facilitates the application developer in designing and executing heterogeneous orchestrations. Furthermore, it is highly extensible, enabling easy integration of support for new middleware platforms. To demonstrate the applicability of our approach, we have implemented an application workflow integrating a JMEDS DPWS Web Service (CS), a JMS system based on Apache ActiveMQ (PS), and a Jini JavaSpaces system (TS). Our software will soon be released under open source license.

5.5. Srijan: Data-driven Macroprogramming for Sensor Networks

Participant: Animesh Pathak [correspondent].

Macroprogramming is an application development technique for wireless sensor networks (WSNs) where the developer specifies the behavior of the system, as opposed to that of the constituent nodes. As part of our work in this domain, we are working on Srijan, a toolkit that enables application development for WSNs in a graphical manner using data-driven macroprogramming.

It can be used in various stages of application development, viz.:

1. Specification of application as a task graph,
2. Customization of the auto-generated source files with domain-specific imperative code,
3. Specification of the target system structure,
4. Compilation of the macroprogram into individual customized runtimes for each constituent node of the target system, and finally
5. Deployment of the auto generated node-level code in an over-the-air manner to the nodes in the target system.
The current implementation of Srijan targets both the Sun SPOT sensor nodes and larger nodes with J2SE. Most recently, Srijan also includes rudimentary support for incorporating Web services in the application being designed. The software is released under open source license, and available as an Eclipse plug-in at http://code.google.com/p/srijan-toolkit/.

5.6. Yarta: Middleware for supporting Mobile Social Applications

Participant: Animesh Pathak [correspondent].

With the increased prevalence of advanced mobile devices (the so-called “smart” phones), interest has grown in Mobile Social Ecosystems (MSEs), where users not only access traditional Web-based social networks using their mobile devices, but are also able to use the context information provided by these devices to further enrich their interactions. We are developing a middleware framework for managing mobile social ecosystems, having a multi-layer middleware architecture consisting of modules, which will provide the needed functionalities, including:

- Extraction of social ties from context (both physical and virtual),
- Enforcement of access control to protect social data from arbitrary access,
- A rich set of MSE management functionalities, using which mobile social applications can be developed.

Our middleware adopts a graph-based model for representing social data, where nodes and arcs describe socially relevant entities and their connections. In particular, we exploit the Resource Description Framework (RDF), a basic Semantic Web standard language that allows representing and reasoning about social vocabulary, and creating an interconnected graph of socially relevant information from different sources.

The current implementation of the Yarta middleware targets both desktop/laptop nodes running Java 2 SE, as well as Android smart phones. The software is released under open source license at https://gforge.inria.fr/projects/yarta/.

5.7. iBICOOP: Mobile Data Management in Multi-* Networks

Participant: Valérie Issarny [correspondent].

Building on the lessons learned with the development of pervasive service oriented middleware and of applications using them, we have been developing the custom iBICOOP middleware. iBICOOP specifically aims at assisting the development of advanced mobile, collaborative application services by supporting interactions between mobile users. Target application services in particular include the U-EVENT suite of services for professional events.

Briefly, the iBICOOP middleware addresses the challenges of easily accessing content stored on mobile devices, and consistent data access across multiple mobile devices by targeting both fixed and mobile devices, leveraging their characteristics (e.g., always on and unlimited storage for home/enterprise servers, ad hoc communication link between mobile devices), and by leveraging the capabilities of all available networks (e.g., ad hoc networks, Internet, Telecoms infrastructure networks). It also relies on Web and Telecoms standards to promote interoperability.

The base architecture of the iBICOOP middleware consists of core modules on top of which we can develop applications that may arise in the up-coming multi-device, multi-user world:

- The Communication Manager provides mechanisms to communicate over different available network interfaces of a device — Bluetooth, WiFi, Cellular — and also using different technologies e.g., Web services, HTTP/TCP sockets, ad hoc mode.
- The Security Manager uses well-established techniques of cryptography and secure communication to provide necessary security.
- The Partnership Manager provides device or user information in the form of profiles.
• iBICOOP relies on service location protocols for naming and discovery of nearby services on currently active network interfaces that support IP multicast.
• Besides normal file managing tasks, the Local File Manager gives the user clear cues to the files that have been replicated across multiple devices or shared among different users by using different icons.

The iBICOOP middleware has been licensed by AMBIENTIC (http://www.ambientic.com/), a start-up that specifically develops innovative mobile distributed services on top of the iBICOOP middleware that allows for seamless interaction and content sharing in today’s multi-* networks.

5. Software

5.1. WhatsUp: A Distributed News Recommender

Participants: Antoine Boutet, Davide Frey, Arnaud Jegou, Anne-Marie Kermarrec.
Contact: Antoine Boutet
Licence: Open Source
Presentation: A Distributed News Recommender
Status: Beta version

This work has lead to the development of WhatsUp, a distributed recommendation system aimed to distribute instant news in a large scale dynamic system. WhatsUp has two parts, an embedded application server in order to exchange with others peers in the system and a fully dynamic web interface for displaying news and collecting opinions about what the user reads. Underlying this web-based application lies Beep, a biased epidemic dissemination protocol that delivers news to interested users in a fast manner while limiting spam. Beep is parametrized on the fly to manage the orientation and the amplification of news dissemination. Every user forwards the news of interest to a randomly selected set of users with a preference towards those that have similar interests (orientation). The notion of interest does not rely on any explicit social network or subscription scheme, but rather on an implicit and dynamic overlay capturing the commonalities between users with respect to they are interested in. The size of the set of users to which a news is forwarded depends on the interest of the news (amplification). A centralized version of WhatsUp is already up and running and the decentralized one is still in beta version.

5.2. GossipLib: effective development of gossip-based applications

Participants: Davide Frey, Heverson Ribeiro, Anne-Marie Kermarrec.
Contact: Davide Frey
Licence: Open Source
Presentation: Library for Gossip protocols
Status: released version 0.7alpha

GossipLib is a library consisting of a set of JAVA classes aimed to facilitate the development of gossip-based application in a large-scale setting. It provides developers with a set of support classes that constitute a solid starting point for building any gossip-based application. GossipLib is designed to facilitate code reuse and testing of distributed application and as thus also provides the implementation of a number of standard gossip protocols that may be used out of the box or extended to build more complex protocols and applications. These include for example the peer-sampling protocols for overlay management.

GossipLib also provides facility for the configuration and deployment of applications as final-product but also as research prototype in environments like PlanetLab, clusters, network emulators, and even as event-based simulation. The code developed with GossipLib can be run both as a real application and in simulation simply by changing one line in a configuration file.

5.3. YALPS

Participants: Davide Frey, Heverson Ribeiro, Anne-Marie Kermarrec.
Contact: Davide Frey
Licence: Open Source
Presentation: Library for Gossip protocols
Status: released version 0.3alpha
YALPS is an open-source Java library designed to facilitate the development, deployment, and testing of distributed applications. Applications written using YALPS can be run both in simulation and in real-world mode without changing a line of code or even recompiling the sources. A simple change in a configuration file will load the application in the proper environment. A number of features make YALPS useful both for the design and evaluation of research prototypes and for the development of applications to be released to the public. Specifically, YALPS makes it possible to run the same application as a simulation or in a real deployment without a single change in the code. Applications communicate by means of application-defined messages which are then routed either through UDP/TCP or through YALPS’s simulation infrastructure. In both cases, YALPS’s communication layer offers features for testing and evaluating distributed protocols and applications. Communication channels can be tuned to incorporate message losses or to constrain their outgoing bandwidth. Finally, YALPS includes facilities to support operation in the presence of NATs and firewalls using relaying and NAT-traversal techniques.

The work has been done in collaboration with Maxime Monod (EPFL).

5.4. HEAP: Heterogeneity-aware gossip protocol.

Participants: Davide Frey, Arnaud Jegou, Anne-Marie Kermarrec.
Contact: Davide Frey
Licence: Open Source
Presentation: Java Application
Status: release & ongoing development

This work has been done in collaboration with Vivien Quéma (CNRS Grenoble), Maxime Monod and Rachid Guerraoui (EPFL), and has lead to the development of a video streaming platform based on HEAP, HEterogeneity-Aware gossip Protocol. The platform is particularly suited for environments characterized by heterogeneous bandwidth capabilities such as those comprising ADSL edge nodes. HEAP is, in fact, able to dynamically leverage the most capable nodes and increase their contribution to the protocol, while decreasing by the same proportion that of less capable nodes. During the last few months, we have integrated HEAP with the ability to dynamically measure the available bandwidth of nodes, thereby making it independent of the input of the user.
ASCOLA Project-Team

5. Software

5.1. AWED

**Participants:** Mario Südholt [correspondent], Ismael Mejia.

The model of Aspects With Explicit Distribution (AWED) supports the modularization of crosscutting functionalities of distributed applications. It addresses the problem that common aspect systems do not provide features for distributed programming. It notably features three main aspect abstractions: remote pointcuts, remotely-executed advice, and distributed aspects.

This year a gray-box model for distributed composition has been built and implemented based on the AWED model using the notion of invasive distributed patterns (see Sec. 6.1 ). Furthermore, the resulting model has been applied to the evolution of grid applications and OpenMRS, an open-source health information system. The AWED system has also been employed in the CESSA project proposal (see Sec. 8.1 ) as a basis for our work on the secure evolution of service-oriented architectures. Finally, the development of a new, more modular, implementation of the AWED system has started in 2011.

AWED is available at [http://awed.gforge.inria.fr](http://awed.gforge.inria.fr).

5.2. ECaesarJ, EJava and EScala

**Participants:** Jacques Noyé [correspondent], Angel Núñez, Jurgen Van Ham.

ECaesarJ is a language developed in the context of the European project AMPLE, as joint work with the Technische Universität Darmstadt (TUD). The basic objective was to provide support for directly mapping the high-level features defined by a software product line onto implementation-level features, beyond standard feature-oriented programming. But the language has much wider applications. ECaesarJ can actually be seen as a language which smoothly integrates Object-Oriented Programming, Feature-Oriented Programming, Aspect-Oriented Programming, and Event-based Programming.

It is an extension of Java with virtual classes and propagating mixin composition (as its ancestor CaesarJ, developed at TUD), but also declarative events and state machines. Unlike AspectJ, ECaesarJ does not include a class-like concept of aspect. Instead, it deals with pointcuts and pieces of advice as (implicit) events and event handlers, which are standard class members. This makes it possible to use standard inheritance to reuse and refine them. Explicit events can also be used when events must be explicitly triggered as in traditional event-based programming. Finally, in the same way as pointcuts can be composed using logical operators, declarative events can be defined as a composition of other events.

This provides a symmetric version of AOP where virtual classes can be used to deal with structural aspects whereas events can be used to deal with behavioral aspects.

In ECaesarJ, a class can also include, as class members, state transitions. Combining this with virtual classes makes it possible to define, at the programming language level, refinable hierarchical state machines. The combination of state machines and events provides, in particular, effective language support for the State design pattern as well as a form of Event-based AOP.

EJava and EScala are more recent developments of the same ideas applied to Java and Scala, respectively. EJava benefits from Java tooling with an eclipse plugin developed with the Spoofax Language Workbench. Unlike EJava and ECaesarJ, EScala makes it possible to dynamically register and unregister event handlers. It also benefits from a more efficient, compiler-based, implementation. As ECaesarJ, EScala is joint work with TUD.

Prototype implementations of these languages are available through [http://ecaesarj.gforge.inria.fr/](http://ecaesarj.gforge.inria.fr/).
5.3. Entropy

**Participants:** Jean-Marc Menaud [correspondent], Fabien Hermenier, Adrien Lèbre, Hien Nguyen Van, Rémy Pottier, Thomas Chavrier, Guillaume Le Louët.

Entropy is a virtual machine (VM) manager for clusters. The current prototype acts as an infinite control loop, which performs a globally optimized placement according to cluster resource usage, scheduler objectives and administrative rules.

Relying on an encapsulation of jobs into VMs, Entropy enables the implementation of finer scheduling policies through cluster-wide context switches, i.e., permutations of VMs present in the cluster. It thus supports a more flexible use of cluster resources and frees end-users from the burden of dealing with time estimates.

The major advantage of the Entropy system concerns the cluster-wide context switch process itself. Entropy computes a new viable configuration and an optimized reconfiguration plan. This plan describes the sequences of transitions to perform (i.e. the run, migrate, suspend/resume, stop VM operations) in order to transit from the current situation to the new one. As the cost of each action and the dependencies between them is considered, Entropy reduces the duration of each cluster-wide context switch by performing a minimum number of actions in the most efficient way.

Around this solution, we developed VMScript, a domain-specific language for administration of virtualized grid infrastructures. This language relies on set manipulation and is used to introspect physical and virtual grid architectures, thanks to query expressions, and notably to modify VM placement on machines. VMScript interacts with Entropy and can be used to define administrative placement rules.

In 2011, Entropy has been integrated into a product of a newly founded start-up EasyVirt (see Sec. 7.1 ). Entropy has also been tested or used by our partners Orange Labs, DGFiP (direction Générale des Finances Publiques), Bull, MACIF, Logica.

Entropy is available under the LGPL license at http://entropy.gforge.inria.fr/.

5.4. FPath and FScript

**Participants:** Thomas Ledoux [correspondent], Frederico Alvares.

FPath and FScript are two domain-specific languages (DSLs) dealing respectively with the navigation and the dynamic reconfiguration of Fractal architectures. *FPath* is a DSL for querying Fractal architectures. It is restricted to the introspection of architectures by browsing elements identified by their properties or location in the architecture. This focused domain allows FPath to offer a very concise and readable syntax and ensures correctness properties by construction (e.g. any query terminates in a finite time). *FScript* is a DSL dedicated to the reconfiguration of Fractal component architectures. It enables reconfiguration scripts to modify a Fractal architecture. Like FPath, FScript guarantees several properties by construction, e.g. termination of scripts by excluding the possibility of infinite loops. Moreover the FScript interpreter supports a transactional model of reconfigurations and the preservation of ACID properties.

An adaptation of FPath/FScript to FraSCAti, a component framework providing runtime support for the Service Component Architecture (SCA), has been developed by the INRIA Adam project-team. In that way, software architects are able to navigate using FPath notation through FraSCAti architectures and to reconfigure them with FScript. We have used this adaptation in our recent work [22] for reconfiguring cloud applications in order to reduce the energy footprint in cloud infrastructures.

FScript and its extensions are available under the LGPL license at http://fractal.ow2.org/fscript.

5.5. WildCAT

**Participants:** Thomas Ledoux [correspondent], Frederico Alvares.
WildCAT is a generic Java framework for context-aware applications. It permits the monitoring of large-scale applications by allowing developers to easily organize and access resources through a hierarchical organization backed with a powerful SQL-like language to inspect sensors data and to trigger actions upon particular conditions. WildCAT proposes two modes to inspect the resources: a pull mode relies on synchronous communication and a push one relies on asynchronous communication. In the pull mode, developers programmatically get and set attributes. In the push mode, developers register listeners on queries expressed over the events generated by the backend.

WildCAT has been developed by the team in the last years. We have used WildCAT in our recent work [22] for allowing cloud applications to listen events notification fired by the cloud infrastructure (e.g. whenever the pricing policy of cloud resources changes) or to detect changes on the application activity (e.g. to detect whenever the number of requests/s sharply increases/decreases) in order to launch the reconfiguration of cloud applications.

WildCAT is available under GPL v2 at http://wildcat.ow2.org.
5. Software

5.1. The ATL Model Transformation Language

URL: http://www.eclipse.org/m2m/atl/

With an eye on the normative work of the OMG (MOF, OCL, QVT, etc.), a new conceptual framework has been developed based on a second generation model transformation language called ATL. Although ATL influenced the OMG standard, the approach is more general as discussed in [8]. In 2004 IBM gave an Eclipse innovation award to the ATL project. In 2007 Eclipse recognized ATL as one central solution for model transformation and promoted it to the M2M project (see Eclipse.org/m2m). There are more than 200 industrial and academic sites using ATL today, and several Ph.D. thesis in the world are based on this work.

In 2011 we started a new evolution phase for ATL. Our mid-term plan is making of ATL the leading solution for building autonomous reactive transformation systems, i.e. transformation networks that can autonomously manage a set of dataflows among the application models.

Following this line, we implemented in [43] a new refinement mode for ATL, to support in-place transformations. This extension allows the dynamic manipulation of models while keeping them connected to runtime applications.

In [31] we presented a lazy execution algorithm for ATL. With it, the elements of the target model are generated only when and if they are accessed. This extension allows to build reactive transformation systems that react to requests of model elements, by triggering the necessary computation. Our lazy version of ATL enable also transformations that generate infinite target models, extending the application space of the model-transformation paradigm.

The latest evolution of the ATL engine is a full reactive version, able to activate the minimal computation for responding to updates or request on the involved models. This engine is studied to scalably support large ATL networks. In this line we also introduced an algorithm for simplifying ATL transformation chains.

5.2. MoDisco (Model Discovery)

URL: http://www.eclipse.org/modisco/

MoDisco is an open source Eclipse project that provides a generic and extensible framework dedicated to the elaboration of Model Driven Reverse Engineering (MDRE) solutions. Gathering contributions from both academics and industrials, the goal of the project is to federate common efforts in the model-based transformation of legacy software systems implemented using different technologies (e.g.: Java, COBOL, C). The first principle is to discover models out of legacy artifacts, representing appropriately all the relevant information, to be then used as part of reverse engineering processes for software understanding, evolution or modernization. Targeted scenarios include software (technical or architectural) migration of large legacy systems, but also retro-documentation, refactoring, quality assurance, etc. Within this context, MoDisco has collaborations with the OMG Architecture Driven Modernization (ADM) Task Force, for which the project provides several reference implementations of its standards: Knowledge Discovery Metamodel (KDM), Software Measurement Metamodel (SMM) and Abstract Syntax Tree Metamodel (ASTM).
The MoDisco framework is composed of a set of Eclipse plugins, and relies on the de-facto standard Eclipse Modeling Framework (EMF) for model handling. Thanks to its modular architecture, it allows completely covering the three steps of a standard MDRE approach: 1) Discovery (i.e. extracting a complete model of the source code), 2) Understanding (i.e. browsing and providing views on this model for a given purpose) and 3) Transformation (evolving the model towards a new technology, architecture, etc). More specifically, as part of its Infrastructure layer, MoDisco offers the set of generic (i.e.; legacy technology-independent) reusable components really useful to build the core of MDRE solutions: Discovery Manager and Workflow for MDRE task orchestration, Model Browser for advanced navigation in complex models, model extension and customization capabilities for understanding (e.g.; views definition), etc. As part of its Technologies layer, it provides an advanced support for the Java, JEE and XML technologies, including complete metamodels, corresponding model discoverers, transformations, code generators, customizations, query libraries, etc.

MoDisco (or some of its components) is being used by different partners including other academics, industrials (e.g.; Sodifrance on several of their real modernization projects for their customers) or Eclipse projects (e.g.; Eclipse-MDT Papyrus as developed by CEA). Moreover, the Eclipse-EMFT EMF Facet project has been initiated as a MoDisco spin-off, in order to externalize some features which are not actually specific to reverse engineering problems and thus may be reused in many different contexts (cf. corresponding EMF Facet section).

5.3. EMFtoCSP

URL: http://code.google.com/a/eclipse.org/p/emftocsp/

EMFtoCSP is a tool for the verification of precisely defined conceptual models and metamodels. For these models, the definition of the general model structure (using UML or EMF) is supplemented by OCL constraints. The Eclipse Modeling Development Tools (MDT 2) provides mature tool support for such OCL-annotated models with respect to model definition, transformation, and validation.

However, an additional important task that is not supported by Eclipse MDT is the assurance of model quality. A systematical assessment of the correctness of such models is a key issue to ensure the quality of the final application. EMFtoCSP fills this gap by provided support for automated model verification in Eclipse.

Essentially, the EMFtoCSP is a sophisticated bounded model finder that yields instances of the model that conform not only to the structural definition of the model (e.g., the multiplicity constraints), but also to the OCL constraints. Based on this core, several correctness properties can be verified:

1. Satisfiability – is the model able to express our domain? For this check, the minimal number of instances and links can be specified to ensure non-trivial instances.
2. Unsatisfiability – is the model unable to express undesirable states? To verify this, we add further constraints to the model that state undesired conditions. Then we can check if is it impossible to instantiate the amended model.
3. Constraint subsumption – is one constraint already implied by others (and could therefore be removed)?
4. Constraint redundancy – do different constraints express the same fact (and could therefore be removed)?

To solve these search problems, EMFtoCSP translates the EMF/OCL (resp. UML/OCL) model into a problem of constraint logic programming and employs the Eclipse CLP solver 3 to solve it. This way, constraint propagation is exploited to tackle the (generally NP-hard) search.

---

2 http://www.eclipse.org/modeling/mdt/?project=ocl
3 http://eclipseclp.org/
The tool is a continuation of the UMLtoCSP approach [52] developed previously by Jordi Cabot, Robert Clarísó and Daniel Riera. It provides a generic plugin framework for Eclipse to solve OCL-annotated models using constraint logic programming. Apart from already supported Ecore and UML metamodels, further metamodels can be added easily in the future. Similarly, other constraint solving back-ends can be integrated. It is provided under the Eclipse Public License.

5.4. AMW (AtlanMod Model Weaver)

URL: http://wiki.eclipse.org/AMW

AMW is an open source Eclipse project, under the Eclipse Public License (EPL), that provides a generic and extensible tooling dedicated to model weaving. It can be used to establish, represent and manage any kind of links (correspondences) between elements coming from different models (or metamodels). These links are stored in separated weaving models, allowing them to be exchanged and reused in various contexts using different modeling techniques. Thus, weaving models are actually relevant in many concrete application scenarios where several models are involved: (model driven) tool interoperability, mapping definition, transformation specification, traceability, model annotation, model merging, model comparison, etc.

The AMW tooling is composed of several Eclipse plugins, and relies on the de-facto standard Eclipse Modeling Framework (EMF) for model handling. The provided model weaving workbench comes notably with a base weaving editor, reusable matching algorithms and weaving model serialization capabilities. These components are fully generic, so that they can be extended and adapted to any specific weaving metamodel (defining any kind of weaving links). AMW also offers a connector allowing to both use and produce weaving models as part of model-to-model transformations written in ATL.

AMW is being used by more than 40 user sites, including research labs and major companies (NASA, BAE, Versata, Mia-Software, Obeo, etc.). Currently part of the Eclipse-GMT project (to be terminated in the coming months), AMW is now in the process of being migrated into the Eclipse-M2M ATL project.

5.5. AM3 (AtlanMod MegaModel Management)

URL: http://wiki.eclipse.org/index.php/AM3

The AtlanMod Megamodel Management tool offers several functionalities for modeling in the large [3], i.e. for handling several related models (either terminal models, metamodels or transformation models) used as part of a complex modeling project.

The main component in AM3 is a generic megamodel manager that allows the user to browse and manipulate a set of related models. This manager knows the semantic relations between all these models. These relations are often associated to a given weaving model allowing not only navigating the traces between models, but also the traces between model elements. Since the links are stored externally as weaving models, the participating models do not get polluted and may be used as they are. Furthermore it is possible to handle multiple traceability chains going through similar models.

AM3 provides also a textual domain-specific language for model management called MoScript (URL: http://wiki.eclipse.org/MoScript). With MoScript, users can automate model management tasks by means of textual scripts written in an extension of the OCL language. For instance, user may write queries (based on model content, structure, relationships, and behaviour derived through on-the-fly simulation) to retrieve models from model repositories, manipulate them (e.g., by running transformations on sets of models), and store them back in the repository. MoScript also allows to populate and update the megamodel automatically by doing reverse engineer of simple modeling artifact repositories.

The generic tool for megamodel management has been used by different partners for several use cases like operationalization of chains of transformations.
5.6. Virtual EMF (Model Virtualization)

URL: http://code.google.com/a/eclipselabs.org/p/virtual-emf/

Virtual EMF is an Eclipse plugin built on top of EMF that enables the creation and manipulation of virtual models, i.e., models whose elements do not contain concrete data, but are rather proxies to elements contained in other models. The idea is related to that of model composition, as it aims capturing the (often overlapping) concepts a set of models as one single global model. This is a frequently faced problem as, in complex scenarios, modelers often have to deal with a large number of heterogeneous and interrelated models, and most times the view a specific kind of user requires does not correspond to any of these models, but is a combination of cross-domain information scattered among several ones.

Current composition techniques rely on the materialization of the composed model, an approach that poses some important limitations in terms of (i) efficiency, as they do not scale (the data duplication mechanism they use implies in extra memory usage and time-consuming generation of the composed model), (ii) synchronicity, as updates in the composed model are not propagated to the original ones (or vice-versa), thus losing consistency, or even (iii) interoperability, as in some cases the composed model requires a specific API/tool to be handled.

Virtual EMF allows overcoming the limitations above. A virtual model provides to tools/user the illusion of working with a regular model whereas, in fact, all model access and manipulation requests are transparently redirected to its set of virtualized models. It serves as a centralized and transparent access point to a set of interconnected models, allowing users to easily compose, weave and link them. Thus providing the following beneficial properties:

- **Interoperability**: it behaves as a normal model. Therefore, compatibility with existing EMF-based solutions/tools (e.g. models transformations, model editors, ...) is guaranteed;
- **Synchronization**: changes are automatically and transparently propagated between virtual and original models;
- **Scalability**: support for very big models;
  - low memory usage: no data duplication, direct access to original model elements;
  - faster generation time: no need for (time-consuming) information cloning operations (e.g. executing a model transformation);
- **Genericity**: support for several types of inter-model relationships (e.g. merge, association, filter) and extension capabilities for their semantics.

Virtual EMF is available as an open-source project on Eclipse Labs. It has been presented in a talk in EclipseCon Europe 2011 and contributed by the AtlanMod team to the CESAR project.

5.7. EMF Facet

URL: http://www.eclipse.org/modeling/emft/facet/

EMF Facet is an open source Eclipse project, under the Eclipse Public License (EPL), that provides a generic and extensible framework dedicated to the dynamic and non-intrusive extension of models. It can be used to extend already existing metamodels with additional concepts and properties, the corresponding models being then transparently augmented, reduced or modified accordingly at runtime. Such a metamodel extension is called a facet, and can be specified on top of any metamodel in EMF Ecore. The underlying mechanism is based on the runtime execution of queries on the models corresponding to the faceted metamodels. Facets are notably particularly relevant for obtaining different views on existing models without having to actually alter them with any extra data.
The EMF Facet framework is composed of several Eclipse plugins, and relies on the de-facto standard Eclipse Modeling Framework (EMF) for model handling. The facet definitions are stored as facet models, allowing them to be exchanged and reused in various contexts. The queries can be implemented using any suitable query language (e.g.; ATL, OCL, Java, XPath), as far as the corresponding adaptors exist and are correctly registered within the framework. The proposed tooling includes dedicated editors for creating, editing and saving both facet and query definitions, the implemented support for Java, OCL and ATL queries, a Table Editor for visualizing query results. An advanced support for the model display customization (e.g.; icons, colors, fonts) is also provided as part of the framework.

EMF Facet is currently intensively used in MoDisco for extracting and displaying different specific views from large models of legacy systems. Its extension and customization capabilities are actually integrated into several MoDisco components, such as notably the MoDisco Model Browser. However, different other integration possibilities will be also explored in the future.

5.8. Portolan (Model-Driven Cartography)

URL: [http://code.google.com/a/eclipselabs.org/p/portolan/](http://code.google.com/a/eclipselabs.org/p/portolan/)

Processing large amounts of data to extract useful information is an essential task within companies. To help in this task, visualization techniques have been commonly used due to their capacity to present data in synthesized views, easier to understand and manage. However, achieving the right visualization display for a data set is a complex cartography process that involves several transformation steps to adapt the (domain) data to the (visualization) data format expected by visualization tools. With its simple core principles and set of base generic techniques (metamodeling, model transformation, model weaving, etc), Model-Driven Engineering (MDE) provides the relevant support for bridging the gap between data sets and visualization tools and thus for designing and implementing Cartography solutions. The proposed Portolan prototype is a concrete illustration of both a model-based and model-driven Cartography platform. Thus, the objective of Portolan is to facilitate the identification of interoperability solutions between tools by:

1. discovering (at least semi-automatically) maps of given situations in terms of deployed tools and relationships between them;
2. easily navigating and editing these maps;
3. augmenting or specializing them with both manually-entered and computed information;
4. visualizing them, using different customizable ways, in order to facilitate their understanding.

To this aim, the Portolan platform integrates:

- a set of default DSLs like GraphML, KML, Excel;
- visual displays based on Prefuse, Google Maps;
- modeling tools such as ATL language, Ecore modeler, etc.

It includes also an extension mechanism allowing the tool customization for advanced users. This recently developed generic tooling for cartography has already been used during the first action of our collaboration with BNP Paribas, as well as in the context of the IDM++ project.

5.9. The AmmA ToolBox

ATL, AMW, TCS, MoDisco, and AM3 are among the most important Eclipse.org components produced by the AtlanMod team. However there are also other components and a lot of functionalities, examples, and use cases made available and necessary to express solutions to many problems. The whole set of contributions composes the AmmA platform.
5.10. Industrialization strategy for research prototypes

Research labs, as priority innovation providers, are also indirectly key actors of the Software Engineering market. However, even if they already initiate the promotion of many new innovations to the industry, an important collaborative effort is still needed in order to actually transfer the corresponding techniques or technologies from the research lab to the company. Based on the AtlanMod concrete experience with the previously mentioned open source tools/projects, we have extracted a pragmatic approach [37] for transforming the results of scientific experimentation into practical industrial solutions.

While dealing with innovation, this approach is also innovation-driven itself, as the action is actually conducted by the research lab via a technology transfer. Three different partners are directly involved in this process, using open source as the medium for maintaining a constant interaction between all of them:

- **Use Case Provider.** Usually a company big enough to have to face real complex industrial scenarios which need to be solved (at least partially) by applying new innovative principles and techniques;
- **Research Lab.** Usually a group from a research institute (public or private) or university evaluating the scientific relevance of the problems, identifying the research challenges and prototyping possible solutions;
- **Technology Provider.** Usually a small or medium company, with a particular technical expertise on the given domain or Software Engineering field, building and delivering the industrial version of the designed solutions;

From our past and current experience, three main characteristics of this industrialization business model can be highlighted:

- **Win-win situation.** Each partner can actually focus on its core activity while also directly benefiting from the results obtained by the others (notably the research lab can continue to do research);
- **Application-driven context.** The end-user need is at the origin of the process, which finally makes the developed solution actually relevant;
- **Iterative process.** The fact of having three distinct partners requires different regular and consecutive exchanges between all of them.
5. Software

5.1. Intrusion Detection

Members of Supélec have developed several intrusion detectors. Blare implements our approach of illegal information flow detection at the OS level. This implementation is a modification of a standard Linux kernel and it monitors information flows between typical OS containers as files, sockets or IPC. System active entities are processes viewed as black-boxes as we only observe their inputs and outputs. Detection at the OS level is in some cases too coarse-grained to avoid the generation of false positives and to detect attacks targeting the application logic. Even if it remains convenient to define the security policy at the OS-level, sound illegal information flow detection implies an additional detection at the language level. This has led us to implement a detector for Java applications, JBlare, to complement the detection at the OS level. JBlare extends the OS-level one by refining the observation of information flows at the language level.

GNG is an intrusion detection system that correlates different sources (such as different logs) in order to identify attacks against the system. The attack scenarios are defined using the Attack Description Language (ADeLe) proposed by our team, and are internally translated to attack recognition automats. GNG intends to define time efficient algorithms based on these automats to recognize complex attack scenarios.

SIDAN (Software Instrumentation for Detecting Attacks on Non-control-data) is a tool that aims to instrument automatically C-language software with assertions whose role is to detect attacks against the software. This tool is implemented as a plugin of the FRAMA-C framework that provides an implementation of static analysis techniques.

5.2. Privacy

GEPETO (GEoPrivacy-Enhancing TOolkit) is an open source software for managing geolocated data (currently in development in cooperation with LAAS). GEPETO can be used to visualize, sanitize, perform inference attacks and measure the utility of a particular geolocated dataset. For each of these actions, a set of different techniques and algorithms can be applied. The global objective of GEPETO is to enable a user to design, tune, experiment, and evaluate various sanitization algorithms and inference attacks as well as visualizing the following results and assessing their utility.

5.3. Reliable Programming

The Prometeus project, part of the Inria Gforge, is a software environment for reliable programming. The basic elements of Prometeus are Eva, a component-based framework and Adam, a set of group communication services. Eva is an implementation of a component model that aims at supporting the development of distributed abstractions and high-level communication protocols. Adam is a library of agreement components, based on the component model implemented by Eva. The central element of the Adam library is GAC (Generic Agreement Component). It implements a generic and adaptive fault-tolerant consensus algorithm that can be customized to cope with the characteristics of the environment. Moreover, thanks to a set of versatile methods, its behavior can be tuned to fit the exact needs of a specific agreement problem. A range of fundamental Adam components are implemented as specializations of this GAC component. The Adam library currently includes the most important components for reliable distributed programming (Group Membership, Atomic Broadcast).
5. Software

5.1. Jolie

Members of Focus have recently developed Jolie [7] (Java Orchestration Language Interpreter Engine, see http://www.jolie-lang.org/). Jolie is a service-oriented programming language. Jolie can be used to program services that interact over the Internet using different communication protocols. Differently from other Web Services programming languages such as WS-BPEL, Jolie is based on a user-friendly C/Java-like syntax (more readable than the verbose XML syntax of WS-BPEL) and, moreover, the language is equipped with a formal operational semantics. This language is used for the proof of concepts developed around Focus activities. For instance, contract theories can be exploited for checking the conformance of a Jolie program with respect to a given contract. A spin-off, called “Italiana Software”, has been launched around Jolie, its general aim is to transfer the expertise in formal methods for Web Services matured in the last few years onto Service Oriented Business Applications. The spin-off is a software producer and consulting company that offers service-oriented solutions (for instance, a “single sign-on” application) based on the Jolie language.

In 2011 the development of Jolie has continued. The main activities have been:

- A new session message-routing mechanism, based on correlation sets has been implemented. This mechanism makes message routing programmable from inside Jolie code.
- A new primitive for (smart) service aggregation.
- A graphical editor.
- An integrated development environment.

Moreover, this year Jolie has been used for teaching, in a 30-hour master course at IT University of Copenhagen, Denmark.

5.2. Others

Below we list some software that has been developed in Focus in previous years but that during 2011 has not substantially changed. Short descriptions of these items can be found in the Focus activity report for 2010.

- PiDuce (see http://www.cs.unibo.it/PiDuce/) is a prototype for experimenting Web services technologies, based on theories of process calculi and XML documents and schemas [3].
- IntML is a functional programming language guaranteeing sublinear space bounds for all programs [57].
5. Software

5.1. Introduction

Most INDES software packages, even the older stable ones that are not described in the following sections are freely available on the Web. In particular, some are available directly from the INRIA Web site:

http://www.inria.fr/centre/sophia/innovation

Most other software packages can be downloaded from the INDES Web site:

http://www-sop.inria.fr/teams/indes

5.2. Functional programming

Participants: Frédéric Boussinot [Inria], Thomas Gazagnaire [Inria], Zhengqin Luo [Inria], Cyprien Nicolas [Inria], Tamara Rezk [Inria], Bernard Serpette [Inria], Manuel Serrano [correspondant].

5.2.1. The Bigloo compiler

The programming environment for the Bigloo compiler [5] is available on the INRIA Web site at the following URL: http://www-sop.inria.fr/teams/indes/fp/Bigloo . The distribution contains an optimizing compiler that delivers native code, JVM bytecode, and .NET CLR bytecode. It contains a debugger, a profiler, and various Bigloo development tools. The distribution also contains several user libraries that enable the implementation of realistic applications.

BIGLOO was initially designed for implementing compact stand-alone applications under Unix. Nowadays, it runs harmoniously under Linux and MacOSX. The effort initiated in 2002 for porting it to Microsoft Windows is pursued by external contributors. In addition to the native back-ends, the BIGLOO JVM back-end has enabled a new set of applications: Web services, Web browser plug-ins, cross platform development, etc. The new BIGLOO .NET CLR back-end that is fully operational since release 2.6e enables a smooth integration of Bigloo programs under the Microsoft .NET environment.

5.2.2. Camloo

Camloo is a caml-light to bigloo compiler, which was developed few years ago to target bigloo 1.6c. New major releases 0.4.x of camloo have been done to support bigloo 3.4 and bigloo 3.5. Camloo make it possible for the user to develop seamlessly a multi-language project, where some files are written in caml-light, in C, and in bigloo. Unlike the previous versions of camloo, 0.4.x versions do not need a modified bigloo compiler to obtain good performance. Currently, the only supported backend for camloo is bigloo/C. We are currently rewriting the runtime of camloo in bigloo to get more portability and to be able to use HOP and camloo together.

5.2.3. The FunLoft language

FunLoft (described in http://www-sop.inria.fr/teams/indes/rp/FunLoft ) is a programming language in which the focus is put on safety and multicore.

FunLoft is built on the model of FairThreads which makes concurrent programming simpler than usual preemptive-based techniques by providing a framework with a clear and sound semantics. FunLoft is designed with the following objectives:

- provide a safe language, in which, for example, data-races are impossible.
- control the use of resources (CPU and memory), for example, memory leaks cannot occur in FunLoft programs, which always react in finite time.
- have an efficient implementation which can deal with large numbers of concurrent components.
- benefit from the real parallelism offered by multicore machines.
A first experimental version of the compiler is available on the Reactive Programming site http://www-sop.inria.fr/teams/indes/rp. Several benchmarks are given, including cellular automata and simulation of colliding particles.

5.3. Web programming

Participants: Gérard Berry [Inria], Cyprien Nicolas [Inria], Manuel Serrano [correspondant].

5.3.1. The HOP web programming environment

HOP is a higher-order language designed for programming interactive web applications such as web agendas, web galleries, music players, etc. It exposes a programming model based on two computation levels. The first one is in charge of executing the logic of an application while the second one is in charge of executing the graphical user interface. HOP separates the logic and the graphical user interface but it packages them together and it supports strong collaboration between the two engines. The two execution flows communicate through function calls and event loops. Both ends can initiate communications.

The HOP programming environment consists in a web broker that intuitively combines in a single architecture a web server and a web proxy. The broker embeds a HOP interpreter for executing server-side code and a HOP client-side compiler for generating the code that will get executed by the client.

An important effort is devoted to providing HOP with a realistic and efficient implementation. The HOP implementation is validated against web applications that are used on a daily-basis. In particular, we have developed HOP applications for authoring and projecting slides, editing calendars, reading RSS streams, or managing blogs.

HOP has won the software open source contest organized by the ACM Multimedia Conference 2007 http://mmc36.informatik.uni-augsburg.de/acmmm2007/. It is released under the GPL license. It is available at http://hop.inria.fr

5.4. Language-based security

Participants: Zhengqin Luo [Inria], Tamara Rezk [correspondant].

5.4.1. CFlow

The prototype compiler “CFlow” takes as input code annotated with information flow security labels for integrity and confidentiality and compiles to F# code that implements cryptography and protocols that satisfy the given security specification.

Cflow has been coded in F#, developed mainly on Linux using mono (as a substitute to .NET), and partially tested under Windows (relying on .NET and Cygwin). The code is distributed under the terms of the CeCILL-B license http://www.msr-inria.inria.fr/projects/sec/cflow/index.html.

5.4.2. FHE type-checker

We have developed a type checker for programs that feature modern cryptographic primitives such as fully homomorphic encryption. The type checker is thought as an extension of the “CFlow” compiler developed last year on the same project. It is implemented in F#. The code is distributed under the terms of the CeCILL-B license http://www.msr-inria.inria.fr/projects/sec/cflow/index.html.

5.4.3. Mashic compiler

The Mashic compiler is applied to mashups with untrusted scripts. The compiler generates mashups with sandboxed scripts, secured by the same origin policy of the browsers. The compiler is written in Bigloo and can be found at http://www.mashic.net.
5.5. Old software

5.5.1. Skribe

SKRIBE is a functional programming language designed for authoring documents, such as Web pages or technical reports. It is built on top of the SCHEME programming language. Its concrete syntax is simple and looks familiar to anyone used to markup languages. Authoring a document with SKRIBE is as simple as with HTML or LaTeX. It is even possible to use it without noticing that it is a programming language because of the conciseness of its original syntax: the ratio tag/text is smaller than with the other markup systems we have tested.

Executing a SKRIBE program with a SKRIBE evaluator produces a target document. It can be HTML files for Web browsers, a LaTeX file for high-quality printed documents, or a set of info pages for on-line documentation.

5.5.2. Scheme2JS

Scm2JS is a Scheme to JavaScript compiler distributed under the GPL license. Even though much effort has been spent on being as close as possible to R5RS, we concentrated mainly on efficiency and interoperability. Usually Scm2JS produces JavaScript code that is comparable (in speed) to hand-written code. In order to achieve this performance, Scm2JS is not completely R5RS compliant. In particular it lacks exact numbers. Interoperability with existing JavaScript code is ensured by a JavaScript-like dot-notation to access JavaScript objects and by a flexible symbol-resolution implementation.

Scm2JS is used on a daily basis within HOP, where it generates the code which is sent to the clients (web-browsers). Scm2JS can be found at http://www-sop.inria.fr/indes/scheme2js.
5. Software

5.1. SAFDIS

Contact: Jean-Louis Pazat, Jean-Louis.Pazat@irisa.fr
URL: http://www.irisa.fr/myriads/software/folder.2011-12-13.8949308917/
Status: Version 1.0
License: TBD

Presentation: SAFDIS (Self Adaptation for Distributed Services) is a generic framework allowing the self-adaptation of distributed service based applications within a highly volatile context. Compared to other adaptation frameworks, the main advantages of SAFDIS are its genericity, its distributed nature and the focus on SOAs. SAFDIS is in its final implementation and testing phase within the Myriads team and is being used with a real life use case for emergency services.

The current implementation of SAFDIS is based on a Java OSGi implementation. SAFDIS is written in Java and organized into OSGi bundles. SAFDIS is not tight to any specific operating system and work within any JAVA 1.6 platform. An OSGi implementation is needed (such as the Apache Felix http://felix.apache.org or Equinox eclipse.org/equinox implementations). In order to benefit from the reactive adaptation tools, the Jess engine is also needed as an OSGi bundle (http://www.jessrules.com).

5.2. HOCL-tools

Contact: Cédric Tedeschi, Cedric.Tedeschi@irisa.fr
Status: Version 1.0 to be released
License: TBD

Presentation: HOCL (Higher Order Chemical Language) is a chemical programming language based on the chemical metaphor presented before (see Section 3.5). It was developed for several years within the PARIS team. Within HOCL, following the chemical metaphor, computations can be regarded as chemical reactions, and data can be seen as molecules which participate in these reactions. If a certain condition is held, the reaction will be triggered, thus continuing until it gets inert: no more data can satisfy any computing conditions. To realize this program paradigm, a multiset is implemented to act as a chemical tank, containing necessary data and rules. An HOCL program is then composed of two parts: chemical rule definitions (reaction rules) and multiset definition (data). More specifically, HOCL provides the high order: reaction rules are molecules that can be manipulated like any other molecules. In other words, HOCL programs can manipulate other HOCL programs.

An HOCL compiler was developed using java to execute some chemical programs expressed with HOCL. This compiler is based on the translation of HOCL programs to java code. As a support for service coordination and service adaptation (refer to Section 6.4), we recently extended the HOCL compiler with the support of decentralized workflow execution. Works around the implementation of a distributed multiset gave birth to an underlying layer for this compiler, making it able to deploy HOCL programs transparently over large scale platforms. This last part is currently considered to be interfaced with the current HOCL compiler. All these features are planned to be released under the common name of HOCL-tools.

- Version: 1
5.3. XtreemOS

Contact: Yvon Jégou, Yvon.Jegou@inria.fr
Status: Version 3.0
License: GPL-2/BSD depending on software packages composing the system

Presentation: XtreemOS is a Grid Operating system based on Linux with native support for virtual organizations. Three flavours of XtreemOS are developed for individual PCs, clusters and mobile devices (PDA, notebooks and smartphones). XtreemOS has been developed by the XtreemOS consortium. The third public version of XtreemOS has been released in December 2010. XtreemOS has been presented at the Contrail summer school, Presqu’île de Giens, France in July 2011 (Y. Jégou), SC’11, Seattle, USA (Y. Jégou, Ch. Morin) in November 2011. XtreemOS software is a set of services developed in Java, C++ and C. XtreemOS cluster version leverages KERRIGHED single system image operating system. A permanent testbed composed of computers provided by several XtreemOS partners has been public since fall 2010. In 2011, XtreemOS has been packaged for the OpenSuse Linux distribution. Ready-to-use XtreemOS virtual machine images have been made available for the community.

Active contributors (from Myriads team): Amine Belhaj, Jérôme Gallard, Rémy Garrigue, Yvon Jégou, Christine Morin, Yann Radenac, Pierre Riteau.

5.4. Contrail Virtual Execution Platform (VEP)

Contact: Yvon Jégou, Yvon.Jegou@inria.fr
URL: http://www.contrail-project.eu
Status: Version 1.0
License: BSD

Presentation: Virtual Execution Platform (VEP) is a Contrail service that sits just above IaaS layer at the service provider end of the Contrail cloud federation. The VEP provides a uniform interface for managing the whole lifecycle of elastic applications on the cloud and hides the details of the IaaS layer to the user. VEP applications are described in OVF (Open Virtualization Format) standard format. Resource usage is controlled by CEE (Constrained Execution Environment) rules which can be derived from SLAs (Service Level Agreement). The VEP integrates a monitoring system where the major events about the application, mainly resource usage, are made available to the user.

The VEP service provides a RESTful interface and can be exploited directly by users on top of the provider IaaS. OpenNebula and OCCI-based IaaS interfaces are currently supported.

Active contributors (from Myriads team): Roberto-Gioacchino Cascella, Florian Dudouet, Piyush Harsh, Yvon Jégou, Christine Morin.

5.5. Snooze

Contact: Christine Morin, Christine.Morin@inria.fr
URL: http://www.irisa.fr/myriads/software/snooze/
Status: Version 1.0
License: TBD
Presentation: Snooze [56], is a novel VM management framework for private clouds which is designed to scale across thousands of nodes. Unlike the existing cloud management frameworks, Snooze utilizes a self-organizing hierarchical architecture and performs distributed VM management. Particularly, VM management tasks are performed by multiple managers, with each manager having only a partial view of the system. Moreover, fault-tolerance is provided at all levels of the hierarchy by replication and integrated leader election algorithm. Consequently, the system is able to self-heal and continue its operation despite system component failures. In addition, VM monitoring is integrated into the framework and a generic scheduling engine exists to support advanced (e.g., consolidation) scheduling policies. Last but not least, a RESTful command line interface (CLI) exists to support virtual cluster (VC) definitions and management (i.e., start, shutdown, destroy, suspend, etc.) as well hierarchy visualization and exporting in GraphML format.

Snooze is fully implemented from scratch in Java and currently comprises approximately 15,000 lines of maintainable abstractions-based code. The leader election algorithm is built on top of the Apache ZooKeeper [59] highly available and reliable coordination system. In order to provide a uniform interface to the underlying hypervisors and support transparent VM monitoring and management, Snooze integrates the libvirt virtualization library.

The first Snooze prototype [41] has been developed and evaluated on Grid5000 experimental testbed.

Active contributors (from Myriads team): Eugen Feller, Christine Morin.

5.6. Resilin

Contact: Christine Morin, Christine.Morin@inria.fr

URL: http://www.irisa.fr/myriads/software/resilin/

Status: Version 0.1

License: TBD

Presentation: Resilin [45] is a system to easily create execution platforms over distributed cloud resources for executing MapReduce computations. Resilin implements the Amazon Elastic MapReduce web service API with resources from other clouds than Amazon EC2, such as private and community clouds. Resilin allows users to perform MapReduce computations on other infrastructures than Amazon EC2, and offers more flexibility: users are free to select different types of virtual machines, different operating systems or newer Hadoop versions. Users only have to submit computations to the service through a web service API, and Resilin takes care of provisioning, configuring and managing cloud-based Hadoop execution platforms, potentially using multiple clouds.

Resilin is implemented in the Python language. It uses the boto library in order to interact with EC2-compatible clouds. Resilin has been evaluated on the Grid’5000 experimental testbed. Our comparison with the Amazon Elastic MapReduce service shows similar performance.

Active contributors (from Myriads team): Pierre Riteau, Christine Morin.

5.7. Saline

Contact: Christine Morin, Christine.Morin@inria.fr

URL: https://www.grid5000.fr/mediawiki/index.php/VMdeploy

Status: Version V1.0 (experimental)

License: BSD
Presentation: Saline (former called VMdeploy) is a generic framework to deploy and manage encapsulated user jobs in virtual machines (VMs) at grid level by moving them from one site to another transparently for the encapsulated jobs [58]. Moreover, Saline is non-intrusive and can be used with any non-modified Grid resource management systems (RMSs).

Saline deploys and configures a set of VMs according to the user needs. Then, periodically, Saline takes snapshots of the running VMs and saves them on a dedicated node in an efficient way [57]. In addition, Saline checks the status of the running VMs. If something wrong happens i.e. one or more VMs have failed due for instance to a node failure or to the arrival of a higher priority job, Saline redeploys the set of VMs from the latest snapshot taken on new available resources provided by the Grid RMS. Thanks to Saline, the redeployment of the snapshot is done in a transparent way from the encapsulated job point of view.

In its current implementation, Saline is programmed in bash and C. It uses Libvirt in order to create, to snapshot, and to restart the VMs. It means that Saline can deploy and manage KVM and XEN VMs or any other VMs usable with Libvirt. In addition, the architecture of Saline is very modular in order to have a clear and easily extensible code.

Active contributors (from Myriads team): Jérôme Gallard, Christine Morin
OASIS Project-Team

5. Software

5.1. ProActive


url: Proactive Parallel Suite

ProActive is a Java library (Source code under AGPL license) for parallel, distributed, and concurrent computing, also featuring mobility and security in a uniform framework. With a reduced set of simple primitives, ProActive provides a comprehensive API to simplify the programming of applications that are distributed on a Local Area Network (LAN), on cluster of workstations, Clouds, or on Internet Grids.

The library is based on an Active Object pattern that is a uniform way to encapsulate:

- a remotely accessible object,
- a thread,
- an actor with its own script,
- a server of incoming requests,
- a mobile and potentially secure agent.

and has an architecture to inter-operate with (de facto) standards such as:

- Web Service exportation (Apache Axis2 and CXF),
- HTTP transport,
- ssh, rsh, RMI/ssh tunnelling,
- Globus: GT2, GT3, GT4, gsi, Unicore, ARC (NorduGrid)
- LSF, PBS, Sun Grid Engine, OAR, Load Leveler

ProActive is only made of standard Java classes, and requires no changes to the Java Virtual Machine, no preprocessing or compiler modification; programmers write standard Java code. Based on a simple Meta-Object Protocol, the library is itself extensible, making the system open for adaptations and optimisations. ProActive currently uses the RMI Java standard library as default portable transport layer, but others such as Ibis or HTTP can be used instead, in an adaptive way.

ProActive is particularly well-adapted for the development of applications distributed over the Internet, thanks to reuse of sequential code, through polymorphism, automatic future-based synchronisations, migration of activities from one virtual machine to another. The underlying programming model is thus innovative compared to, for instance, the well established MPI programming model.

In order to cope with the requirements of large-scale distributed and heterogeneous systems like the Grid, many features have been incorporated into ProActive, including support for many transport and job submission protocols, GCM component support, graphical visualization interface, object migration, distributed and non-functional exception handling, fault-tolerance and checkpointing mechanisms; file transfer capabilities, a job scheduler, a resource manager able to manage various hosting machines, support for JMX and OSGi capabilities, web service object exposition, an SCA personality, etc.

ProActive is a project of the former ObjectWeb, now OW2 Consortium. OW2 is an international consortium fostering the development of open-source middleware for cutting-edge applications: EAI, e-business, clustering, grid computing, managed services and more. For more information, refer to [5] [42] and to the web pages http://www.objectweb.org and http://proactive.inria.fr/ which list several white papers.
ProActive management, distribution, support, and commercialisation is now ensured by the start-up company ActiveEon (http://www.activeeon.com), in the context of a collaboration with INRIA and UNS.

5.2. Vercors platform

**Participants:** E. Madelaine, R. Halalai, L. Henrio, A. Savu, M. Alexe.

The Vercors tools (http://www-sop.inria.fr/oasis/Vercors) include front-ends for specifying the architecture and behaviour of components in the form of UML diagrams. We translate these high-level specifications, into behavioural models in various formats, and we also transform these models using abstractions. In a final step, abstract models are translated into the input format for various verification toolsets. Currently we mainly use the various analysis modules of the CADP toolset.

- Our main effort this year was based on the development of a quite large case-study, two orders of magnitude larger than our previous experiments. This study was the opportunity to develop new methods for encoding our models using a new combination of CADP formalisms, combining compositional approaches, abstraction techniques, and distributed model-checking [22]. The implementation of these methods in the Vercors tools has started.

We have also been conducting experiments towards the next generation of specification formalism editors for VerCors, using the Papyrus UML-based environment.
5. Software

5.1. DiaSuite: a Development Environment for Sense/Compute/Control Applications

Participants: Charles Consel [correspondent], Benjamin Bertran, Ghislain Deffrasnes, Amélie Marzin, Damien Cassou, Julien Bruneau, Emilie Balland.

Despite much progress, developing a pervasive computing application remains a challenge because of a lack of conceptual frameworks and supporting tools. This challenge involves coping with heterogeneous devices, overcoming the intricacies of distributed systems technologies, working out an architecture for the application, encoding it in a program, writing specific code to test the application, and finally deploying it.

DiaSuite is a suite of tools covering the development life-cycle of a pervasive computing application:

- **Defining an application area.** First, an expert defines a catalog of entities, whether hardware or software, that are specific to a target area. These entities serve as building blocks to develop applications in this area. They are gathered in a taxonomy definition, written in the taxonomy layer of the Diaspec language.

- **Designing an application.** Given a taxonomy, the architect can design and structure applications. To do so, the Diaspec language provides an application design layer [33]. This layer is dedicated to an architectural pattern commonly used in the pervasive computing domain [24]. Describing the architecture application allows to further model a pervasive computing system, making explicit its functional decomposition.

- **Implementing an application.** We leverage the taxonomy definition and the architecture description to provide dedicated support to both the entity and the application developers. This support takes the form of a Java programming framework, generated by the Diagen compiler. The generated programming framework precisely guides the developer with respect to the taxonomy definition and the architecture description. It consists of high-level operations to discover entities and interact with both entities and application components. In doing so, it abstracts away from the underlying distributed technologies, providing further separation of concerns.

- **Testing an application.** Diagen generates a simulation support to test pervasive computing applications before their actual deployment. An application is simulated in the Diasim tool, without requiring any code modification. Diasim provides an editor to define simulation scenarios and a 2D-renderer to monitor the simulated application. Furthermore, simulated and actual entities can be mixed. This hybrid simulation enables an application to migrate incrementally to an actual environment.

- **Deploying a system.** Finally, the system administrator deploys the pervasive computing system. To this end, a distributed systems technology is selected. We have developed a back-end that currently targets the following technologies: Web Services, RMI, SIP and OSGI. This targeting is transparent for the application code. The variety of these target technologies demonstrates that our development approach separates concerns into well-defined layers.

This development cycle is summarized in the Figure 1.

See also the web page [http://diasuite.inria.fr](http://diasuite.inria.fr).
5.1.1. DiaSpec: a Domain-Specific Language for Networked Entities

The core of the DIA SUITE development environment is the domain specific language called DIA SPEC and its compiler DIA GEN:

- **DIA SPEC** is composed of two layers:
  - The **Taxonomy Layer** allows the declaration of entities that are relevant to the target application area. An entity consists of sensing capabilities, producing data, and actuating capabilities, providing actions. Accordingly, an entity description declares a data source for each one of its sensing capabilities. As well, an actuating capability corresponds to a set of method declarations. An entity declaration also includes attributes, characterizing properties of entity instances. Entity declarations are organized hierarchically allowing entity classes to inherit attributes, sources and actions. A taxonomy allows separation of concerns in that the expert can focus on the concerns of cataloging area-specific entities. The entity developer is concerned about mapping a taxonomical description into an actual entity, and the application developer concentrates on the application logic.
  - The **Architecture Layer** is based on an architectural pattern commonly used in the pervasive computing domain [24]. It consists of context components fueled by sensing entities. These components process gathered data to make them amenable to the application needs. Context data are then passed to controller components that trigger actions on entities. Using an architecture description enables the key components of an application to be identified, allowing their implementation to evolve with the requirements (e.g., varying light management implementations in a controller component to optimize energy consumption).

- **DIA GEN** is the DIA SPEC compiler that performs both static and runtime verifications over DIA SPEC declarations and produces a dedicated programming framework that guides and eases the implementation of components. The generated framework is independent of the underlying distributed technology. As of today, DIA GEN supports multiple targets: Local, RMI, SIP, Web Services and OSGI.

5.1.2. DiaSim: a Parametrized Simulator for Pervasive Computing Applications
Pervasive computing applications involve both software and integration concerns. This situation is problematic for testing pervasive computing applications because it requires acquiring, testing and interfacing a variety of software and hardware entities. This process can rapidly become costly and time-consuming when the target environment involves many entities.

To ease the testing of pervasive applications, we are developing a simulator for pervasive computing applications: D\textsc{ia}S\textsc{im}. To cope with widely heterogeneous entities, D\textsc{ia}S\textsc{im} is parameterized with respect to a D\textsc{ia}Spec specification describing a target pervasive computing environment. This description is used to generate with D\textsc{ia}Gen both a programming framework to develop the simulation logic and an emulation layer to execute applications. Furthermore, a simulation renderer is coupled to D\textsc{ia}S\textsc{im} to allow a simulated pervasive system to be visually monitored and debugged. The simulation renderer is illustrated in Figure 2.

5.2. DiaSuiteBox: an Open Service Platform

Participants: Benjamin Bertran [correspondent], Julien Bruneau, Charles Consel, Emilie Balland.

The DiaSuiteBox platform runs an open-ended set of applications leveraging a range of appliances and web services. Our solution consists of a dedicated development environment, a certifying application store, and a lightweight runtime platform. This solution is based on the D\textsc{ia}Suite project.

The DiaSuiteBox platform can be embedded in a small plug-computer. This box can be easily deployed, runs silently, and has a reduced energy consumption. Thanks to the application store and the developer community, the platform is fed by a full offer of new innovative applications. During the submission process, an application is automatically analyzed and checked in order to be certified. The user is ensured of the behavior of its applications are innocuous and correct beside the provided information. This box relies on several technology standards like UPnP, Bluetooth, USB, etc. As shown in Figure 3, this platform can be easily extended by plugging appliances directly on the box or by connecting devices on the local network.

See also the web page http://diabox.inria.fr.

5.3. Pantagruel: a Visual Domain-Specific Language for Ubiquitous Computing

Participants: Ghislain Deffrasnes [correspondent], Julien Mercadal, Charles Consel.
Figure 3. DiaSuiteBox platform architecture

Figure 4. A screenshot of the Pantagruel graphical editor
Pantagruel aims at easing the description of an orchestration logic between networked entities of a pervasive environment. First, the developer defines a taxonomy of entities that compose the environment. This step provides an abstraction of the entities capabilities and functionalities. Second, the developer defines the orchestration logic in terms of rules. To facilitate its programming, we provide a visual domain-specific language based on the sensor-controller-actuator paradigm. An example of a visual orchestration is given in Figure 4 where a shower automatically runs at the right temperature when someone enters the bathroom and closes the door.

Pantagruel brings a high-level layer intended to complement existing tools in the activity of safe orchestration logic description, allowing novice-programmers to prototype pervasive applications. The Pantragruel compiler generates code compliant with the D1ASUITE toolset. Pantagruel is being completed by tools aimed at verifying safety properties like termination and reachability.

See also the web page http://phoenix.inria.fr/software/pantagruel.
POPS Project-Team

4. Software

4.1. ASPIRE TDT

Participants: Nathalie MITTON, Loic Schmidt [correspondant], David Simplot-Ryl.

Tag Data Translation (TDT) is an EPCGlobal Inc. standard allowing the translation of identifiers EPC in different representation standard. EPCGlobal standards deal only EPC identifiers. We have extended it to other RFID GS1 and smartcard standards (as ISO 14443 or 15693 and EAN/UPC).

See also the web page http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/TDT .

- Version: version 0.5

4.2. ASPIRE ALECC

Participants: Nathalie MITTON [correspondant], David Simplot-Ryl, Lei Zhang.

According to the feedback of several RFID application SMEs. They are more likely to accept a light and efficient ALE scheme which only includes the most-used basic modules defined by EPC standard. They desire that such light scheme can be encapsulated and be flexibly used to establish their own RFID application.

The AspireALECC scheme is encapsulated in jar and aims to supply an easy and efficient framework for developers to realize the most used basic operations defined by the EPC ALECC standard.

- Version: 1.0

4.3. EPC TAG CONVERTER

Participants: Roudy Dagher [correspondant], Nathalie MITTON, Loic Schmidt, David Simplot-Ryl.

This module is an EPC-compliant module that aims to convert any tag format into an EPC tag understandable by the middleware.

- Version: 1.0

4.4. EPC TAG GENERATOR

Participants: Roudy Dagher [correspondant], Nathalie MITTON, Loic Schmidt, David Simplot-Ryl.

This module aims to generate tag ids in hexadecimal format in order to validate the EPC grouping and filtering engines of the ALE.

See also the web page http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation.EmbeddedTools/TagGenerator .

- Version: 1.0

4.5. EVe - TCF

Participants: Arnaud Fontaine, Isabelle Simplot-Ryl [correspondant].

Verification of transitive control flow policies on JavaCard 2.x bytecode. Control flow policies expressed using a DSL language are embedded in JavaCard packages (CAP files) using EVe-TCF convert tool. Control flow policies are then statically verified on-device at loading-time thanks to an embedded verifier (designed for smart cards in EVe-TCF). EVe-TCF (Embedded Verifier for Transitive Control Flow ) also contains an off-device (i.e. PC tool) to simulate on-device loading process of JavaCard 2.x platforms with GlobalPlatform 2.x installed.

- Version: v2.0 - 07/09/2011
4.6. GOLIATH 1.0

Participants: Tony Ducrocq, Nathalie MITTON, David Simplot-Ryl [correspondant], Julien Vandaele.

GOLIATH (Generic Optimized LIghtweight communication stack for Ambient TecHnologies) is a full protocol stack for wireless sensor networks.

4.7. JITS

Participants: Geoffroy Cogniaux, Gilles GRIMAUD [correspondant].

Initial goal of Java was to allow high level software development on small devices. Eventually it founds success and promotion with software deployment on the Web, and more recently as a solution for huge enterprise servers and massive parallel computing. Today small targets are still supported, but with dedicated (Java-like) APIs and VMs. These specific technologies dramatically restrain the context in which Java applications can be deployed.

JITS focuses on these technologies and on enhancements to allow the use of a real Java Runtime Environment and a Java Virtual Machine everywhere by targeting tiny devices such as SmartCards. These devices usually don’t use a Virtual Machine layer over an OS, but expect the Virtual Machine to be the OS. This is possible thanks to the JVM features which can be presented as a specific hardware abstraction for most of them.

See also the web page http://jits.gforge.inria.fr/

- Version: 1

4.8. Light ALE

Participants: Roudy Dagher [correspondant], Nathalie Mitton, Loic Schmidt, David Simplot-Ryl.

In order to provide minimal inventory services, at interface level, subsets of the Reading and the Logical Reader APIs are implemented:

1. Immediate mode: sufficient for user-triggered inventory.
2. Fixed readers configuration: only some properties (Power, Session and InitialQ) can be updated.

The ALE engine manages tag grouping and filtering according to EPC standard patterns in input ECSpec objects. A lightweight custom CODEC was developed as well, in order to decode tag IDs using binary format (array of bytes) and in a garbage-free fashion. A filter engine is also made available for software filtering of tags. This leaves the choice for Reader Connectors to choose the best tradeoff between software and hardware filtering. Note that, because of the Java CDC constraint, the ECSpec and ECReports classes and subclasses were written manually despite of automatic generation from XSD files.

See also the web page http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation.Filtering%26Collection/EmbededALE

- Version: 1.0

4.9. Light RP

Participants: Roudy Dagher [correspondant], Nathalie Mitton, Loic Schmidt, David Simplot-Ryl.

This wrapper defines the Reader Protocol interface classes that are used to dialog with an RP-compliant reader device. Based on each vendor-specific driver, two implementations were developed in order to provide minimal required services (inventory). Note that the communication with the Reader Device is done locally and directly via method calls. This avoids overhead when using MTB layers for message bindings.

See also the web page http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/LightRP

- Version: 1.0

4.10. NFC Light ALE

Participants: Nathalie Mitton [correspondant], Loic Schmidt, David Simplot-Ryl, Lei Zhang.
In order to provide minimal inventory services, at interface level, subsets of the Reading and the Logical Reader APIs are implemented: (1) Immediate mode: sufficient for user-triggered inventory. ii Fixed readers configuration: only some properties (Power, Session and InitialQ) can be updated.

The ALE engine manages tag grouping and filtering according to EPC standard patterns in input ECSpec objects. A lightweight custom CODEC was developed as well, in order to decode tag IDs using binary format (array of bytes) and in a garbage-free fashion. A filter engine is also made available for software filtering of tags. This leaves the choice for Reader Connectors to choose the best tradeoff between software and hardware filtering. Note that, because of the Java CDC constraint, the ECSpec and ECReports classes and subclasses were written manually despite of automatic generation from XSD files. This package has been developed for NFC connection on a mobile phone.

- Version: 1.0

4.11. RFID Tag Searcher

Participants: Roudy Dagher [correspondant], Nathalie Mitton, Loic Schmidt, David Simplot-Ryl.

The objective is to use the PDA to look for an item in a given neighborhood. The user would be notified of item proximity via the change of the beep frequency.

Tag Searcher is composed of three main modules:

- The Reader interface is an abstraction of the required services for searching for a tag at a given RF power. The wrappers for CAEN and Intermec readers are straightforward.
- The Ticker class represents a periodic thread that beeps periodically using the standard java call java.awt.Toolkit.getDefaultToolkit().beep(). The ticker’s period is synchronized inside the Period-Semaphore class.
- The Scanner class is able to scan for a tag ID and update the Ticker period according to the RF power at which the tag was observed: observation at a small RF power leads to a small tick period, and vice versa.

After testing on both PDAs (i.e. Psion, Intermec), a standalone application with an IHM was developed. It is able to retrieve a list (from a local file or an URL) so that the user choose the item to search for. See also the web page http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation.EmbeddedTools/TagSearcher.

- Version: 1.0

4.12. SINGLE

Participants: Tony Ducrocq, Nathalie Mitton, David Simplot-Ryl [correspondant].

SINGLE pour Simple IN-door Geo-Localization systEm est une application pour réseaux de capteurs permettant la localisation géographique de capteurs sans fils dans un environnement intérieur.

See also the web page http://www.senslab.info/.

- Version: 1.0
5. Software

5.1. Coccinelle

**Participants:** Julia Lawall [correspondent], Gilles Muller [correspondent], Gaël Thomas, Suman Saha, Arie Middlekoop.

Coccinelle is a program matching and transformation engine which provides the language SmPL (Semantic Patch Language) for specifying desired matches and transformations in C code. Coccinelle was initially targeted towards performing collateral evolutions in Linux. Such evolutions comprise the changes that are needed in client code in response to evolutions in library APIs, and may include modifications such as renaming a function, adding a function argument whose value is somehow context-dependent, and reorganizing a data structure.

Beyond collateral evolutions, Coccinelle has been successfully used for finding and fixing bugs in systems code. One of the main recent results is an extensive study of bugs in Linux 2.6 [51] that has permitted us to demonstrate that the quality of code has been improving over the last six years, even though the code size has more than doubled.


5.2. Telex

**Participants:** Marc Shapiro [correspondent], Lamia Benmouffok, Pierre Sutra, Pierpaolo Cincilla.

Developing write-sharing applications is challenging. Developers must deal with difficult problems such as managing distributed state, disconnection, and conflicts. Telex is an application-independent platform to ease development and to provide guarantees. Telex is guided by application-provided parameters: actions (operations) and constraints (concurrency control statements). Telex takes care of replication and persistence, drives application progress, and ensures that replicas eventually agree on a correct, common state. Telex supports partial replication, i.e., sites only receive operations they are interested in. The main data structure of Telex is a large, replicated, highly dynamic graph; we discuss the engineering trade-offs for such a graph and our solutions. Our novel agreement protocol runs Telex ensures, in the background, that replicas converge to a safe state. We conducted an experimental evaluation of the Telex based on a cooperative calendar application and on benchmarks.

We report on application experience, building a collaborative application for model-oriented software engineering above Telex, in SAC 2011 [50]. Future work includes extending Telex to cloud computing, opportunistic mobile networks, and real-time collaboration, within several ANR projects: PROSE (Section 7.1.5), STREAMS (Section 7.1.4) and ConcoRDaNT (Section 7.1.3).

The code is freely available on http://gforge.inria.fr/ under a BSD license.

5.3. Treedoc

**Participants:** Marc Shapiro [correspondent], Marek Zawirski.

A Commutative Replicated Data Type (CRDT) is one where all concurrent operations commute. The replicas of a CRDT converge automatically, without complex concurrency control. We designed and developed a novel CRDT design for cooperative text editing, called Treedoc. It is designed over a dense identifier space based on a binary trees. Treedoc also includes an innovative garbage collection algorithm based on tree rebalancing.

In the best case, Treedoc incurs no overhead with respect to a linear text buffer. The implementation has been validated with performance measurements, based on real traces of social text editing in Wikipedia and SVN.
Work in 2010 has focused on studying large-scale garbage collection for Treedoc, and design improvements. Future work includes engineering a large-scale collaborative Wiki, and studying CRDTs more generally. This is the subject the PROSE, STREAMS and ConcoRDanT ANR projects (Sections 7.1.5, 7.1.4 and 7.1.3 respectively).

The code is freely available on [http://gforge.inria.fr/](http://gforge.inria.fr/) under a BSD license.

### 5.4. VMKit and .Net runtimes for LLVM

**Participants:** Harris Bakiras, Bertil Folliot [correspondent], Julia Lawall, Jean-Pierre Lozi, Gaël Thomas [correspondent], Gilles Muller, Thomas Preud’homme.

Many systems research projects now target managed runtime environments (MRE) because they provide better productivity and safety compared to native environments. Still, developing and optimizing an MRE is a tedious task that requires many years of development. Although MREs share some common functionalities, such as a Just In Time Compiler or a Garbage Collector, this opportunity for sharing has not been yet exploited in implementing MREs. We are working on VMKit, a first attempt to build a common substrate that eases the development and experimentation of high-level MREs and systems mechanisms. VMKit has been successfully used to build two MREs, a Java Virtual Machine and a Common Language Runtime, as well as a new system mechanism that provides better security in the context of service-oriented architectures.

VMKit project is an implementation of a JVM and a CLI Virtual Machines (Microsoft .NET is an implementation of the CLI) using the LLVM compiler framework and the MMTk garbage collectors. The JVM, called J3, executes real-world applications such as Tomcat, Felix or Eclipse and the DaCapo benchmark. It uses the GNU Classpath project for the base classes. The CLI implementation, called N3, is in early stages but can execute simple applications and the “pnetmark” benchmark. It uses the pnetlib project or Mono as its core library. The VMKit VMs compare in performance with industrial and top open-source VMs on CPU-intensive applications. VMKit is publicly available under the LLVM license.

4. Software

4.1. Moose

Participants: Stéphane Ducasse [correspondant], Usman Bhatti, Andre Hora, Nicolas Anquetil, Cyrille Delaunay, Jannik Laval, Tudor Gîrba [University of Bern].

Web: http://www.moosetechnology.org/

The platform. Moose is a language-independent environment for reverse- and re-engineering complex software systems. Moose provides a set of services including a common meta-model, metrics evaluation and visualization, a model repository, and generic GUI support for querying, browsing and grouping. The development of Moose began at the Software Composition Group in 1997, and is currently contributed to and used by researchers in at least seven European universities. Moose offers an extensible meta-described metamodel, a query engine, a metric engine and several visualizations. Moose is currently in its fourth release and comprises 55,000 lines of code in 700 classes.

The RMoD team is currently the main maintainer of the Moose platform. There are 200 publications (journal, international conferences, PhD theses) based on execution or use of the Moose environment.

The first version running on top of Pharo (Moose 4.0) was released in June 2010. In 2011, Moose saw five releases, with Moose 4.6 in beta since October 2011.

Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation .

- (A5) Audience : 5 – Moose is used by several research groups, a consulting company, and some companies using it in ad-hoc ways.
- (SO4) Software originality : 4 – Moose aggregates the last results of the teams that use it.
- (SM3) Software Maturity : 3 – Moose is developed since 1996 and got two main redesign phases.
- (EM4) Evolution and Maintenance : 4 – Moose will be used as a foundation of our start up so its maintenance is planned.
- (SDL4) Software Distribution and Licensing : 4 – BSD
- (OC) Own Contribution : (Design/Architecture)DA-4, (Coding/Debugging)-4, (Maintenance/Support)-4, (Team/Project Management)-3

4.2. Pharo

Participants: Stéphane Ducasse, Marcus Denker [correspondant], Damien Pollet, Mariano Martinez-Peck, Veronica Uquillas-Gomez, Igor Stasenko.

Web: http://www.pharo-project.org/

The platform. Pharo is a new open-source Smalltalk-inspired language and environment. It provides a platform for innovative development both in industry and research. By providing a stable and small core system, excellent developer tools, and maintained releases, Pharo’s goal is to be a platform to build and deploy mission critical Smalltalk applications.

The first stable version, Pharo 1.0, was released in 2010. The development of Pharo accelerated in 2011: Version 1.2 and 1.3 have been released, the development branch (1.4a) has seen already over 230 incremental releases as of mid November 2011. For 1.2 and 1.3, over 1000 bug tracker issues have been resolved. In 2011, the community organized five Pharo Sprints, RMoD organized the Deep into Smalltalk School in March 2011.
RMoD is the main maintainer and coordinator of Pharo. It is used widely in both research and industry. With Inria, RMoD is in the process of setting up a Pharo Consortium. There are 25 companies interested in supporting the consortium.

Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- (A5) Audience: 5 – Used in many universities for teaching, more than 25 companies.
- (SO3) Software originality: 3 – Pharo offers a classical basis for some aspects (UI). It includes new frameworks and concepts compared to other implementations Smalltalk.
- (SM4) Software Maturity: 4 – Bug tracker, continuous integration, large test suite are on place.
- (EM4) Evolution and Maintenance: 4 – Active user group, consortium is being set up.
- (SDL4) Software Distribution and Licensing: 4 – Pharo is licensed under MIT.
- (OC5) Own Contribution: (Design/Architecture) DA-5, (Coding/Debugging) CD-5, (Maintenance/Support) MS-5, (Team/Project Management) TPM-5

4.3. Coral

Participants: Damien Pollet [correspondant], Camillo Bruni.

Web: http://rmod.lille.inria.fr/coral.

Coral extends the standard Pharo image, to integrate it into the host operating system shell environment and define system commands in Pharo. In term it will provide facilities for image preparation, configuration and deployment.

4.4. VerveineJ

Participants: Nicolas Anquetil [correspondant], Andre Hora.

Web: Inria project https://gforge.inria.fr/projects/verveinej/.

VerveineJ is a tool to export Java projects into the MSE format, which can then be imported inside Moose (see above). Although VerveineJ is not a research project in itself, it is an important building block for our research in that it allows us to run the Moose platform on legacy Java projects. Another similar tool, Infusion, already existed to fulfill the same needs, but it was closed sources and presented some errors that tainted the results we could obtain.

4.5. VerveineSharp

Participant: Usman Bhatti [correspondant].

Web: Inria project https://gforge.inria.fr/projects/verveinesharp/.

Similar to VerveineJ (see above), VerveineSharp is a tool to export C# projects into the MSE format, which can then be imported inside Moose. The reasons for creating this project are the same as for VerveineJ: it is an important building block for our research in that it allows us to run the Moose platform on legacy C# projects. Because C# is a proprietary platform, there are no other tools that can give us the same functionality.
SARDES Project-Team

4. Software

4.1. AAC_tactics

Participants: Thomas Braibant, Damien Pous [correspondant].

AAC_tactics is a plugin for the Coq proof-assistant that implements new proof tactics for rewriting modulo associativity and commutativity. It is available at http://sardes.inrialpes.fr/~braibant/aac_tactics and as part of the Coq distribution.

- ACM: D.2.4 Software/Program Verification
- Keywords: Rewriting, rewriting modulo AC, proof tactics, proof assistant
- Software benefit: AAC_tactics provides novel efficient proof tactics for rewriting modulo associativity and commutativity.
- License: LGPL
- Type of human computer interaction: N/A
- OS/Middleware: Windows, Linux, MacOS X
- Programming language: Coq

4.2. ATBR

Participants: Thomas Braibant, Damien Pous [correspondant].

ATBR (Algebraic Tools for Binary Relations) is library for the Coq proof assistant that implements new proof tactics for reasoning with binary relations. Its main tactics implements a decision procedure for inequalities in Kleene algebras. It is available at http://sardes.inrialpes.fr/~braibant/atbr and as part of the Coq distribution contributed modules.

- ACM: D.2.4 Software/Program Verification
- Keywords: Binary relations, Kleene algebras, proof tactics, proof assistant
- Software benefit: ATBR provides new proof tactics for reasoning with binary relations.
- License: LGPL
- Type of human computer interaction: N/A
- OS/Middleware: Windows, Linux, MacOS X
- Programming language: Coq

4.3. MoKa

Participant: Sara Bouchenak [correspondant].
MoKA is a software framework for the modeling and capacity planning of distributed systems. It first provides a set of tools to build analytical models that describe the behavior of distributed computing systems, in terms of performance, availability, cost. The framework allows to include several model algorithms and to compare them regarding their accuracy and their efficiency. Furthermore, MoKA provides a set of tools to build capacity planning methods. A capacity planning method allows to find a distributed system configuration that guarantee given quality-of-service objectives. MoKA is able to include different capacity planning algorithms and to compare them regarding their efficiency and the optimality of their results. MoKA is available at: [http://sardes.inrialpes.fr/research/moka](http://sardes.inrialpes.fr/research/moka).

- ACM: C.2.4 Distributed Systems, C.4 Performance of Systems, D.2.9 Management
- Keywords: Caching, multi-tier systems, consistency, performance
- Software benefit: a novel end-to-end caching protocol for multi-tier services.
- License: TBD
- Type of human computer interaction: command-line interface
- OS/Middleware: Windows, Linux, MacOS X
- Programming language: Java

### 4.4. ConSer

**Participant**: Sara Bouchenak [correspondant].

ConSER is a software framework for the modeling and the concurrency and admission control of servers systems. It implements a fluid-based model that exhibits the dynamics and behavior of a server system in terms of service performance and availability. ConSER implements various novel admission control laws for servers such as AM-C, PM-C, AA-PM-C and PA-AM-C. A control law produces the server concurrency level that allows to trade-off and meet given service level objectives. ConSER’s modeling and control laws algorithms are implemented following a proxy-based approach for more transparency.

- ACM: C.4 Performance of Systems; D.2.9 Management
- Keywords: System management, capacity planning, performance management
- Software benefit: MoKa provides modeling, capacity planning and performance management facilities for application server clusters. Thanks to its model-based capacity planning, MoKa is able to enforce service level objectives while minimizing the service cost.
- License: LGPL
- Type of human computer interaction: web interface
- OS/Middleware: Windows, Linux, MacOS X
- Programming language: Java, AspectJ

### 4.5. e-Caching

**Participants**: Dàmian Serrano, Sara Bouchenak [correspondant].

E-CACHING is a software framework for higher scalability of multi-tier Internet services through end-to-end caching of dynamic data. It provides a novel caching solution that allows to cache different types of data (e.g. Web content, database query results, etc.), at different locations of multi-tier Internet services. The framework allows to combine different caches and, thus, to provide higher scalability of Internet services. E-CACHING maintains the integrity of the cached data through novel distributed caching algorithms that guarantee the consistency of the underlying data.

- ACM: C.2.4 Distributed Systems, C.4 Performance of Systems
- Keywords: Caching, multi-tier systems, consistency, performance
- Software benefit: a novel end-to-end caching protocol for multi-tier services, consistency management, performance improvement.
- License: TBD
- Type of human computer interaction: command-line interface
- OS/Middleware: Windows, Linux, MacOS X
- Programming language: Java
4.6. MRB

**Participants:** Amit Sangroya, Dàmian Serrano, Sara Bouchenak [correspondant].

MRB is a software framework for benchmarking the performance and dependability of MapReduce distributed systems. It includes five benchmarks covering several application domains and a wide range of execution scenarios such as data-intensive vs. compute-intensive applications, or batch applications vs. interactive applications. MRB allows to characterize application workload, faultload and dataload, and it produces extensive performance and dependability statistics.

- ACM: C.2.4 Distributed Systems, C.4 Performance of Systems
- Keywords: Benchmark, performance, dependability, MapReduce, Hadoop, Cloud Computing
- Software benefit: the first performance and dependability benchmark suite for MapReduce systems.
- License: TBD
- Type of human computer interaction: GUI and command-line interface
- OS/Middleware: Windows, Linux, MacOS X
- Programming language: Java, Unix Shell scripts

4.7. BZR

**Participants:** Eric Rutten [correspondant], Gwenaël Delaval [POP ART team].

BZR is a reactive language, belonging to the synchronous languages family, whose main feature is to include discrete controller synthesis within its compilation. It is equipped with a behavioral contract mechanisms, where assumptions can be described, as well as an "enforce" property part: the semantics of the latter is that the property should be enforced by controlling the behaviour of the node equipped with the contract. This property will be enforced by an automatically built controller, which will act on free controllable variables given by the programmer.

BZR is now further developed with the Pop-Art team, where G. Delaval got a position. It has been designed and developed in the Sardes team in relation with the research topic on Model-based Control of Adaptive and Reconfigurable Systems. It is currently applied in different directions: component-based design and the Fractal framework; real-time control systems and the Orccad design environment; operating systems and administration loops in virtual machines; hardware and reconfigurable architecture (FPGAs).

See also the web page [http://bzr.inria.fr](http://bzr.inria.fr).

- ACM: D.3.3 [Programming Languages]: Language Constructs and Features—Control structures; C.3 [Special-purpose and Application-based Systems]: Real-time and embedded systems; D.2.2 [Software Engineering]: Design Tools and Techniques—Computer-aided software engineering, State diagrams; D.2.4 [Software Engineering]: Software / Program Verification—Formal methods, Programming by contract
- Keywords: Discrete controller synthesis, modularity, components, contracts, reactive systems, synchronous programming, adaptive and reconfigurable systems
- Software benefit: the first integration of discrete control synthesis in a compiler, making it usable at the level of the programming language.
- License: TBD
- Type of human computer interaction: programming language and command-line interface
- OS/Middleware: Linux
- Programming language: Caml; generates C or Java or Caml executable code
5. Software

5.1. QualiPSo Factory: Next Generation Forge

Participants: Gérald Oster [contact], Jérôme Blanchard, Christophe Bouthier.

The QualiPSo Factory ¹ is a next generation forge based on Service Oriented Architecture developed within the QualiPSo european project ². Forges transform foreigners into collaborators, sometimes into developers. Forges are online services that allow instantiation, composition and management of collaborative services. Traditionally, provided collaborative services are version control systems, issue trackers, forums, mailing lists or wikis. In the framework of the european QualiPSo project, we are designing and implementing the next generation of forges. The QualiPSo factory framework aims to ease collaboration between forge users and integration of new collaborative services by developers. Our proposal relies on a software oriented architecture (SOA) and thereby allows composition of services. The current architecture provides core services such as security, notification, indexation, composition and naming which are externalized to other collaborative services. The Factory has been delivered as an outcome of the Qualipso project. Its future needs to be clarified.

¹ http://qualipso.gforge.inria.fr/
² http://www.qualipso.org
TRISKELL Project-Team

5. Software

5.1. Kermeta

Participants: Didier Vojtisek [correspondant], Olivier Barais, Cédric Bouhours, Xavier Dolques, Jacques Falcou, François Fouquet, Marie Gouyette, Jean-Marc Jézéquel, Hajanirina Johary Rambelontsalama.

Nowadays, object-oriented meta-languages such as MOF (meta-object Facility) are increasingly used to specify domain-specific languages in the model-driven engineering community. However, these meta-languages focus on structural specifications and have no built-in support for specifications of operational semantics. Integrated with the industrial standard Ecore and aligned with the OMG standard EMOF 2.0, the Kermeta language consists in a extension to these meta languages to support behavior definition. The language adds precise action specifications with static type checking and genericity at the meta level. Based on object-orientation and aspect orientation concepts, the Kermeta language adds model specific concepts. It is used in several use cases:

- to give a precise semantic of the behavior of a metamodel which then can be simulated.
- to act as a model transformation language.
- to act as a constraint language.

The development environment built for the Kermeta language provides an integrated workbench based on Eclipse. It offers services such as : model execution, text editor (with syntax highlighting, code autocompletion), additional views and various import/export transformations.

Thanks to Kermeta it is possible to build various frameworks dedicated to domain specific metamodels. Those frameworks are organised into MDKs (Model Development Kits). For example, Triskell proposes MDKs to work with the metamodels such as Java5, UML2, RDL (requirements), Ecore, Traceability,...

In 2011, Kermeta tooling has been refactored into a version 2.0.x in order to ease the integration of various MOF related languages in the tool chain. This new version also focuses on a fully compiled mode that allows to deploy kermeta programs in production environments.

See also the web page http://www.kermeta.org .

- APP: IDDN.FR.001.420009.000.S.P.2005.000.10400
- Version: 2.0.1
- Programming language: Java, Scala, Kermeta

5.2. Kevoree

Participants: Olivier Barais [correspondant], François Fouquet, Erwan Daubert, Johann Bourcier, Gregory Nain, Noël Plouzeau.

The Kevoree project defines a framework dedicated to distributed systems design, using the models at runtime paradigm and a component-based software architecture approach. This framework offers a high-level abstraction for managing components and their interactions. It also provides concepts to describe the underlying infrastructure: resources, logical nodes and their topology.

Kevoree also provides a set of tools to manipulate model abstraction easily, relying in part on a Domain Specific Language (DSL) called KevScript. This DSL makes the architecture model modifications easier. Our DSL can also be used in a reasoning engine to dynamically adapt the running system by applying some changes at different level (SaaS, PaaS and IaaS). Kevoree has several runtime platform implementations, allowing execution of applications on various devices (e.g. JavaSeE, Android, µController such as Arduino, and cloud virtual nodes).

1http://kevoree.org
5.3. Pramana

Participants: Benoit Baudry, Juan-Jose Cadavid Gomez, Benoit Combemale, Xavier Dolques, Hajanirina Johary Rambelontsalama, Didier Vojtisek [correspondant].

Pramana is an open-source tool, which automatically generates valid instances of a metamodel. These instances can then be used for analysis, verification, simulation or validation of the metamodel. The core mechanism for model generation relies on the bounded constraint-solver of Alloy, a lightweight model checker developed at the MIT. Alloy is integrated in Kermeta to allow the generation of instances of Ecore or Kermeta metamodels. Pramana implements this integration through a series of transformations and analysis, all implemented in Kermeta.

Metamodel instances can be used as input data for model transformation testing, and in particular for the testing of Kermeta code. For this purpose Pramana includes the K-Yeti module implementing a binding between the Kermeta language and the generic testing framework Yeti. This module allows a Kermeta user to automatically run test cases.

See also the web page https://www.irisa.fr/triskell/Softwares/pramana.

- Version: 1.0
- Programming language: Java, Alloy, Kermeta

ALGORILLE Project-Team

5. Software

5.1. parXXL

Participants: Jens Gustedt, Stéphane Vialle.

parXXL is a library for large scale computation and communication that executes fine grained algorithms (computation and communication are of the same order of magnitude) on coarse grained architectures (clusters, grids, mainframes). Historically, parXXL is the result of a collaboration between INRIA and SUPÉLEC. This library fulfills the requirements of our model PRO, i.e., it uses an alternation of computation and communication steps. It realizes an abstraction layer between the algorithm as it was designed and its realization on different architectures and different modes of communication. The current version of this library has been registered at the APP and is available at http://parxxl.gforge.inria.fr/. It integrates a layer for message passing with MPI, a layer for shared memory with POSIX threads, a layer for out-of-core management with file mapping (system call mmap).

All three different realizations of the communication layers are quite efficient. They let us execute programs that are otherwise unchanged within the three different contexts. Usually, they reach the performance of programs that are directly written for a given context. Generally they outperform programs that are executed in a different context than they were written for, such as MPI programs that are executed on a shared memory mainframe, or such as multi-threaded programs that are executed on a distributed shared memory machine.

5.2. Distem

Participants: Tomasz Buchert, Emmanuel Jeanvoine, Lucas Nussbaum, Luc Sarzyniec.

Distem is a distributed systems emulator. In the context of research on Cloud, P2P, High Performance Computing or Grid systems, it can be used to transform an homogeneous cluster (composed of identical nodes) into an experimental platform where nodes have different performance, and are connected together through a complex network topology, thus facilitating the evaluation or benchmarking of applications targeting such environments.

Distem relies on modern Linux features (LXC, cgroups, cpufreq, iptables, traffic control) to steal resources from applications. At the node level, it provides the ability to introduce heterogeneity by splitting a multi-core node into several several virtual nodes of varying number of cores and CPU frequency. At the network level, it allows the user to describe and build virtual network topologies where each link has a given latency, and bandwidth limit.

Distem is controlled through a REST API to ease its integration into experiment scripts, but also provides a Ruby library and a command-line interface.

It has been registered with the APP, and is freely available under the GNU GPL.

More information is available from http://distem.gforge.inria.fr/.

5.3. Wrekavoc

Participants: Jens Gustedt, Lucas Nussbaum, Tomasz Buchert.

Wrekavoc addresses the problem of controlling the heterogeneity of a cluster to provide a configurable environment that allows for reproducible experiments on large sets of configurations using real applications with no emulation of the code. Work on Wrekavoc has stopped: current works are based on the Distem emulator.
5.4. SimGrid

Participants: Pierre-Nicolas Clauss, El Mehdi Fekari, Martin Quinson, Lucas Nussbaum, Cristian Rosa, Christophe Thiéry.

The SimGrid framework aims at being a scientific instrument to the evaluation of algorithmic solutions for large-scale distributed experiments. It is the result of a collaboration with Henri Casanova (Univ. of Hawaii, Manoa) and Arnaud Legrand (MESCAL team, INRIA Grenoble-Rhône-Alpes, France). Simulation is a common answer to the grid specific challenges such as scale and heterogeneity. SimGrid is one of the major simulators in the Grid community.

The main strong point of this is its carefully assessed model validity. To this end, the simulation kernel relies on a blend of analytical models and coarse-grain discrete event simulation. It proves several orders of magnitude faster than usual packet-level simulators used in the networking community (such as ns2 or GTNetS) while providing a good level of accuracy [43].

The SimGrid framework is currently extremely fast. Independent authors demonstrated its superior scalability over its main concurrence [36], [38]. In addition to the efficiency of the simulation models, this scalability is ensured by a layered architecture, with a simulation kernel computing the time taken by actions which need to consume resources to complete. Another layer of abstraction introduces the notion of processes and network routing between hosts. On top of this come the user interfaces aiming at providing the syntactic sugar easing the tool usage.

Several such user interfaces exist, ensuring the versatility of the SimGrid framework by adapting to the user goal: MSG helps the study of distributed heuristics. This is the historical interface of SimGrid, and remains the most used one. SMPI is a new interface which allows the simulation of MPI programs designed for multi-processor systems on a single computer [4]. SimDag eases the study of scheduling heuristics for DAGs of (parallel) tasks, which helps the work on parallel task scheduling. GRAS (Grid Reality And Simulation) eases the development of Grid services and infrastructures [39] through a specific interface implemented twice: once on top of the simulator for the comfort of development, and once using regular sockets for live deployments.

SimGrid can be freely downloaded from its web page and its user base is rapidly growing. Over the last decade, it grounded the experimental section of more than hundred scientific publications, not counting the ones being co-authored by members of the development team.

5.5. ORWL

Participant: Jens Gustedt.

ORWL is a reference implementation of the Ordered Read-Write Lock tools as described in [3]. It implements interfaces for locking and data management that easily allow to have an overlap between communication and computation. The main tool here is the introduction of a “handle” on a local or remote resource that can be used to trigger asynchronous prefetching of control and/or data. Also it implements a second layer of abstraction for the seamless programming of iterative tasks. With that layer iterative algorithms can be implemented that have guarantees for equity and deadlock-freeness.

ORWL is a standalone library that works on shared memory and in distributed settings. The implementation is uniquely based on C99 and POSIX. ORWL has already been registered at the APP. Final tests and benchmarks are on the way to ensure the quality of the implementation before it will be made publicly available.

5.6. P99

Participant: Jens Gustedt.

P99 is a toolbox of header files designated to ease programming in C, in particular modern C99. Originally, these macro definitions and tools for programming in C99 have been implemented for ORWL, but now they are separated out into a separate toolbox.
This toolbox allows e.g. the simplified use of variable length argument list for macros and functions, default arguments of functions, compile time code unrolling, scope bound resource management, transparent allocation and initialization. It has been registered at the APP and is available at http://p99.gforge.inria.fr/.
5. Software

5.1. Hubble

**Participants:** Ludovic Courtès [correspondant], Nicolas Bonichon.

Hubble is implemented in Scheme, using GNU Guile version 2. Details of the simulation, such as keeping track of processor occupation and network usage, are taken care of by SimGrid, a toolkit for the simulation of distributed applications in heterogeneous distributed environments.

The input to Hubble is an XML description of the DAG of build tasks. For each task, a build duration and the size in bytes of the build output are specified. For our evaluation purposes, we collected this data on a production system, the http://hydra.nixos.org/ build farm hosted at the Technical University of Delft. The DAG itself is the snapshot of the Nix Package Collection (Nixpkgs) corresponding to this data. Hubble has its own in-memory representation of the DAG in the form of a purely functional data structure.

The Nixpkgs DAG contains fixed-output nodes, i.e., nodes whose output is known in advance and does not require any computation. These nodes are typically downloads of source code from external web sites. The raw data collected on http://hydra.nixos.org/ specifies a non-zero duration for these nodes, which represents the time it took to perform the download. This duration info is irrelevant in our context, since they don’t require any computation, and Hubble views these nodes as instantaneous.

See also the web page http://hubble.gforge.inria.fr/.


5.2. namdP2P

**Participants:** Nicolas Bonichon [correspondant], Olivier Beaumont.

NamdP2P is a distributed implementation of ABF method using NAMD. It is worth noting that NAMD is designed to run on high-end parallel platforms or clusters, but not to run efficiently on instable and distributed platforms.


5.3. Malleable minWCT

**Participants:** Nicolas Bonichon [correspondant], Olivier Beaumont, Lionel Eyraud-Dubois.

This applet considers the scheduling of malleable tasks with bounded amount of processing resources. The goal is to compute schedules that minimize the weighted completion time of tasks. The applet generates all possible greedy schedules for a given instance and displays only the best ones.

This applet illustrates the complexity of finding an optimal order of tasks [31].

See also the web page http://www.labri.fr/perso/bonichon/malleable


5.4. Bedibe

**Participants:** Lionel Eyraud-Dubois [correspondant], Przemyslaw Uznanski.

Bedibe (Benchmarking Distributed Bandwidth Estimation) is a software to compare different models for bandwidth estimation on the Internet, and their associated instantiation algorithms. The goal is to ease the development of new models and algorithms, and the comparison with existing solutions. The development of this software is just starting.
See also the web page http://bedibe.gforge.inria.fr/.

5.5. MineWithRounds

Participants: Sofian Maabout [correspondant], Nicolas Hanusse.

This software extracts the Maximal Frequent Itemsets from a transaction database. It is designed in C++ using OpenMP Library to take full advantage of multicore, multi-cpu machines.

5. Software

5.1. DIET

Participants: Yves Caniou, Eddy Caron [correspondent], Frédéric Desprez, Maurice Djibril Faye, Adrian Muresan, Jonathan Rouzaud-Cornabas.

Huge problems can now be processed over the Internet thanks to Grid and Cloud middleware systems. The use of on-the-shelf applications is needed by scientists of other disciplines. Moreover, the computational power and memory needs of such applications may of course not be met by every workstation. Thus, the RPC paradigm seems to be a good candidate to build Problem Solving Environments on the Grid or Cloud. The aim of the DIET project (http://graal.ens-lyon.fr/DIET) is to develop a set of tools to build computational servers accessible through a GridRPC API.

Moreover, the aim of a middleware system such as DIET is to provide a transparent access to a pool of computational servers. DIET focuses on offering such a service at a very large scale. A client which has a problem to solve should be able to obtain a reference to the server that is best suited for it. DIET is designed to take into account the data location when scheduling jobs. Data are kept as long as possible on (or near to) the computational servers in order to minimize transfer times. This kind of optimization is mandatory when performing job scheduling on a wide-area network. DIET is built upon Server Daemons. The scheduler is scattered across a hierarchy of Local Agents and Master Agents. Applications targeted for the DIET platform are now able to exert a degree of control over the scheduling subsystem via plug-in schedulers [85]. As the applications that are to be deployed on the Grid vary greatly in terms of performance demands, the DIET plug-in scheduler facility permits the application designer to express application needs and features in order that they be taken into account when application tasks are scheduled. These features are invoked at runtime after a user has submitted a service request to the MA, which broadcasts the request to its agent hierarchy. DIET has been validated on several applications. Example of them have been described in Sections 4.3 through 4.5.

5.1.1. DIET Security

We have worked on extending DIET to include security mechanisms. The first work was to provide authentication of users and components within DIET without breaking DIET distributed architecture. Our security mechanism must also be simple to use by the end users but we need a strong authentication. Recently, we have opted for Kerberos as it provided a Single Sign One that eases the security from the user point of view. Moreover, Kerberos provides strong authentication and works with heterogeneous systems. Work in progress is to integrate Kerberos within DIET. First, it will be used to provide traceability of user’s actions and authentication of all DIET inner components. Then, it will be integrated in an authorization mechanism and other higher level security mechanisms.

5.1.2. GridRPC Data Management API

The GridRPC paradigm is an OGF standard, but the API appeared to lack of precision in order to make a GridRPC code portable to any GridRPC compliant middleware. Additionally required data have to be present on the client side (this can involve a potential transfer from where the data is stored onto the client), and transfers must be performed during the GridRPC call, both degrading performance, and can even make a calculus unfeasible.

Thus the GridRPC community has interests in Data Management within the GridRPC paradigm – Because of previous works performed in the DIET middleware concerning Data Management, Eddy Caron is co-chair of the GridRPC working group.
In consequence, we worked on a Data Management API which has been presented to almost all OGF sessions since OGF’21. Since September 2011, the proposal is an OGF standard, published at http://www.ogf.org/documents/GFD.186.pdf under the title “Data Management API within the GridRPC. Y. Caniou and others, via GRIDRPC-WG”. Some work are still in progress, like 1) the implementation of a library and its integration into GridRPC middleware, in order to publish a proof of concept of both realization and collaboration between two different GridRPC middleware supervising different domain platforms, and 2) a specific OGF document describing some parts of implementation to achieve code portability.

5.1.3. Latest Releases

- November 14th 2011, DIET 2.8 release.
- June 16th 2011, DIET 2.7 release.
- March 7th 2011, DIET 2.6.1 release
- January 14th 2011, DIET 2.6 release

5.2. MUMPS

Participants: Maurice Brémond, Guillaume Joslin, Jean-Yves L’Excellent [correspondent], Mohamed Sid-Lakhdar, Bora Uçar.

MUMPS (for MUltifrontal Massively Parallel Solver, see http://graal.ens-lyon.fr/MUMPS ) is a software package for the solution of large sparse systems of linear equations. The development of MUMPS was initiated by the European project PARASOL (Esprit 4, LTR project 20160, 1996-1999), whose results and developments were public domain. Since then, research and developments have been supported by CERFACS, CNRS, ENS Lyon, INPT-ENSEEIHT-IRIT (main contributor), INRIA, and University of Bordeaux. MUMPS implements a direct method, the multifrontal method, and is a parallel code capable of exploiting distributed-memory computers; its main originalities are its performance, its numerical robustness and the wide range of functionalities available.

The latest release is MUMPS 4.10.0 (May 2011). Its main new functionalities concern the determinant, the possibility to compute entries of the inverse of a sparse matrix and an option to discard factors. Some memory and performance improvements have also been obtained thanks to specific users’ testcases. This year, we have also worked on generic tools and scripts for experimentation, validation and performance study.


5.3. HLCMi

Participants: Julien Bigot, Cristian Klein, Christian Pérez [correspondent], Vincent Pichon.

HLCMi is an implementation of the HLCM component model defined during the PhD of Julien Bigot. HLCM is a generic extensible component model with respect to component implementations and interaction concerns. Moreover, HLCM is abstract; it is its specialization—such as HLCM/CCM—that define the primitive elements of the model, such as the primitive components and the primitive interactions.

HLCMi is making use of Model-driven Engineering (MDE) methodology to generate a concrete assembly from a high level description. It is based on the Eclipse Modeling Framework (EMF). HLCMi contains 700 Emfatic lines to describe its models and 7000 JAVA lines for utility and model transformation purposes. HLCMi is a general framework that supports several HLCM specialization: HLCM/CCM, HLCM/JAVA, HLCM/C++ (known as L2C) and HLCM/Charm++ (known as Gluon++).

5.4. BitDew

Participants: Gilles Fedak [correspondent], Haiwu He, Bing Tang, José Francisco Saray Villamizar, Mircea Moca, Lu Lu.
BitDEW is an open source middleware implementing a set of distributed services for large scale data management on Desktop Grids and Clouds. BitDEW relies on five abstractions to manage the data: i) replication indicates how many occurrences of a data should be available at the same time on the network, ii) fault-tolerance controls the policy in presence of hardware failures, iii) lifetime is an attribute absolute or relative to the existence of other data, which decides the life cycle of a data in the system, iv) affinity drives movement of data according to dependency rules, v) protocol gives the runtime environment hints about the protocol to distribute the data (http, ftp or bittorrent). Programmers define for every data these simple criteria, and let the BitDEW runtime environment manage operations of data creation, deletion, movement, replication, and fault-tolerance operation.

The current status of the software is the following: BitDEW is open source under the GPLv3 or Cecill licence at the user’s choice, 10 releases were produced in the last two years, and it has been downloaded approximatively 6000 times on the INRIA forge. Known users are Université Paris-XI, Université Paris-XIII, University of Florida, Cardiff University and University of Sfax. In term of support, the development of BitDew is partly funded by the INRIA ADT BitDew and by the ANR MapReduce projects. Thanks to this support, we have developed and released the first prototype of the MapReduce programming model for Desktop Grids on top of BitDew. In 2011, 5 versions of the software have been released, including the version 1.0.0 considered as the first stable release of BitDew. Our most current work focuses on providing reliable storage on top of hybrid distributed computing infrastructures.

5.5. XtremWeb

Participants: Gilles Fedak [correspondent], Haiwu He, Bing Tang, Simon Delamare.

XtremWeb is an open source software for Desktop Grid computing, jointly developed by INRIA and IN2P3. XtremWeb allows to build lightweight Desktop Grid by gathering the unused resources of Desktop Computers (CPU, storage, network). Its primary features permit multi-users, multi-applications and cross-domains deployments. XtremWeb turns a set of volatile resources spread over LAN or Internet into a runtime environment executing high throughput applications.

XtremWeb is a highly programmable and customizable middleware which supports a wide range of applications (bag-of tasks, master/worker), computing requirements (data/CPU/network-intensive) and computing infrastructures (clusters, Desktop PCs, multi-Lan) in a manageable, scalable and secure fasion. Known users include LIFL, LIP, LIG, LRI (CS), LAL (physics Orsay), IBBMC (biology), Université Paris-XIII, Université de Guadeloupe, IFP (petroleum), EADS, CEA, University of Wisconsin Madison, University of Tsukuba (Japan), AIST (Australia), UCSD (USA), Université de Tunis, AlmerGrid (NL), Fundecyt (Spain), Hobai (China), HUST (China).

There are two branches of XtremWeb: XtremWeb-HEP is a production version developed by IN2P3. It features many security improvements such as X509 support which allows its usage within the EGEE context. XtremWeb-CH is a research version developed by HES-SO, Geneva, which aims at building an effective Peer-To-Peer system for CPU time consuming applications.

XtremWeb has been supported by national grants (ACI CGP2P) and by major European grants around Grid and Desktop Grid such as FP6 CoreGrid: European Network of Excellence, FP6 Grid4all, and more recently FP7 EDGeS : Enabling Desktop Grid for E-Science and FP7 EDGI: European Desktop Grid Initiative.

On going developments include: providing Quality-of-Service for Desktop Grids (SpeQuloS), inclusion of the BitDew middleware to distribute data as well as inclusion of virtualization technologies.
5. Software

5.1. APMC-CA

Participants: Sylvain Peyronnet [correspondant], Joel Falcou, Pierre Esterie, Khaled Hamidouche, Alexandre Borghi.

The APMC model checker implements the state-of-the-art approximate probabilistic model checking methods. Last year we develop a version of the tool dedicated to the CELL architecture. Clearly, it was very pedagogic, but the conclusion is that the CELL is not adapted to sampling based verification methods.

This year we develop, thanks to the BSP++ framework, a version compatible with SPM/multicores machines, clusters and hybrid architectures. This version outperforms all previous ones, thus showing the interest of both these new architectures and of the BSP++ framework.

5.2. YML

Participants: Serge Petiton [correspondant], Nahid Emad, Maxime Hugues.

Scientific end-users face difficulties to program P2P large scale applications using low level languages and middleware. We provide a high level language and a set of tools designed to develop and execute large coarse grain applications on peer-to-peer systems. Thus, we introduced, developed and experimented the YML for parallel programming on P2P architectures. This work was done in collaboration with the PRiSM laboratory (team of Nahid Emad).

The main contribution of YML is its high level language for scientific end-users to develop parallel programs for P2P platforms. This language integrates two different aspects. The first aspect is a component description language. The second aspect allows to link components together. A coordination language called YvetteML can express graphs of components which represent applications for peer-to-peer systems.

Moreover, we designed a framework to take advantage of the YML language. It is based on two component catalogues and an YML engine. The first one concerns end-user’s components and the second one is related to middleware criteria. This separation enhances portability of applications and permits real time optimizations.

Currently we provide support for the XtremWeb Peer-to-Peer middleware and the OmniRPC grid system. The support for Condor is currently under development and a beta-release will be delivered soon (in this release, we plan to propagate semantic data from the end-users to the middleware). The next development of YML concerns the implementation of a multi-backend scheduler. Therefore, YML will be able to schedule at runtime computing tasks to any global computing platform using any of the targeted middleware.

We experimented YML with basic linear algebra methods on a XtremWeb P2P platform deployed between France and Japan. Recently, we have implemented complex iterative restarted Krylov methods, such as Lanczos-Bisection, GMRES and MERAM methods, using YML with the OmniRPC back-end. The experiments are performed either on the Grid5000 testbed of on a Network of Workstations deployed between Lille, Versailles and Tsukuba in Japan. Demos was proposed on these testbeds from conferences in USA. We recently finished evaluations of the overhead generated using YML, without smart schedulers and with extrapolations due to the lack of smart scheduling strategies inside targeted middleware.

In the context of the FP3C project funded by ANR-JST, we have recently extended YML to support a directive distributed parallel language, XcalableMP http://www.xcalablemp.org/ . This extension is based on the support of the XcalableMP language inside YML components. This allows to develop parallel programs with two programming paradigm and thus two parallelism levels. This work is a part of the project that targets post-Petascale supercomputer that would be composed of heterogeneous and massively parallel hardware.

The software is available at http://yml.prism.uvsq.fr/
5.3. The Scientific Programming InterNet (SPIN)

**Participant:** Serge Petiton [correspondant].

SPIN (Scientific Programming on the InterNet), is a scalable, integrated and interactive set of tools for scientific computations on distributed and heterogeneous environments. These tools create a collaborative environment allowing the access to remote resources.

The goal of SPIN is to provide the following advantages: Platform independence, Flexible parameterization, Incremental capacity growth, Portability and interoperability, and Web integration. The need to develop a tool such as SPIN was recognized by the GRID community of the researchers in scientific domains, such as linear algebra. Since the P2P arrives as a new programming paradigm, the end-users need to have such tools. It becomes a real need for the scientific community to make possible the development of scientific applications assembling basic components hiding the architecture and the middleware. Another use of SPIN consists in allowing to build an application from predefined components ("building blocks") existing in the system or developed by the developer. The SPIN users community can collaborate in order to make more and more predefined components available to be shared via the Internet in order to develop new more specialized components or new applications combining existing and new components thanks to the SPIN user interface.

SPIN was launched at ASCI CNRS lab in 1998 and is now developed in collaboration with the University of Versailles, PRiSM lab. SPIN is currently under adaptation to incorporate YML, cf. above. Nevertheless, we study another solution based on the Linear Algebra KErnel (LAKE), developed by the Nahid Emad team at the University of Versailles, which would be an alternative to SPIN as a component oriented integration with YML.

5.4. V-DS

**Participant:** Franck Cappello [correspondant].

This project started officially in September 2004, under the name V-Grid. V-DS stands for Virtualization environment for large-scale Distributed Systems. It is a virtualization software for large scale distributed system emulation. This software allows folding a distributed systems 100 or 1000 times larger than the experimental testbed. V-DS virtualizes distributed systems nodes on PC clusters, providing every virtual node its proper and confined operating system and execution environment. Thus compared to large scale distributed system simulators or emulators (like MicroGrid), V-DS virtualizes and schedules a full software environment for every distributed system node. V-DS research concerns emulation realism and performance.

A first work concerns the definition and implementation of metrics and methodologies to compare the merits of distributed system virtualization tools. Since there is no previous work in this domain, it is important to define what and how to measure in order to qualify a virtualization system relatively to realism and performance. We defined a set of metrics and methodologies in order to evaluate and compared virtualization tools for sequential system. For example a key parameter for the realism is the event timing: in the emulated environment, events should occur with a time consistent with a real environment. An example of key parameter for the performance is the linearity. The performance degradation for every virtual machine should evolve linearly with the increase of the number of virtual machines. We conducted a large set of experiments, comparing several virtualization tools including Vserver, VMware, User Mode Linux, Xen, etc. The result demonstrates that none of them provides both enough isolation and performance. As a consequence, we are currently studying approaches to cope with these limits.

We have made a virtual platform on the GDX cluster with the Vserver virtualization tool. On this platform, we have launched more than 20K virtual machines (VM) with a folding of 100 (100 VM on each physical machine). However, some recent experiments have shown that a too high folding factor may cause a too long execution time because of some problems like swapping. Currently, we are conducting experiments on another platform based on the virtualization tool named Xen which has been strongly improved since 2 years. We expect to get better result with Xen than with Vserver. Recently, we have been using the V-DS version based on Xen to evaluate at large scales three P2P middleware [89].
5.5. PVC: Private Virtual Cluster

**Participant:** Franck Cappello [correspondant].

Current complexity of Grid technologies, the lack of security of Peer-to-Peer systems and the rigidity of VPN technologies make sharing resources belonging to different institutions still technically difficult.

We propose a new approach called "Instant Grid" (IG), which combines various Grid, P2P and VPN approaches, allowing simple deployment of applications over different administration domains. Three main requirements should be fulfilled to make Instant Grids realistic: simple networking configuration (Firewall and NAT), no degradation of resource security, no need to re-implement existing distributed applications.

Private Virtual Cluster, is a low-level middleware that meets Instant Grid requirements. PVC turns dynamically a set of resources belonging to different administration domains into a virtual cluster where existing cluster runtime environments and applications can be run. The major objective of PVC is to establish direct connections between distributed peers. To connect firewall protected nodes in the current implementation, we have integrated three techniques: UPnP, TCP/UDP Hole Punching and a novel technique Traversing-TCP.

One of the major application of PVC is the third generation desktop Grid middleware. Unlike BOINC and XtremWeb (which belong to the second generation of desktop Grid middleware), PVC allows the users to build their Desktop Grid environment and run their favorite batch scheduler, distributed file system, resource monitoring and parallel programming library and runtime software. PVC ensures the connectivity layer and provide a virtual IP network where the user can install and run existing cluster software.

By offering only the connectivity layer, PVC allows to deploy P2P systems with specific applications, like file sharing, distributed computing, distributed storage and archive, video broadcasting, etc.

5.6. OpenWP

**Participant:** Franck Cappello [correspondant].

Distributed applications can be programmed on the Grid using workflow languages, object oriented approaches (Proactive, IBIS, etc), RPC programming environments (Grid-RPC, DIET), component based environments (generally based on Corba) and parallel programming libraries like MPI.

For high performance computing applications, most of the existing codes are programmed in C, Fortran and Java. These codes have 100,000 to millions of lines. Programmers are not inclined to rewrite then in a "non standard" programming language, like UPC, CoArray Fortran or Global Array. Thus environments like MPI and OpenMPI remain popular even if they require hybrid approaches for programming hierarchical computing infrastructures like cluster of multi-processors equipped with multi-core processors.

Programming applications on the Grid add a novel level in the hierarchy by clustering the cluster of multi-processors. The programmer will face strong difficulties in adapting or programming a new application for these runtime infrastructures featuring a deep hierarchy. Directive based parallel and distributed computing is appealing to reduce the programming difficulty by allowing incremental parallelization and distribution. The programmer add directives on a sequential or parallel code and may check for every inserted directive its correction and performance improvement.

We believe that directive based parallel and distributed computing may play a significant role in the next years for programming High performance parallel computers and Grids. We have started the development of OpenWP. OpenWP is a directive based programming environment and runtime allowing expressing workflows to be executed on Grids. OpenWP is compliant with OpenMP and can be used in conjunction with OpenMP or hybrid parallel programs using MPI + OpenMP.
The OpenWP environment consists in a source to source compiler and a runtime. The OpenWP parser, interprets the user directives and extracts functional blocks from the code. These blocks are inserted in a library distributed on all computing nodes. In the original program, the functional blocks are replaced by RPC calls and calls to synchronization. During the execution, the main program launches non blocking RPC calls to functions on remote nodes and synchronize the execution of remote functions based on the synchronization directives inserted by the programmer in the main code. Compared to OpenMP, OpenWP does not consider a shared memory programming approach. Instead, the source to source compiler insert data movements calls in the main code. Since the data set can be large in Grid application, the OpenWP runtime organize the storage of data sets in a distributed way. Moreover, the parameters and results of RPC calls are passed by reference, using a DHT. Thus, during the execution, parameter and result references are stored in the DHT along with the current position of the datasets. When a remote function is called, the DHT is consulted to obtain the position of the parameter data sets in the system. When a remote function terminates its execution, it stores the result data sets and store a reference to the data set in the DHT.

We are evaluating OpenWP from an industrial application (Amibe), used by the European aerospace company EADS. Amibe is the mesher module of jCAE. Amibe generates a mesh from a CAD geometry in three steps. It first creates edges between every patch of the CAD (mesh in one dimension), then generates a surface mesh for every unfolded patch (mesh in two dimensions) and finally adds the third dimension to the mesh by projecting the 2D mesh into the original CAD surfaces. The first and third operation cannot be distributed. However the second step can easily be distributed following a master/worker approach, transferring the mesh1d results to every computing node and launching the distributed execution of the patches.

5.7. Parallel solvers for solving linear systems of equations

Participant: Laura Grigori.

In the last several years, there has been significant research effort in the development of fully parallel direct solvers for computing the solution of large unsymmetric sparse linear systems of equations. In this context, we have designed and implemented a parallel symbolic factorization algorithm, which is suitable for general sparse unsymmetric matrices. The symbolic factorization is one of the steps that is sequential and represents a memory bottleneck. The code is intended to be used with very large matrices when because of the memory usage, the sequential algorithm is not suitable. This code is available in the SuperLU_DIST, a widely used software, developed at UC Berkeley and LBNL by Professor James W. Demmel and Dr. Xiaoye S. Li. The algorithm is presented in [77]. The SuperLU_DIST is available at http://crd.lbl.gov/~xiaoye/SuperLU/.

5.8. OpenScop

Participant: Cédric Bastoul.

OpenScop is an open specification which defines a file format and a set of data structures to represent a static control part (SCoP for short), i.e., a program part that can be represented in the polyhedral model, an algebraic representation of programs used for automatic parallelization and optimization (used, e.g., in GNU GCC, LLVM, IBM XL or Reservoir Labs R-Stream compilers). The goal of OpenScop is to provide a common interface to various polyhedral compilation tools in order to simplify their interaction.

OpenScop provides a single format for tools that may have different purposes (e.g., as different as code generation and data dependence analysis). We could observe that most available polyhedral compilation tools during the last decade were manipulating the same kind of data (polyhedra, affine functions...) and were actually sharing a part of their input (e.g., iteration domains and context concepts are nearly everywhere). We could also observe that those tools may rely on different internal representations, mostly based on one of the major polyhedral libraries (e.g., Polylib, PPL or isl), and this representation may change over time (e.g., when switching to a more convenient polyhedral library). OpenScop aims at providing a stable, unified format that offers a durable guarantee that a tool can use an output or provide an input to another tool without breaking a compilation chain because of some internal changes in one element of this chain. The other promise of

---

1 project page: http://jcae.sourceforge.net
OpenScop is the ability to assemble or replace the basic blocks of a polyhedral compilation framework at no, or at least low engineering cost. The OpenScop Library (licensed under the 3-clause BSD license) has been developed as an example, yet powerful, implementation of the OpenScop specification.

5.9. CALU for multicore architectures

Participant: Laura GRIGORI [correspondant].

The communication avoiding algorithms are implemented in the form of a portable library. In its current form, this library is designed for multicore architectures and uses a hybrid scheduling technique that exploits well the data locality and can adapt to dynamic changes in the machine. The library will be publicly available since February 2012.

See also the web page http://www-rocq.inria.fr/who/Laura.Grigori/COALA2010/coala.html.

- Version: 1.0

5.10. Fast linear system solvers in public domain libraries

Participant: Marc Baboulin [correspondant].

Hybrid multicore+GPU architectures are becoming commonly used systems in high performance computing simulations. In this research, we develop linear algebra solvers where we split the computation over multicore and graphics processors, and use particular techniques to reduce the amount of pivoting and communication between the hybrid components. This results in efficient algorithms that take advantage of each computational unit [12]. Our research in randomized algorithms yields to several contributions to propose public domain libraries PLASMA and MAGMA in the area of fast linear system solvers for general and symmetric indefinite systems. These solvers minimize communication by removing the overhead due to pivoting in LU and LDLT factorization.

See also the web page http://icl.cs.utk.edu/magma/.

5.11. cTuning: Repository and Tools for Collective Characterization and Optimization of Computing Systems

Participant: Grigori Fursin [correspondant].

Designing, porting and optimizing applications for rapidly evolving computing systems is often complex, ad-hoc, repetitive, costly and error prone process due to an enormous number of available design and optimization choices combined with the complex interactions between all components. We attempt to solve this fundamental problem based on collective participation of users combined with empirical tuning and machine learning.

We developed cTuning framework that allows to continuously collect various knowledge about application characterization and optimization in the public repository at cTuning.org. With continuously increasing and systematized knowledge about behavior of computer systems, users should be able to obtain scientifically motivated advices about anomalies in the behavior of their applications and possible solutions to effectively balance performance and power consumption or other important characteristics.

Currently, we use cTuning repository to analyze and learn profitable optimizations for various programs, datasets and architectures using machine learning enabled compiler (MILEPOST GCC). Using collected knowledge, we can quickly suggest better optimizations for a previously unseen programs based on their semantic or dynamic features [7].

We believe that such approach will be vital for developing efficient Exascale computing systems. We are currently developing the new extensible cTuning2 framework for automatic performance and power tuning of HPC applications.

For more information, see the web page http://cTuning.org.
5. Software

5.1. Introduction

We describe in this section the software that we are developing. The first two (MaPHyS and EPSN) will be the main milestones of our project. The other software developments will be conducted in collaboration with academic partners or in collaboration with some industrial partners in the context of their private R&D or production activities. For all these software developments, we will use first the various (very) large parallel platforms available through CERFACS and GENCI in France (CCRT, CINES and IDRIS Computational Centers), and next the high-end parallel platforms that will be available via European and US initiatives or projects such that PRACE.

5.2. MaPHyS

MaPHyS (Massively Parallel Hybrid Solver) is a software package whose prototype was initially developed in the framework of the PhD thesis of Azzam Haidar (CERFACS) and further consolidated thanks to the ANR-CIS Solstice funding. This parallel linear solver couples direct and iterative approaches. The underlying idea is to apply to general unstructured linear systems domain decomposition ideas developed for the solution of linear systems arising from PDEs. The interface problem, associated with the so-called Schur complement system, is solved using a block preconditioner with overlap between the blocks that is referred to as Algebraic Additive Schwarz.

In the framework of the INRIA technologic development actions; 24 man-month engineer (Yohan Lee-Tin-Yien) have been allocated to this software activity for the 2009-2011 period. The initial software prototype has been completely redesigned in order to enable us to easily interface any sparse direct solvers and develop new preconditioning techniques. The first public release of the software is planned early 2012. The same software effort has been undertaken for interfacing any graph partitioning tools.

The MaPHyS package is very much a first outcome of the research activity described in Section 3.3. Finally, MaPHyS is a preconditioner that can be used to speed-up the convergence of any Krylov subspace method. We foresee to either embed in MaPHyS some Krylov solvers or to release them as standalone packages, in particular for the block variants that will be some outcome of the studies discussed in Section 3.3.

5.3. EPSN

EPSN (Environment for Computational Steering) is a software environment for the steering of legacy parallel-distributed simulations with simple GUI or more complex (possibly parallel) visualization programs (see Figure 1). In order to make a legacy simulation steerable, the user annotates the source code with the EPSN API. These annotations provide the EPSN environment with two kinds of information: the description of the program structure according to a Hierarchical Task Model (HTM) and the description of the distributed data that will be remotely accessible. EPSN provides a distributed data model, that handles common scientific objects such as parameters, structured grids, particles/atoms and unstructured meshes. It is then possible to dynamically connect EPSN with a client program, that provides a GUI with some visualization & interaction features, as for instance SIMONE (Simulation MONitoring for Epsn). Once a client is connected, it interacts with the simulation via EPSN API. It is possible: 1) to control the execution flow of the remote simulation; 2) to access/modify its data on the fly; and 3) finally to invoke advanced user-defined routines in the simulation. The current version of EPSN is fully based on CORBA for communication on heterogeneous system and VTK/Paraview for visualization. A new release of EPSN, that will be fully based on MPI to handle efficient communication, is currently under development. A prototype is already working.
Figure 1. EPSN: software environment for $M \times N$ computational steering.
EPSN has been supported by the ACI-GRID program (grant number PPL02-03), the ARC RedGRID, the ANR MASSIM (grant number ANR-05-MMSA-0008-03) and the ANR CIS NOSSI (2007). More informations are available on our web site: http://www.labri.fr/projet/epsn. This software is publicy available at Inria Gforge (http://epsn.gforge.inria.fr).

5.4. MPICPL

MPICPL (MPI CouPLing) is a software library dedicated to the coupling of parallel legacy codes, that are based on the well-known MPI standard. It proposes a lightweight and comprehensive programing interface that simplifies the coupling of several MPI codes (2, 3 or more). MPICPL facilitates the deployment of these codes thanks to the mpicplrun tool and it interconnects them automatically through standard MPI inter-communicators. Moreover, it generates the universe communicator, that merges the world communicators of all coupled-codes. The coupling infrastructure is described by a simple XML file, that is just loaded by the mpicplrun tool. Future releases will incorporate new features for checkpoint/restart and dynamic parallel code connection.

MPICPL was developed by the Inria HiePACS project-team for the purpose of the ANR CIS NOSSI. It uses advanced features of MPI2 standard. The framework is publicy available at Inria Gforge: http://mpicpl.gforge.inria.fr.

5.5. MONIQA

MONIQA (MONitoring graphic user Interface for Qm/mm Applications) is a GUI specially designed for the monitoring & steering of the QM/MM application in the ANR CIS NOSSI project. It is based on Tulip, a graph visualization software http://tulip.labri.fr), used to display atoms and molecules. It proposes two working modes: offline or online. The offline mode is mainly used to load input files of DL_POLY & Siesta, and to prepare the quantum region for the QM/MM coupling. In online mode, the end-user can monitor & interact with the running QM/MM application thanks to EPSN. It is thus possible to visualize molecular and physical data (distances, angles, charges, energies), and to change simulation parameters on-the-fly, such as the target temperature of the system, thermo or barostat parameters, verbosity of output, ... MONIQA is based on QT4. It was developed specifically for the ANR NOSSI project and is available (restricted access) at Inria Gforge: http://nossi.gforge.inria.fr.

5.6. ScalFMM

ScalFMM (Parallel Fast Multipole Library for Large Scale Simulations) is a software library to simulate N-body interactions using the Fast Multipole Method. ScalFMM is based on the FMB prototype developed by Pierre Fortin during his PhD thesis.

In the framework of the INRIA technologic development actions; 24 man-month engineer (Bérenger Bramas) have been allocated to this software activity started in January 2011.

ScalFMM intends to offer all the functionalities needed to perform large parallel simulations while enabling an easy customization of the simulation components: kernels, particles and cells. It works in parallel in a shared/distributed memory model using OpenMP and MPI. The software architecture has been designed with two major objectives: being easy to maintain and easy to understand. The code is extremely documented and the naming convention fully respected. Driven by its user-oriented philosophy, ScalFMM is using CMAKE as a compiler/installer tool. Even if ScalFMM is written in C++ it will support a C and fortran API soon.

The ScalFMM package is very much a first outcome of the research activity described in Section 3.4.
5.7. Other software

These software packages are or will be developed in collaboration with some academic partners (LIP6, LaBRI, CPMOH, IPREM, EPFL) or in collaboration with industrial partners (CEA, TOTAL, EDF) in the context of their private R&D or production activities.

- For the materials physics applications, a lot of development will be done in the context of ANR projects (NOSSI and proposal OPTIDIS, see Section 4.2 ) in collaboration with LaBRI, CPMOH, IPREM, EPFL and with CEA Saclay and Bruyère-le-Châtel.

- In the context of the PhD thesis of Mathieu Chanaud (collaboration with CEA/CESTA), we have developed a new parallel platform based on a combination of a geometric full multigrid solver and a direct solver (the PaStiX solver developed in the previous ScAlAp1x project-team) to solve huge linear systems arising from Maxwell equations discretized with first-order Nédélec elements (see Section 3.3 ).

- Finally, we contribute to software developments for seismic analysis and imaging and for wave propagation in collaboration with TOTAL (use of GPU technology with CUDA).
5. Software

5.1. BlobSeer

Contact: Gabriel Antoniu, gabriel.antoniu@inria.fr.

Participants from the KerData team: Alexandra Carpen-Amarie, Diana Moise, Viet-Trung Tran, Alexandu Costan, Gabriel Antoniu, Luc Bougé.

Presentation: BlobSeer is the core software platform for most current projects of the KerData team. It is a data storage service specifically designed to deal with the requirements of large-scale data-intensive distributed applications that abstract data as huge sequences of bytes, called BLOBs (Binary Large OBjects). It provides a versatile versioning interface for manipulating BLOBs that enables reading, writing and appending to them.

BlobSeer offers both scalability and performance with respect to a series of issues typically associated with the data-intensive context: scalable aggregation of storage space from the participating nodes with minimal overhead, ability to store huge data objects, efficient fine-grain access to data subsets, high throughput in spite of heavy access concurrency, as well as fault-tolerance.

Users: Work is currently in progress in several formalized projects (see previous section) to integrate and leverage BlobSeer as a data storage back-end in the reference cloud environments: a) Microsoft Azure; b) the Nimbus cloud toolkit developed at Argonne National Lab (USA); and c) in the OpenNebula IaaS cloud environment developed at UCM (Madrid).

URL: http://blobseer.gforge.inria.fr/

License: GNU Lesser General Public License (LGPL) version 3.

Status: This software is available on INRIA’s forge. Version 1.0 (released late 2010) registered with APP: IDDN.FR.001.310009.000.S.P.000.10700.

5.2. Damaris

Contact: Gabriel Antoniu, gabriel.antoniu@inria.fr.

Participants from the KerData team: Matthieu Dorier, Gabriel Antoniu.

Presentation: Damaris is a middleware for multicore SMP nodes enabling them to efficiently handle data transfers for storage and visualization. The key idea is to dedicate one or a few cores of each SMP node to the application I/O. It is developed within the framework of a collaboration between KerData and the Joint Laboratory for Petascale Computing (JLPC). The current version enables efficient asynchronous I/O, hiding all I/O related overheads such as data compression and post-processing. On-going work is targeting fast direct access to the data from running simulations, and efficient I/O scheduling.

Users: Damaris has been preliminarily evaluated at NCSA (Urbana-Champaign) with the CM1 tornado simulation code. CM1 is one of the target applications of the Blue Waters supercomputer developed by at NCSA/UIUC (USA), in the framework of the INRIA/UIUC Joint Lab (JLPC). Work is currently in progress to use Damaris as an intermediate data layer optimizing simulation/visualization coupling for several HPC scientific applications intended to run on the Blue Waters.

URL: http://damaris.gforge.inria.fr/

License: GNU Lesser General Public License (LGPL) version 3.

Status: This software is available on INRIA’s forge. Registration with APP is in progress.
5.3. Derived software

Derived from BlobSeer, two additional platforms are currently being developed within KerData: 1) Pyramid, a software service for array-oriented active storage developed within the framework of the PhD thesis of Viet-Trung Tran (see Section 6.7); and 2) TomusBlobs, a PaaS-level storage service for Azure clouds developed within the framework of the thesis of Radu Tudoran in relation to the A-Brain project. These platforms have not been publicly released yet.
MESCAL Project-Team

5. Software

5.1. Tools for cluster management and software development

Participant: Olivier Richard [correspondant].

The KA-Tools is a software suite developed by MESCAL for exploitation of clusters and grids. It uses a parallelization technique based on spanning trees with a recursive starting of programs on nodes. Industrial collaborations were carried out with Mandrake, BULL, HP and Microsoft.

5.1.1. KA-Deploy

KA-DEPLOY is an environment deployment toolkit that provides automated software installation and reconfiguration mechanisms for large clusters and light grids. The main contribution of KA-DEPLOY 2 toolkit is the introduction of a simple idea, aiming to be a new trend in cluster and grid exploitation: letting users concurrently deploy computing environments tailored exactly to their experimental needs on different sets of nodes. To reach this goal KA-DEPLOY must cooperate with batch schedulers, like OAR, and use a parallel launcher like TAKTUK (see below).

5.1.2. Taktuk

TAKTUK is a tool to launch or deploy efficiently parallel applications on large clusters, and simple grids. Efficiency is obtained thanks to the overlap of all independent steps of the deployment. We have shown that this problem is equivalent to the well known problem of the single message broadcast. The performance gap between the cost of a network communication and of a remote execution call enables us to use a work stealing algorithm to realize a near-optimal schedule of remote execution calls. Currently, a complete rewriting based on a high level language (precisely Perl script language) is under progress. The aim is to provide a light and robust implementation. This development is lead by the MOAIS project-team.

5.2. OAR: Batch scheduler for clusters and grids

Participant: Olivier Richard [correspondant].

The OAR project focuses on robust and highly scalable batch scheduling for clusters and grids. Its main objectives are the validation of grid administration tools such as TAKTUK, the development of new paradigms for grid scheduling and the experimentation of various scheduling algorithms and policies.

The grid development of OAR has already started with the integration of best effort jobs whose purpose is to take advantage of idle times of the resources. Managing such jobs requires a support of the whole system from the highest level (the scheduler has to know which tasks can be canceled) down to the lowest level (the execution layer has to be able to cancel awkward jobs). The OAR architecture is perfectly suited to such developments thanks to its highly modular architecture. Moreover, this development is used for the CiGri grid middleware project.

The OAR system can also be viewed as a platform for the experimentation of new scheduling algorithms. Current developments focus on the integration of theoretical batch scheduling results into the system so that they can be validated experimentally.

See also the web page http://oar.imag.fr .

5.3. FTA: Failure Trace Archive

Participant: Derrick Kondo [correspondant].
With the increasing functionality, scale, and complexity of distributed systems, resource failures are inevitable. While numerous models and algorithms for dealing with failures exist, the lack of public trace data sets and tools has prevented meaningful comparisons. To facilitate the design, validation, and comparison of fault-tolerant models and algorithms, we led the creation of the Failure Trace Archive (FTA), an on-line public repository of availability traces taken from diverse parallel and distributed systems.

While several archives exist, the FTA differs in several respects. First, it defines a standard format that facilitates the use and comparison of traces. Second, the archive contains traces in that format for over 20 diverse systems over a time span of 10 years. Third, it provides a public toolbox for failure trace interpretation, analysis, and modeling. The FTA was released in November 2009. It has received over 11,000 hits since then. The FTA has had national and international impact. Several published works have already cited and benefited from the traces and tools of the FTA. Simulation toolkits for distributed systems, such as SimGrid (CNRS, France) and GridSim (University of Melbourne, Australia), have incorporated the traces to allow for simulations with failures.

5.4. SimGrid: simulation of distributed applications

**Participants:** Arnaud Legrand [correspondant], Lucas Schnorr, Pierre Navarro, Sascha Hunold, Laurent Bobelin.

SimGrid is a toolkit that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. The specific goal of the project is to facilitate research in the area of distributed and parallel application scheduling on distributed computing platforms ranging from simple network of workstations to Computational Grids.

We have released one new major version (3.6) of SimGrid (June 2011) and two minor versions (June and October 2011). These versions include our current work on visualization, analysis of large scale distributed systems, and extremely scalable simulation. See also the web page [http://simgrid.gforge.inria.fr/](http://simgrid.gforge.inria.fr/).

5.5. TRIVA: interactive trace visualization

**Participants:** Lucas Schnorr [correspondant], Arnaud Legrand.

TRIVA is an open-source tool used to analyze traces (in the Pajé format) registered during the execution of parallel applications. The tool serves also as a sandbox for the development of new visualization techniques. Some features include: Temporal integration using dynamic time-intervals; Spatial aggregation through hierarchical traces; Scalable visual analysis with squarified treemaps; A Custom Graph Visualization.

See also the web page [http://triva.gforge.inria.fr/](http://triva.gforge.inria.fr/).

5.6. $\psi$ and $\psi^2$: perfect simulation of Markov Chain stationary distributions

**Participant:** Jean-Marc Vincent [correspondant].

$\psi$ and $\psi^2$ are two software tools implementing perfect simulation of Markov Chain stationary distributions using coupling from the past. $\psi$ starts from the transition kernel to derive the simulation program while $\psi^2$ uses a monotone constructive definition of a Markov chain. They are available at [http://www-id.imag.fr/Logiciels/psi/](http://www-id.imag.fr/Logiciels/psi/).
MOAIS Project-Team

5. Software

5.1. KAAPI

Participants: Thierry Gautier [correspondant], Vincent Danjean, Pierre Neyron.

KAAPI means Kernel for Adaptative, Asynchronous Parallel and Interactive programming. It is a C++ library that allows to execute multithreaded computation with data flow synchronization between threads. The library is able to schedule fine/medium size grain program on distributed machine. The data flow graph is dynamic (unfold at runtime). Target architectures are clusters of SMP machines. Main features are:
- It is based on work-stealing algorithms;
- It can run on various processors;
- It can run on various architectures (clusters or grids);
- It contains non-blocking and scalable algorithms.

See also the web page http://kaapi.gforge.inria.fr.

- ACM: D.1.3
- License: CeCILL
- OS/Middleware: Unix (Linux, MacOSX, ...)
- Programming language: C/C++, Fortran

5.2. OAR

Participants: Pierre Neyron [correspondant MOAIS], Grégory Mounié.

OAR is a batch scheduler developed by Mescal team (correspondant: Olivier Richard). The MOAIS team develops the central automata and the scheduling module that includes successive evolutions and improvements of the policy. OAR is used to schedule jobs both on the CiGri (Grenoble region) and Grid5000 (France) grids. CiGri is a production grid that federates about 500 heterogeneous resources of various Grenoble laboratories to perform computations in physics. MOAIS has also developed the distributed authentication for access to Grid5000.

See also the web page http://oar.imag.fr.

5.3. SOFA

Participant: Bruno Raffin [correspondant].

SOFA is an Open Source framework primarily targeted at real-time simulation, with an emphasis on medical simulation. It is mostly intended for the research community to help develop newer algorithms, but can also be used as an efficient prototyping tool. Based on an advanced software architecture, it allows to:
- create complex and evolving simulations by combining new algorithms with algorithms already included in SOFA;
- modify most parameters of the simulation (deformable behavior, surface representation, solver, constraints, collision algorithm, etc.) by simply editing an xml file;
- build complex models from simpler ones using a scene-graph description;
- efficiently simulate the dynamics of interacting objects using abstract equation solvers;
- reuse and easily compare a variety of available methods.

See also the web page http://www.sofa-framework.org/.

- ACM: J.3
- Programming language: C/C++

5.4. TakTuk - Adaptive large scale remote execution deployment

Participants: Guillaume Huard [correspondant], Pierre Neyron.
TakTuk is a tool for deploying remote execution commands to a potentially large set of remote nodes. It spreads itself using an adaptive algorithm and set up an interconnection network to transport commands and perform I/Os multiplexing/demultiplexing. The TakTuk algorithms dynamically adapt to environment (machine performance and current load, network contention) by using a reactive algorithm that mix local parallelization and work distribution. Characteristics:

- **adaptivity:** efficient work distribution is achieved even on heterogeneous platforms thanks to an adaptive work-stealing algorithm
- **scalability:** TakTuk has been tested to perform large size deployments (hundreds of nodes), either on SMPs, regular clusters or clusters of SMPs
- **portability:** TakTuk is architecture independent (tested on x86, PPC, IA-64) and distinct instances can communicate whatever the machine they’re running on
- **configurability:** mechanics are configurable (deployment window size, timeouts, ...) and TakTuk outputs can be suppressedformatted using I/O templates
- **auto-propagation:** the engine can spread its own code to remote nodes in order to deploy itself
- **communication layer:** nodes successfully deployed are numbered and perl scripts executed by TakTuk can send multicast communications to other nodes using this logical number
- **information redirection:** I/O and commands status are multiplexed from/to the root node. [http://taktuk.gforge.inria.fr](http://taktuk.gforge.inria.fr) under GNU GPL licence.

### 5.5. KRASH - Kernel for Reproduction and Analysis of System Heterogeneity

**Participants:** Guillaume Huard [correspondant], Swann Perarnau.

KRASH is a tool for reproducible generation of system-level CPU load. This tool is intended for use in shared memory machines equipped with multiple CPU cores that are usually exploited concurrently by several users. The objective of KRASH is to enable parallel application developers to validate their resources use strategies on a partially loaded machine by replaying an observed load in concurrence with their application. To reach this objective, KRASH relies on a method for CPU load generation which behaves as realistically as possible: the resulting load is similar to the load that would be produced by concurrent processes run by other users. Nevertheless, contrary to a simple run of a CPU-intensive application, KRASH is not sensitive to system scheduling decisions. The main benefit brought by KRASH is this reproducibility: no matter how many processes are present in the system the load generated by our tool strictly respects a given load profile. This last characteristic proves to be hard to achieve using simple methods because the system scheduler is supposed to share the resources fairly among running processes. [http://krash.ligforge.imag.fr](http://krash.ligforge.imag.fr) under GNU GPL licence.

### 5.6. Cache Control

**Participants:** Guillaume Huard [correspondant], Swann Perarnau.

Cache Control is a Linux kernel module enabling user applications to restrict their memory allocations to a subset of the hardware memory cache. This module reserves and exports available physical memory as virtual devices that can be mmap’d to. It gives to calling processes physical memory using only a subset of the cache (similarly to page coloring). It actually creates cache partitions that can be used simultaneously by a process to control how much cache a data structure can use.
5. Software

5.1. Common Communication Interface

**Participant:** Brice Goglin.

- The Common Communication Interface aims at offering a generic and portable programming interface for a wide range of networking technologies (Ethernet, InfiniBand, ...) and application needs (MPI, storage, low latency UDP, ...).
- CCI is developed in collaboration with the Oak Ridge National Laboratory and several other academics and industrial partners.
- CCI is in early development and currently composed of 19 000 lines of C.
- [http://www.cci-forum.org](http://www.cci-forum.org)

5.2. Hardware Locality

**Participants:** Brice Goglin, Samuel Thibault.

- **Hardware Locality** (HWLOC) is a library and set of tools aiming at discovering and exposing the topology of machines, including processors, cores, threads, shared caches, NUMA memory nodes and I/O devices.
- It builds a widely-portable abstraction of these resources and exposes it to the application so as to help them adapt their behavior to the hardware characteristics.
- HWLOC targets many types of high-performance computing applications [6], from thread scheduling to placement of MPI processes. Most existing MPI implementations, several resource managers and task schedulers already use HWLOC.
- HWLOC is developed in collaboration with the OPEN MPI project. The core development is still mostly performed by Brice GOGLIN and Samuel THIBAULT from the RUNTIME team-project, but many outside contributors are joining the effort, especially from the OPEN MPI and MPICH2 communities.
- HWLOC is composed of 33 000 lines of C.
- [http://runtime.bordeaux.inria.fr/hwloc/](http://runtime.bordeaux.inria.fr/hwloc/)

5.3. KNem

**Participants:** Brice Goglin, Stéphanie Moreaud.

- **KNEM (Kernel Nemesis)** is a Linux kernel module that offers high-performance data transfer between user-space processes.
- KNEM offers a very simple message passing interface that may be used when transferring very large messages within point-to-point or collective MPI operations between processes on the same node.
- Thanks to its kernel-based design, KNEM is able to transfer messages through a single memory copy, much faster than the usual user-space two-copy model.
- KNEM also offers the optional ability to offload memory copies on INTEL I/O AT hardware which improves throughput and reduces CPU consumption and cache pollution.
- KNEM is developed in collaboration with the MPICH2 team at the Argonne National Laboratory and the OPEN MPI project. These partners already released KNEM support as part of their MPI implementations.
- KNEM is composed of 7000 lines of C. Its main contributor is Brice GOGLIN.
- [http://runtime.bordeaux.inria.fr/knem/](http://runtime.bordeaux.inria.fr/knem/)
5.4. Marcel

**Participants:** Olivier Aumage, Yannick Martin, Samuel Thibault.

- **MARCEL** is the two-level thread scheduler (also called N:M scheduler) of the PM² software suite.
- The architecture of **MARCEL** was carefully designed to support a large number of threads and to efficiently exploit hierarchical architectures (e.g. multicore chips, NUMA machines).
- **MARCEL** provides a *seed* construct which can be seen as a precursor of thread. It is only when the time comes to actually run the seed that **MARCEL** attempts to reuse the resources and the context of another, dying thread, significantly saving management costs.
- In addition to a set of original extensions, **MARCEL** provides a POSIX-compliant interface which thus permits to take advantage of it by just recompiling unmodified applications or parallel programming environments (API compatibility), or even by running already-compiled binaries with the Linux NPTL ABI compatibility layer.
- For debugging purpose, a trace of the scheduling events can be recorded and used after execution for generating an animated movie showing a replay of the execution.
- The **MARCEL** thread scheduling library is made of 80 000 lines of code.
- [http://runtime.bordeaux.inria.fr/marcel/](http://runtime.bordeaux.inria.fr/marcel/)
- Marcel has been supported for 2 years (2009-2011) by the INRIA ADT Visimar.

5.5. ForestGOMP

**Participants:** Olivier Aumage, Yannick Martin, Pierre-André Wacrenier.

- **FORESTGOMP** is an OPENMP environment based on both the GNU OPENMP run-time and the **MARCEL** thread library.
- It is designed to schedule efficiently nested sets of threads (derived from nested parallel regions) over hierarchical architectures so as to minimize cache misses and NUMA penalties.
- The **FORESTGOMP** runtime generates nested **MARCEL** bubbles each time an **OPENMP** parallel region is encountered, thereby grouping threads sharing common data.
- Topology-aware scheduling policies implemented by **BUBBLESCHED** can then be used to dynamically map bubbles onto the various levels of the underlying hierarchical architecture.
- **FORESTGOMP** allowed us to validate the **BUBBLESCHED** approach with highly irregular, fine grain, divide-and-conquer parallel applications.
- [http://runtime.bordeaux.inria.fr/forestgomp/](http://runtime.bordeaux.inria.fr/forestgomp/)

5.6. Open-MX

**Participants:** Brice Goglin, Ludovic Stordeur.

- The **OPEN-MX** software stack is a high-performance message passing implementation for any generic ETHERNET interface.
- It was developed within our collaboration with Myricom, Inc. as a part of the move towards the convergence between high-speed interconnects and generic networks.
- **OPEN-MX** exposes the raw ETHERNET performance at the application level through a pure message passing protocol.
- While the goal is similar to the old GAMMA stack [58] or the recent iWarp [57] implementations, **OPEN-MX** relies on generic hardware and drivers and has been designed for message passing.
• **OPEN-MX** is also wire-compatible with Myricom MX protocol and interface so that any application built for MX may run on any machine without Myricom hardware and talk other nodes running with or without the native MX stack.

• **OPEN-MX** is also an interesting framework for studying next-generation hardware features that could help **ETHERNET** hardware become legacy in the context of high-performance computing. Some innovative message-passing-aware stateless abilities, such as multiqueue binding and interrupt coalescing, were designed and evaluated thanks to **OPEN-MX** \[ 23 \], \[ 10 \].

• Brice Goglin is the main contributor to **OPEN-MX**. The software is already composed of more than 45 000 lines of code in the Linux kernel and in user-space.

• [http://open-mx.org/](http://open-mx.org/)

### 5.7. StarPU

**Participants:** Cédric Augonnet, Nicolas Collin, Nathalie Furmento, Cyril Roelandt, Samuel Thibault, Ludovic Courtès.

• **STARPU** permits high performance libraries or compiler environments to exploit heterogeneous multicore machines possibly equipped with GPGPUs or Cell processors.

• **STARPU** offers a unified offloadable task abstraction named codelet. In case a codelet may run on heterogeneous architectures, it is possible to specify one function for each architectures (e.g. one function for CUDA and one function for CPUs).

• **STARPU** takes care to schedule and execute those codelets as efficiently as possible over the entire machine. A high-level data management library enforces memory coherency over the machine: before a codelet starts (e.g. on an accelerator), all its data are transparently made available on the compute resource.

• **STARPU** obtains portable performances by efficiently (and easily) using all computing resources at the same time.

• **STARPU** also takes advantage of the heterogeneous nature of a machine, for instance by using scheduling strategies based on auto-tuned performance models.

• **STARPU** can also leverage existing parallel implementations, by supporting parallel tasks, which can be run concurrently over the machine.

• **STARPU** provides a reduction mode, which permit to further optimize data management when results have to be reduced.

• **STARPU** provides integration in MPI clusters through a lightweight DSM over MPI.

• **STARPU** comes with a plug-in for the GNU Compiler Collection (GCC), which extends languages of the C family with syntactic devices to describe **STARPU**’s main programming concepts in a concise, high-level way.

• [http://runtime.bordeaux.inria.fr/StarPU/](http://runtime.bordeaux.inria.fr/StarPU/)

### 5.8. NewMadeleine

**Participants:** Alexandre Denis, François Trahay, Raymond Namyst.

• **NEWMADELEINE** is communication library for high performance networks, based on a modular architecture using software components.

• The **NEWMADELEINE** optimizing scheduler aims at enabling the use of a much wider range of communication flow optimization techniques such as packet reordering or cross-flow packet aggregation.
• **NEWMADELEINE** targets applications with irregular, mult流 communication schemes such as found in the increasingly common application conglomerates made of multiple programming environments and coupled pieces of code, for instance.

• It is designed to be programmable through the concepts of optimization *strategies*, allowing experimentations with multiple approaches or on multiple issues with regard to processing communication flows, based on basic communication flows operations such as packet merging or reordering.

• The reference software development branch of the **NEWMADELEINE** software consists in 90 000 lines of code. **NEWMADELEINE** is available on various networking technologies: Myrinet, Infini-band, Quadrics and ETHERNET. It is developed and maintained by Alexandre Denis.

• [http://runtime.bordeaux.inria.fr/newmadeleine/](http://runtime.bordeaux.inria.fr/newmadeleine/)

5.9. **PadicoTM**

**Participant:** Alexandre Denis.

• PadicoTM is a high-performance communication framework for grids. It is designed to enable various middleware systems (such as CORBA, MPI, SOAP, JVM, DSM, etc.) to utilize the networking technologies found on grids.

• PadicoTM aims at decoupling middleware systems from the various networking resources to reach transparent portability and flexibility.

• PadicoTM architecture is based on software components. Puk (the PadicoTM micro-kernel) implements a light-weight high-performance component model that is used to build communication stacks.

• PadicoTM component model is now used in **NEWMADELEINE**. It is the cornerstone for networking integration in the projects “LEGO” and “COOP” from the ANR.

• PadicoTM is composed of roughly 60 000 lines of C.

• PadicoTM is registered at the APP under number IDDN.FR.001.260013.000.S.P.2002.000.10000.

• [http://runtime.bordeaux.inria.fr/PadicoTM/](http://runtime.bordeaux.inria.fr/PadicoTM/)

5.10. **MAQAO**

**Participants:** Denis Barthou, Andres Charif-Rubial.

• **MAQAO** is a performance tuning tool for OpenMP parallel applications. It relies on the static analysis of binary codes and the collection of dynamic information (such as memory traces). It provides hints to the user about performance bottlenecks and possible workarounds.

• MAQAO relies on binary codes and inserts probes for instrumentation directly inside the binary. There is no need to recompile. The static/dynamic approach of MAQAO analysis is the main originality of the tool, combining performance model with values collected through instrumentation.

• MAQAO has a static performance model for x86 architecture and Itanium. This model analyzes performance of the predecoder, of the decoder and of the different pipelines of the x86 architecture, in particular for SSE instructions.

• The dynamic collection of data in MAQAO enables the analysis of thread interactions, such as false sharing, amount of data reuse, runtime scheduling policy, ...

• MAQAO is in the project ”ProHMPT” from the ANR. A demo of MAQAO has been made in Jan. 2010 for SME/INRIA days and in Nov. 2010 at SuperComputing, INRIA Booth.

5.11. QIRAL

Participant: Denis Barthou.

- QIRAL is a high level language (expressed through LaTeX) that is used to described Lattice QCD problems. It describes matrix formulations, domain specific properties on preconditionings, and algorithms.
- The compiler chain for QIRAL can combine algorithms and preconditionings, checking validity of the composition automatically. It generates OpenMP parallel code, using libraries, such as BLAS.
- This code is developed in collaboration with other teams participating to the ANR PetaQCD project.

5.12. TreeMatch

Participants: Emmanuel Jeannot, Guillaume Mercier.

- TREEMATCH is a library for performing process placement based on the topology of the machine and the communication pattern of the application.
- TREEMATCH provides a permutation of the processes to the processors/cores in order to minimize the communication cost of the application.
- Important features are: the number of processors can be higher than the number of processes; it assumes that the topology is a tree and does not require valuation of the topology (e.g. communication speed); it implements different placement algorithms that are switched according to the input size.
- TREEMATCH is implemented as a load-balancer in Charm++ and as an tool for performing rank reordering in OpenMPI and MPICH-2 [37]
5. Software

5.1. T3devKit testing toolkit and IPv6 test suites

Participants: Anthony Baire, César Viho.

We have built a toolkit for easing executing tests written in the standardized TTCN-3 test specification language. This toolkit is made of a C++ library together with a highly customizable CoDec generator that allows fast development of external components required to execute a test suite. It also provides a framework for representing and manipulating TTCN-3 events so as to ease the production of test reports. The toolkit addresses issues that are not yet covered by ETSI standards while being fully compatible with the existing standard interfaces: TRI (Test Runtime Interfaces) and TCI (Test Control Interfaces). It has been tested with four TTCN-3 environments (IBM, Elvior, Danet and Go4IT) and on three different platforms (Linux, Windows and Cygwin). It is publicly released under the CeCILL-C License. All these tools with associated test suites (for RIPng, DHCPv6 and examples for DNS) are freely available at http://www.irisa.fr/tipi.

5.2. Interoperability Assessment

Participants: Anthony Baire, Nanxing Chen, César Viho.

In previous works we have developed a software toolkit named T3DevKit for easing the development of TTCN-3 tests. The original tool could only run on POSIX systems with the gcc tool-chain. This year we re-factored the build system using the waf build automation tool, which allowed us to integrate with other operating systems and tool-chains (especially MSVC on Windows). This work allowed us to study interoperability issues between tools in the TTCN-3 standard and a poster was presented at the TTCN-3 User Conference 2011. We also presented an introduction tutorial for T3DevKit at this conference [76]. A method to generate and to execute passive interoperability test suites on recorded traces has been proposed in [40].

5.3. Performance and dependability evaluation

Participants: Gerardo Rubino, Bruno Sericola, Bruno Tuffin.

We develop software tools for the evaluation of two classes of models: Markov models and reliability networks. The main objective is to quantify dependability aspects of the behaviors of the modeled systems, but other aspects of the systems can be handled (performance, performability, vulnerability). The tools are specialized libraries implementing numerical, Monte Carlo and Quasi-Monte Carlo algorithms.

One of these libraries has been developed for the Celar (DGA), and its goal is the evaluation of dependability and vulnerability metrics of wide area communication networks (WANs). The algorithms in this library can also evaluate the sensitivities of the implemented dependability measures with respect to the parameters characterizing the behavior of the components of the networks (nodes, lines).

We are also developing tools with the objective of building Markovian models and to compute bounds of asymptotic metrics such as the asymptotic availability of standard metrics of models in equilibrium, loss probabilities, blocking probabilities, mean backlogs,...). A set of functions designed for dependability analysis is being built under the name DependLib.
5. Software

5.1. SOFAT

Participants: Loïc Hélouët [correspondant], Rouwaida Abdallah.

SOFAT is the acronym for Scenario Oracle and Formal Analysis Toolbox. As this name suggests it is a formal analysis toolbox for scenarios. Scenarios are informal descriptions of behaviors of distributed systems. SOFAT allows the edition and analysis of distributed systems specifications described using Message Sequence Charts, a scenario language standardized by the ITU [Z.120]. The main functionalities proposed by SOFAT are the textual edition of Message Sequence Charts, their graphical visualization, the analysis of their formal properties, and their simulation. The analysis of the formal properties of a Message Sequence Chart specification determines if a description is regular, local choice, or globally cooperative. Satisfaction of these properties allow respectively for model-checking of logical formulae in temporal logic, implementation, or comparison of specifications. All these applications are either undecidable problems or unfeasible if the Message Sequence Chart description does not satisfy the corresponding property. The SOFAT toolbox implements most of the theoretical results obtained on Message Sequence Charts this last decade. It is regularly updated and re-distributed. The purpose of this software is twofold:

- Provide a scenario based specification tool for developers of distributed applications
- Serve as a platform for theoretical results on scenarios and partial orders

SOFAT provides several functionalities, that are: syntactical analysis of scenario descriptions, Formal analysis of scenario properties, Interactive Simulation of scenarios when possible, and diagnosis. This year, SOFAT was extended with code synthesis functionalities, allowing to generate communicating automata, promela code, or rest based web services from HMSCs. A new release of the software is expected before the end of the year.

See also the web page http://www.irisa.fr/distribcom/Prototypes/SOFAT/index.html.

- AMS: Order; lattices; ordered algebraic structures
- APP: IDDN.FR.001.080027.000.S.P.2003.00.10600
- Programming language: Java
5. Software

5.1. WSNet

**Participant:** Guillaume Chelius [correspondant].

WSNet is a wireless sensor network simulator that was designed to offer the following features:

- a modular, flexible and accurate simulation of the radio physical medium;
- support for the simulation of environmental phenomena;
- support for interaction between nodes and their environment (sensor-actuator architecture);
- interconnection with the sensor platform emulator WSim to support the distributed emulation of wireless sensor networks.

WSNet is currently in its second release. The number of WSNet users is still growing and several research works reference the software. Many pointers can be found on the project website. Maintenance and support of the software is handled by the D-NET project but also by several contributors from the CITI laboratory (INSA de Lyon), Orange R&D. The WSNet community is quietly spreading in France as well as abroad.

5.2. WSNet-3

**Participant:** Guillaume Chelius [correspondant].

Driven by the feedback gathered among WSNet users, we have started the development of the third WSNet release. While still private, the project web page is available. The objectives behind this new development is:

- to ease the simulation of new radio architectures / standards : e.g. MIMO schemes, UWB, multi-interfaces system;
- to ease the writing of new modules through the use of **High Level Languages** such as Python or Ruby for the development of protocols, etc;
- to ease the debugging and compilation of results during a simulation.

These developments are handled by a core of developers from different affiliations (INSA de Lyon, Orange R&D, INRIA) lead by the D-NET team.

5.3. Sensor Network Tools: drivers, OS and more

**Participants:** Guillaume Chelius, Eric Fleury [correspondant], Clément Burin des Rosiers, Sandrine Avakian, Guillaume Roche.

As a outcomes of the ANR SensLAB project and the INRIA ADT SensTOOLS and SensAS, several softwares (from low level drivers to OSes) were delivered and made available to the research community. The main goal is to lower the cost of developing/deploying a large scale wireless sensor network application. All software are gathered under the SensLAB web site: [http://www.senslab.info/](http://www.senslab.info/) web page where one can find:

- low C-level drivers to all hardware components;
- ports of the main OS, mainly TinyOS, FreeRTOS and Contiki;
- ports and development of higher level library like routing, localization.
GANG Project-Team (section vide)
5. Software

5.1. RPL P2P

Participants: Emmanuel Baccelli [correspondant], Matthias Philipp.

P2P-RPL is an implementation of draft-ietf-roll-p2p-rpl, providing reactive discovery of point-to-point routes in low power and lossy networks such as wireless sensor networks. The implementation is based on the Contiki operating system. See also the web page http://contiki-p2p-rpl.gforge.inria.fr/.

- Version: 0.4

5.2. OPERA infrastructure

Participants: Cédric Adjih [correspondant], Ichra Amdouni, Pascale Minet, Ridha Soua.

OPERA-infrastructure is the system support code of OPERA, the Optimized Protocol for Energy efficient Routing with node Activity scheduling.

5.3. OPERA perf simul

Participants: Cédric Adjih [correspondant], Ichra Amdouni.

OPERA-perf-simul is a set of tools for simulation and performance evaluation as well as large scale tests of OPERA, the Optimized Protocol for Energy efficient Routing with node Activity scheduling.

5.4. OPERA protocol

Participants: Cédric Adjih [correspondant], Ichra Amdouni, Pascale Minet, Saouene Mahfoudh.

OPERA-protocol is the heart of OPERA, the Optimized Protocol for Energy efficient Routing with node Activity scheduling. It includes EOND a neighborhood discovery protocol, EOSTC a protocol byuiding and maintaining a n energy efficient routing tree and SERENA a node coloring algorithm.

5.5. OPERA validation and tools

Participant: Cédric Adjih [correspondant].

OPERA-validation and tools is a set of tools for validation, debugging, analysis and visualization of OPERA protocol, the Optimized Protocol for Energy efficient Routing with node Activity scheduling. It operates either in a real embedded system or in simulation.
5. Software

5.1. Voip bots

Participants: Mohamed Nassar [contact], Olivier Festor.

VoIPbot is a VoIP security tool created as a demonstrator of how attacks can be launched against VoIP/SIP services and users in a remotely and distributed manner. The environment contains bots that can be remotely managed over an Internet Relay Chat (IRC) channel from a central manager. Our bots are currently able to perform the following tasks:

- send SPAM over IP Telephony (SPIT),
- distributed denial of service through intensive generation of invite messages to a target device,
- active scanning of users through incremental options messages issuance to servers and response analysis,
- cracking through brute-force testing of passwords against an identified user account,
- simple device scanning and fingerprinting,
- target aware device fuzzing.

The tool is developed using the Java programming language. It uses the JAIN-SIP, JMF and PIRCBOT libraries. The tool is distributed under a GPL2 Open Source license. Reports show its use mainly in the testing business so far.

5.2. SecSIP

Participants: Abdelkader Lahmadi [contact], Olivier Festor.

SecSip\(^2\) is developed by the team to defend SIP-based (The Session Initiation Protocol) services from known vulnerabilities. It presents a proactive point of defense between a SIP-based network of devices (servers, proxies, user agents) and the open Internet. Therefore, all SIP traffic is inspected and analyzed against authored Veto specification before it is forwarded to these devices. When initializing, the SecSIP runtime starts loading and parsing authored VeTo blocks to identify different variables, event patterns, operations and actions from each rule. It implements an input and output layer, to capture, inject, send and receive SIP packets from and to the network. Intercepted packets are moved to the SIP Packet parser module. The main function of this module is to extract different fields within a SIP message and trigger events specified within the definition blocks. During each execution cycle when a SIP message arrives, the SecSIP runtime uses a data flow acyclic graph network to find definition matching rules and triggers defined events. The paired events in each operator node are propagated over the graph until a pattern is satisfied. When the pattern is satisfied, the respective rule is fired and the set of actions is executed.

SecSIP is freely available on the Internet and has been demonstrated in various High Security Labs exhibits in 2011.

5.3. NDPMon

Participants: Isabelle Chrisment, Olivier Festor [contact].

\(^2\) [http://secsip.gforge.inria.fr/doku.php](http://secsip.gforge.inria.fr/doku.php)
The Neighbor Discovery Protocol Monitor (NDPMon) is an IPv6 implementation of the well-known ArpWatch tool. NDPMon monitors the pairing between IPv6 and Ethernet addresses (NDP activities: new station, changed Ethernet address, flip flop...). NDPMon also detects attacks on the NDP protocol, as defined in RFC 3756 (bogon, fake Router Advertisements...). New attacks based on the Neighbor Discovery Protocol and Address Auto-configuration (RFC 2461 and RFC 2462) have been identified and integrated in the tool. An XML file describes the default behavior of the network, with the authorized routers and prefixes, and a second XML document containing the neighbors database is used. This second file can be filled during a learning phase. All NDP activities are logged in the syslog utility, and so the attacks, but these ones are also reported by mail to the administrator. Finally, NDPMon can detect stack vulnerabilities, like the assignment of an Ethernet broadcast address on an interface.

NDPMon comes along with a WEB interface acting as a GUI to display the informations gathered by the tool, and give an overview of all alerts and reports. Thanks to color codes, the WEB interface makes possible for the administrator to have an history of what happened on his network and identify quickly problems. All the XML files used or produced by the daemon (neighbor cache, configuration file and alerts list) are translated in HTML via XSL for better readability. A statistic module is also integrated and gives informations about the discovery of the nodes and their type (MAC manufacturer distribution ...).

The software package and its source code is freely distributed under an opensource license (LGPL). It is implemented in C, and is available through a SourceForge project at http://ndpmon.sf.net. An open source community is now established for the tool which has distributions for several Operating Systems (Linux, FreeBSD, OpenBSD, NetBSD and Mac OS X). It is also integrated in FreeBSD ports at http://www.freebsd.org/cgi/cvsweb.cgi/ports/net-mgmt/ndpmon/. Binary distributions are also available for .deb and .rpm based Linux flavors.

5.4. AA4MM

Participants: Laurent Ciarletta, Julien Siebert [main developer].

This work has been undertaken in a joint Phd Thesis between the Madynes and MAIA Teams. Vincent Chevrier (MAIA team, LORIA) has been the co-advisor of this PhD and correspondant for this software.

AA4MM (Agents and Artefacts for Multi-modeling and Multi-simulation) is a framework for coupling existing and heterogeneous models and simulators in order to model and simulate complex systems. This is the first implementation of the AA4MM meta-model proposed in Julien Siebert’s PhD. It is written in Java and relies upon Java Messaging Services (JMS) for its distributed version.

AA4MM can be downloaded at http://www.loria.fr/~siebertj/aa4mm/index.html.

5.5. MASDYNE

Participants: Laurent Ciarletta, Julien Siebert [main developer].

This work is undertaken in a joint Phd Thesis between the Madynes and MAIA Teams. Vincent Chevrier (MAIA team, LORIA) has been the co-advisor of this PhD and correspondant for this software.

Other contributors to this software are: Tom Leclerc (Madynes), Francois Klein, Christophe Torin, Marcel Lamenu, Guillaume Favre and Amir Toly.

MASDYNE (Multi-Agent Simulator of DYnamic Networks usErs) is a multi-agent simulator for modeling and simulating users behaviors in mobile ad hoc network. This software is part of joint work with MAIA team, on modeling and simulation of ubiquitous networks.

It has been notably coupled with a network simulator (JANE : Java Adhoc Network Development Environment) to advanced behavior capabilities to standard network simulations.
MAESTRO Project-Team (section vide)
5. Software

5.1. Grph

Participants: Nathann Cohen, David Coudert, Luc Hogie [correspondant], Aurélien Lancin, Grégory Morel, Issam Tahiri.

Around 20,000 lines of code, developed in Java.

The GRPH project takes over Dipergrafs which was introduced in the activity report of 2010. A drastic change in the model of Dipergrafs justified the name change.

The objective of GRPH is to provide researchers and engineers a suitable graph library for graph algorithms experimentation and network simulation. GRPH is mainly a software library, but it also comes with a set of executable files for user interaction and graph format conversion; as such, it can be used autonomously. Performance and accessibility are the primary targets of the GRPH library. At every stage, it is designed to be efficient in terms of: computation time (use of parallelism, caching, adequate data structures, native code, etc.); memory requirements (use of Java primitives); and portability (it is written in a Java and C). Its model considers mixed graphs composed of (un)directed simple- and hyper-edges. It can handle large dynamic graphs in the order of millions of nodes. GRPH comes with a collection of base graph algorithms which are regularly augmented.

So far, most known users of the GRPH library are part of INRIA and of the FP7 STREP EULER project. GRPH is distributed under the terms of a license defined by its contributors and is available for download. This license allows free usage and access to the source code. See http://www-sop.inria.fr/mascotte/software/grph.

In 2011, GRPH was augmented over Dipergrafs of a number of features suited to its usage within the MASCOTTE research team. These include: addition of numerous graph manipulation methods; introduction of an incidence-list data structure for the representation of graphs; introduction of an adaptive data structure for the representation of sets (based on hash-tables and bit-vectors); integration of implementations of "maximum clique" and "sub-graph isomorphism" algorithms by Christine Solnon (CNRS, INSA Lyon). These sources, written in C, are compiled on-the-fly; integration of implementation of "graph isomorphism" algorithm by Brendan McKay (Australian National University); iteration of implementation of "number of triangles" algorithm by Matthieu Latapy (LIP6); introduction of a bridge to the Mascopt/OpenGVE library; introduction of a bridge to the JUNG library; addition of numerous graph algorithms; introduction of a new layer atop GRPH which allows the representation and manipulation of graph as Java objects, like it is done in other libraries such as Mascopt, Jung, etc.; introduction of an efficient mechanism for the definition of graph properties; addition of graph reporting facilities.

On-going works concern the distributed execution of graph algorithms, a bridge to Sage, and the graphical edition of graphs.

5.2. DRMSim

Participants: David Coudert, Luc Hogie [correspondant], Aurélien Lancin, Nicolas Nisse, Issam Tahiri.

Around 45,000 lines, developed in Java, collaboration between MASCOTTE and researchers in LaBRI (95 % MASCOTTE).
The expansion of the Internet results in a number of issues: BGP (Border Gateway Protocol) starts to show its limits in terms of the number of routing table entries it can manage. More efficient dynamic routing protocols are thus under investigation. However, because deploying under-development routing protocols on the Internet is not practicable at a large-scale, simulation is a necessary step to validate the properties of a newly proposed routing scheme. Unfortunately, the simulation of routing protocols over large networks poses real challenges due to the limited computational capabilities of computers. Existing simulation tools exhibit limitations in terms of the number of nodes they can handle and of the models they propose. This motivated us to conceive and develop DRMSim (Dynamic Routing Model Simulator): a network simulator which addresses the specific problem of large-scale simulations of routing models.

DRMSim relies on a discrete-event simulation engine. It proposes a general routing model which accommodates any network configuration. Aside to this, it includes specific models for Generalized Linear Preference (GLP), and K-chordal network topologies, as well as implementations of routing protocols, including the routing protocol proposed in [99] and lightweight versions of BGP (Border Gateway Protocol).

Recent developments (in 2011) in the DRMSim simulator include the four following elements:

1. The initial framework was composed of a routing model. It now incorporates a system model and a metric model. In addition, the system model now considers the dynamic evolution of the simulated network. This dynamic behavior includes the maintenance operations on the network infrastructure as well as router failures. This model stores the connectivity of routers and links before their failure is simulated. This information is used for the simulation of the recovery procedure. This model takes as its input parameter the distribution of failure probability for both routers and links.

2. The metric model has been fully rewritten and is now geared towards computational performance and flexibility. Taking measures along a discrete-event simulation can be performed in many ways. DRMSim uses a new approach which consists in a metric model listening to the simulation and system models. The user can define its own metrics. Memory and CPU usages depend on which metrics are defined, to which set of routers/links they are applied, how many measures are taken and their computational complexity. It is possible to restrict the model to a small amount of nodes/links by selectors provided as input parameters. At the cost of memory and CPU usage, metrics measures can be stored as time-ordered sequence of values. To reduce the need of resources, a single global measure for each metric can be computed. Finally, metrics can be computed globally on the set of selected entities (links/routers) but also separately for each entity.

3. DRMSim enables the definition of customized simulation scenarios and stateful simulation campaigns. Commonly, a simulation campaign consists in iterating over the set of combinations of parameter values, calling the simulation function for every combination. These combinations cannot be found randomly nor can they be determined using linear functions. Indeed, most of the time there exist correlations between the parameters involved. Also for performance reasons, the end-user will prefer non-linear (most often logarithmic) evolutions for the values of the parameters. The definition of the set of combinations is strongly linked to the simulated system and the time needed to solve it. DRMSim provides a simulation methodology that describes (programmatically) the way a simulation campaign should be conducted.

The duration of a simulation can be as long as several hours (or days). In the context of a simulation campaign where numerous simulations are executed, it is important that re-starting a simulation campaign that was interrupted does not entail the re-computation of already computed results. In order to do this, DRMSim stores on disk every step of the execution of a simulation campaign.

In a simulation campaign, simulation runs are independent (no simulation depends on the result computed by another simulation). Consequently they can be executed in parallel. Because one simulation is most likely to use large amount of memory and to be multi-threaded, parallelizing the simulation campaign on one single computer is a poor parallelization scheme. Instead, we currently work at enabling the remote parallel execution of several simulation runs, with the same distribution framework that is used in the GPH library.

4. Finally, DRMSim manipulates graph abstractions, allowing the user to force the use of a library different from the default one, i.e. GPH.
5.3. Mascopt and openGVE

Participant: Michel Syska [correspondant].

Developed in Java.

MASCPT [98] (MASCOTTE Optimization) is a Java library distributed under the terms of the LGPL license which is dedicated to graph and network processing. MASCPT includes a collection of Java interfaces and classes that implement fundamental data structures and algorithms. The forthcoming public distribution of MASCPT will appear under the name of the openGVE project, MASCPT being one implementation of the bridge graph interface (see http://opengve.inria.fr/bridge-graph-interface/apidocs/fr/inria/opengve/bridge/interfaces/Graph.html). The objective is to allow easy integration of different implementations. The applications already written will not be affected. They will have different choices of internal implementation which may lead to better performances for specific issues such as large graphs processing.

The main objective of MASCPT project is to ease software development in the field of network optimization. Examples of problems include routing, grooming, survivability, and virtual network design. MASCPT helps implementing a solution to such problems by providing a data model of the network and the demands, classes to handle data and ready to use implementations of existing algorithms or linear programs (e.g. shortest paths or integral multicmodity flow).

A key feature of MASCPT is to provide a generic linear programming object interface which allows users to program the same way whether the target solver is IBM ILOG CPLEX, GLPK (GNU Linear Programming Kit) or CLP/CBC (accessed through JNI).

MASCPT has been intensively used in the past within MASCOTTE industrial cooperation programs for experimentation and validation purposes as for example with Alcatel Space Technologies and Orange Labs. Today, the library is used within the framework of the ANR AGAPE to implement FPT algorithms (work done at LIFO).

See also the web page http://www-sop.inria.fr/mascotte/mascopt/.

5.4. Open Simulation Architecture (OSA)

Participants: Olivier Dalle [correspondant], Van Dan Nguyen, Judicaël Ribault.

Developed in Java (80%) and XML, AspectJ, etc. Represent the work of about 8 man/year during the last 6 years.

Component-based modeling has many well-known good properties. One of these properties is the ability to distribute the modeling effort amongst several experts, each having his/her own area of system expertise. Clearly, the less experts have to care about areas of expertise of others, the more efficient they are in modeling sub-systems in their own area. Furthermore, the process of studying complex systems using discrete-event computer simulations involves several areas of non-system expertise, such as discrete-event techniques or experiment planning.

The Open Simulation Architecture (OSA) [97] is designed to enforce a strong separation of the end-user roles and therefore, ensure a successful cooperation of all the experts involved in the process of simulating complex systems.

The OSA architecture is also intended to meet the expectations of a large part of the discrete-event simulation community: it provides an open platform intended to support researchers in a wide range of their simulation activities, and allows the reuse and sharing of system models in the simulation community by means of a flexible and generic component model (Fractal).
Many discrete-event simulators are developed concurrently, but with identical or similar purpose. Another goal of OSA is to favor the reuse and integration of simulation software components and models. To favor reuse, OSA uses a layered approach to combine the modeling, simulation, and related concerns, such as instrumentation or deployment. This ability is demonstrated by the successful integration and reuse of third-party components, such as Scave, the analysis module of Omnet++, or a large number of the James II plugins developed by the University of Rostock. OSA is both a testbed for experimenting new simulation techniques and a tool for real case studies.

OSA is Open Source (LGPL) and is available for download on the INRIA forge server http://osa.gforge.inria.fr/. See also the web page http://osa.inria.fr/.

5.5. SageMath

Participants: Nathann Cohen [correspondant], David Coudert, Leonardo Sampaio.

Developed in Python, Cython, and C++. N. Cohen wrote more than 180 patches and N. Cohen, D. Coudert and L. Sampaio reviewed more than 120 others for inclusion in Sage.

Sagemath is a free open-source mathematics software aiming at becoming an alternative to Maple and Matlab. Initially created by William Stein (Professor of mathematics at Washington University), Sagemath is currently developed by more than 180 contributors around the world (mostly researchers). It has currently more than 200 MB of source code and the graph module consists of 40,000 lines. It was initially of interest for Mascotte because of its large library in Combinatorics and Graph Theory. This year, impressive improvements have been made to this library. In particular, N. Cohen contributed a lot into the following: 1) implementation of a generic interface between Sage and existing (Mixed Integer) Linear Program solvers, 2) implementation of exact algorithms for common Polynomial/NP-Complete graph problems, often through the use of Linear Programs, and 3) improving Sage’s documentation by participating to the writing of a french manual on the use of Sage with 10 other french scientists. New patches are in preparation in the group for possible inclusion in Sage.

Sage’s Graph and Linear Programming libraries are currently used by Mascotte members to test algorithms or compare their performances, as well as to prove/disprove theoretical conjectures and for teaching purposes in the Master IFI, stream UBINET.

5.6. Utilities

5.6.1. Java4unix

Participant: Luc Hogie [correspondant].

More than 5,000 lines, developed in Java.

Java4unix proposes a development and distribution framework which simplifies the use of Java for UNIX software programming/distribution. Until now, Java could hardly be used for the development UNIX applications because invoking Java applications from the UNIX shell must be done through an explicit call to the Java virtual machine and writing simple things in Java often requires long coding. Java4unix aims at filling those two gaps by providing a UNIX installer for java applications, turning them to standard UNIX application and a framework that UNIX programmers may use to manipulate files/text, etc.

Java4unix includes a module which enables the reporting and automatic releasing of Eclipse Java projects. This module was formely separated from Java4unix and was referred to as EPR.

See also the web page http://www-sop.inria.fr/members/Luc.Hogie/java4unix/.

5.6.2. Jalinopt

Participants: Luc Hogie [correspondant], Grégory Morel.
Many mathematical and engineering problems can be expressed as linear programs, and doing so facilitates their resolution. Indeed it is generally more convenient to transform a domain-specific problem into a linear-optimizable one (that can be solved by any solver) rather than writing a complex domain-specific algorithm. In the case of graph theory, problems like flows, minimum vertex cover, maximum stable can be conveniently represented via linear programs.

Jalinopt is a Java toolkit for building and solving linear programs. It consists of a straightforward object-oriented model for linear programs, as well as a bridge to most common solvers, including GLPK and CPLEX. Although Jalinopt is inspired by Mascopt and JavaILP, it provides a significantly different model and an utterly different approach to connecting to the solver. In particular this approach, based in inter-process piping, offers better portability, and the possibility to connect (via SSH) to solvers on remote computers.

See also the web page http://www-sop.inria.fr/members/Luc.Hogie/jalinopt/.

### 5.6.3. JavaFarm

**Participant:** Luc Hogie [correspondant].

More than 1,500 lines, developed in Java.

JavaFarm is a middleware enabling the distribution of Java applications across farms of servers. Its workflow basically enables an application to locally aggregate code and data into an object, called job that will migrate to another computer, where it will be computed. When a job completes, its result is transferred back to the caller. Among other features, JavaFarm supports futures (asynchronous job executions), thereby enabling parallelization of the distributed code. The design objectives of JavaFarm are to make distribution and parallelism as transparent and easy as possible.

See also the web page http://www-sop.inria.fr/members/Luc.Hogie/javafarm/.

### 5.6.4. Mascsim

**Participants:** Luc Hogie [correspondant], Aurélien Lancin, Issam Tahiri.

Around 12,000 lines, developed in Java.

Mascsim is a distributed discrete event simulator whose main target is to be easy to use. Unlike most discrete-event simulators, the researcher who is using Mascsim is required to provide only the bare minimum material needed for the simulation: a model for the system, a set of events describing what is going on in the system, as well as a set of metrics of interest.

The simulation process is then entirely automatized.

See also the web page http://www-sop.inria.fr/mascotte/software/mascsim/.

### 5.6.5. P2PVSim

**Participant:** Remigiusz Modrzejewski [correspondant].

Around 8,000 lines, developed in Python.

P2PVSim is a simple discrete-event simulator created for analyzing theoretical properties of peer-to-peer live video streaming algorithms. Implemented in Python it was designed with clarity and extensibility in mind from the beginning. It is capable of simulating overlays of a few thousands of peers. Multiple control protocols have been implemented. At the same time, a lot of work was put into the performance and scalability aspects of the software. Currently it is meant for simulating overlays of a few thousand peers running multiple control protocols that have been implemented.
PLANETE Project-Team

5. Software

5.1. ns-3

**Participant:** Daniel Camara [correspondant].

ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use. ns-3 includes a solid event-driven simulation core as well as an object framework focused on simulation configuration and event tracing, a set of solid 802.11 MAC and PHY models, an IPv4, UDP, and TCP stack and support for nsc (integration of Linux and BSD TCP/IP network stacks).

See also the web page [http://www.nsnam.org](http://www.nsnam.org).

- Version: ns-3.7
- Keywords: networking event-driven simulation
- License: GPL (GPLv2)
- Type of human computer interaction: programmation C++/python, No GUI
- OS/Middleware: Linux, cygwin, osX
- Required library or software: standard C++ library: GPLv2
- Programming language: C++, python
- Documentation: doxygen

5.2. EphPub

**Participants:** Mohamed Ali Kaafar [correspondant], Claude Castelluccia.

EphPub (Ephemeral Publishing) (previously called EphCom) implements a novel key storage mechanism for time-bounded content, that relies on the caching mechanism of the Domain Name System (DNS). Features of EphPub include: EphPub exploits the fact that DNS servers temporarily cache the response to a recursive DNS query for potential further requests. EphPub provides higher security than Vanish, as it is immune to Sybil attacks. EphPub is easily deployable and does not require any additional infrastructure, such as Distributed Hash Tables. EphPub comes with high usability as it does not require users to install and execute any extra additional software. EphPub lets users define data lifetime with high granularity. We provide EphPub as an Android Application to provide ephemeral exchanged SMS, emails, etc. and as a Firefox or Thunderbird extensions so as to support ephemeral publication of any online document.

For more details about the different software products, see [http://planete.inrialpes.fr/projects/ephemeral-publication/](http://planete.inrialpes.fr/projects/ephemeral-publication/).

- Version: v0.1.2-beta
- ACM: K.4.1
- AMS: 94Axx
- Keywords: Ephemeral communications, Right to Forget, Future Internet Architecture, Privacy
- Software benefit: We provide a Firefox Extension that easily allows users to manage disappearing emails. We also provide a command-line tool to manage disappearing files.
- APP: Under APP deposit internal process
- License: GPL
- Type of human computer interaction: Firefox extension + Unix Console
- OS/Middleware: Firefox under any OS
- Required library or software: Python Ext
- Programming language: Python
- Documentation: No detailed documentation has been released so far. A detailed howto can be consulted however at: [http://code.google.com/p/disappearingdata/source/browse/wiki/EphCOM_Firefox_Extension.wiki?r=77](http://code.google.com/p/disappearingdata/source/browse/wiki/EphCOM_Firefox_Extension.wiki?r=77)
5.3. Username Tester

**Participants:** Claude Castelluccia [correspondant], Mohamed Ali Kaafar, Daniele Perito.

Usernames are ubiquitous on the Internet. Almost every web site uses them to identify its users and, by design, they are unique within each service. In web services that have millions or hundreds of millions of users, it might become difficult to find a username that has not already been taken. For instance, you might have experienced that a specific username you wanted was already taken. This phenomenon drives users to choose increasingly complex and unique usernames.

We built a tool to estimate how unique and linkable usernames are and made it available on this page for you to check. For example, according to our tool, “ladygaga” or “12345678” only carry 24 and 17 bits of entropy, respectively. They are therefore not likely to be unique on the Internet. On the other hand, usernames such as “pdjkwerl” or “yourejerky” carry about 40 bits of entropy and are therefore very good identifiers.

Type your username (for example “zorro1982” or “dan.perito”) to discover how unique it is. This tool can help you to select an username that has low entropy and can’t be used to track you on the Internet.

Alternatively, try typing two usernames separated by a space. The tool will give an estimation on whether the two usernames are linkable. The tool is accessible here: http://planete.inrialpes.fr/projects/how-unique-are-your-usernames/

5.4. DroidMonitor

**Participants:** Claude Castelluccia [correspondant], Mohamed Ali Kaafar, Anasthesia Fedane.

In nowadays world the technological progress evolves very quickly. There are more and more new devices, fully equipped with the latest innovations. The question is: do we adopt our main privacy concerns according to these new technologies as quickly as they grow and become widely available for us?…

We developed a novel tool, private data leakage monitoring tool, DroidMonitor. It aims to serve as an educational tool for regular Android Smartphones users to make them aware of existing privacy threats while they are using Location-Based Services. It can be downloaded here: http://planete.inrialpes.fr/android-privacy/

5.5. NEPI

**Participants:** Thierry Turletti [correspondant], Alina Quereilhac, Claudio Freire.

NEPI stands for Network Experimentation Programming Interface. NEPI implements a new experiment plane used to perform ns-3 simulations, planetlab and emulation experiments, and, more generally, any experimentation tool used for networking research. Its goal is to make it easier for experimenters to describe the network topology and the configuration parameters, to specify trace collection information, to deploy and monitor experiments, and, finally, collect experiment trace data into a central datastore. NEPI is a python API (with an implementation of that API) to perform all the above-mentioned tasks and allows users to access these features through a simple yet powerful graphical user interface called NEF. During the year 2011 we improved the robustness in the experiment control scheme, and we added support for new experimentation environments. We released and registered a second version of the NEPI software (IDDN.FR.001.06003.001.S.A.2010.000.10600). Details on the improvements made can be found in [48]. See also the web page http://nepihome.org.

- Version: 1.0
- ACM: C.2.2, C.2.4
- Keywords: networking experimentation
- License: GPL (2)
- Type of human computer interaction: python library, QT GUI
- OS/Middelware: Linux
- Programming language: python
5.6. Reference implementation for SFA Federation of experimental testbeds

**Participants:** Thierry Parmentelat [correspondant], Baris Metin, Julien Tribino.

We are codevelopping with Princeton University a reference implementation for the Testbed-Federation architecture known as SFA for Slice-based Federation Architecture. During 2011 we have focused on the maturation of the SFA codebase, with several objectives in mind, better interoperability between the PlanetLab world and the EmuLab, a more generic shelter that other testbeds can easily leverage in order to come up with their own SFA-compliant wrapper and support for ‘reservable’ mode, which breaks the usual best-effort PlanetLab model. For more details about this contribution see section

See also the web page [http://planet-lab.eu](http://planet-lab.eu)

- Version: myplc-5.0-rc26
- Keywords: networking testbed virtual machines
- License: Various Open Source Licences
- Type of human computer interaction: Web-UI, XMLRPC-based API, Qt-based graphical client
- OS/Middelware: Linux-Fedora
- Required library or software: Fedora-14 for the infrastructure side; the software comes with a complete software suite for the testbed nodes
- Programming languages: primarily python, C, ocaml
- Documentation: most crucial module plcapi is self-documented using a local format & related tool. See e.g. [https://www.planet-lab.eu/db/doc/PLCAPI.php](https://www.planet-lab.eu/db/doc/PLCAPI.php)
- Codebase: [http://git.onelab.eu](http://git.onelab.eu)

5.7. MultiCast Library Version 3

**Participant:** Vincent Roca [correspondant].

MultiCast Library Version 3 is an implementation of the ALC (Asynchronous Layered Coding) and NORM (NACK-Oriented Reliable Multicast Protocol) content delivery Protocols, and of the FLUTE/ALC file transfer application. This software is an implementation of the large scale content distribution protocols standardized by the RMT (Reliable Multicast Transport) IETF working group and adopted by several standardization organizations, in particular 3GPP for the MBMS (Multimedia Broadcast/Multicast Service), and DVB for the CBMS (Convergence of Broadcast and Mobile Services). Our software is used in operational, commercial environments, essentially in the satellite broadcasting area and for file delivery over the DVB-H system where FLUTE/ALC has become a key component. See [http://planete-bcast.inrialpes.fr/](http://planete-bcast.inrialpes.fr/) for more information.

5.8. OpenFEC.org: because open, free AL-FEC codes and codecs matter

**Participants:** Vincent Roca [correspondant], Jonathan Detchart [engineer], Ferdaouss Mattoussi [PhD student].

The goals of the OpenFEC.org [http://openfec.org](http://openfec.org) are:

- to share IPR-free, open, AL-FEC codes, to share high performance, ready-to-use, open, free, C-language, software codecs and to share versatile and automated performance evaluation environments.

This project can be useful to users who do not want to know the details of AL-FEC schemes but do need to use one of them in the software they are designing, or by users who want to test new codes or new encoding or decoding techniques, and who do know what they are doing and are looking for, or by users who need to do extensive tests for certain AL-FEC schemes in a given use-case, with a well defined channel model.
5.9. BitHoc

**Participants:** Chadi Barakat [correspondant], Thierry Turletti, Amir Krifa.

BitHoc (BitTorrent for wireless ad hoc networks) enables content sharing among spontaneous communities of mobile users using wireless multi-hop connections. It is an open source software developed under the GPLv3 licence. A first version of BitHoc has been made public. We want BitHoc to be the real testbed over which we evaluate our solutions for the support and optimization of file sharing in a mobile wireless environment where the existence of an infrastructure is not needed. The proposed BitHoc architecture includes two principal components: a membership management service and a content sharing service. In its current form it is composed of PDAs and smartphones equipped with WIFI adapters and Windows Mobile 6 operating system.

See also the web page [http://planete.inria.fr/bithoc](http://planete.inria.fr/bithoc)

- **Version:** 1.2
- **Keywords:** Tracker-less BitTorrent for mobile Ad Hoc networks
- **License:** GPL (GPLv3)
- **Type of human computer interaction:** Windows Mobile 6 GUI
- **OS/Middleware:** Windows Mobile 6
- **Programming languages:** C++, C#
- **Documentation:** doxygen

5.10. TICP

**Participant:** Chadi Barakat [correspondant].

TICP is a TCP-friendly reliable transport protocol to collect information from a large number of network entities. The protocol does not impose any constraint on the nature of the collected information: availability of network entities, statistics on hosts and routers, quality of reception in a multicast session, weather monitoring, etc. TICP ensures two main things: (i) the information to collect arrives entirely and correctly to the collector where it is stored and forwarded to upper layers, and (ii) the implosion at the collector and the congestion of the network are avoided by controlling the rate of sending probes. The congestion control part of TICP is designed with the main objective to be friendly with applications using TCP. Experimental results show that TICP can achieve better performance than using parallel TCP connections for the data collection. The code of TICP is available upon request, it is an open source software under the GPLv3 licence.

See also the web page [http://planete.inria.fr/ticp/](http://planete.inria.fr/ticp/)

- **Version:** 1.0
- **Keywords:** Information Collection, Congestion and Error Control
- **License:** GPL (GPLv3)
- **Type of human computer interaction:** XML file
- **OS/Middleware:** Linux/Unix
- **Required library or software:** C/C++ Sockets
- **Programming languages:** C/C++
- **Documentation:** Text
5.11. Experimentation Software

WisMon
WisMon is a Wireless Statistical Monitoring tool that generates real-time statistics from a unified list of packets, which come from possible different probes. This tool fulfills a gap on the wireless experimental field: it provides physical parameters on realtime for evaluation during the experiment, records the data for further processing and builds a single view of the whole wireless communication channel environment. WisMon is available as open source under the Cecill license, at http://planete.inria.fr/software/WisMon/.

WEX Toolbox
The Wireless Experimentation (WEX) Toolbox aims to set up, run and make easier the analysis of wireless experiments. It is a flexible and scalable open-source set of tools that covers all the experimentation steps, from the definition of the experiment scenario to the storage and analysis of results. Sources and binaries of the WEX Toolbox are available under the GPLv2 licence at https://twiki-sop.inria.fr/twiki/bin/view/Projets/Planete/WEXToolkit. WEX Toolbox includes the CrunchXML utility, which aims to make easier the running and the analysis of wireless experimentations. In a nutshell, it implements an efficient synchronization and merging algorithm, which takes XML (or PDML) input trace files generated by multiple probes, and stores only the packets fields that have been marked as relevant by the user in a MySQL database –original pcap traces should be first formatted in XML using wireshark. These operations are done in a smart way to balance the CPU resources between the central server (where the database is created) and the different probes (i.e., PC stations where the capture traces are located). CrunchXML is available under the GNU General Public License v2 at http://twiki-sop.inria.fr/twiki/bin/view/Projets/Planete/CrunchXML.

WiMAX ns-3
This simulation module for the ns-3 network simulator is based on the IEEE 802.16-2004 standard. It implements the PMP topology with TDD mode and aims to provide detailed and standard compliant implementation of the standard, supporting important features including QoS scheduling services, bandwidth management, uplink request/grant scheduling and the OFDM PHY layer. The module is available under the GNU General Public License at http://code.nsnam.org/iamine/ns-3-wimax. It will be included in the official 3.8v release of ns-3.

MonLab
Monitoring Lab is a platform for the emulation and monitoring of traffic in virtual ISP networks. It is supported by the FP7 ECODE project and is available for download at the web page of the tool http://planete.inria.fr/MonLab/ under the terms of the GPL licence. MonLab presents a new approach for the emulation of Internet traffic and for its monitoring across the different routers of the emulated ISP network. In its current version, the traffic is sampled at the packet level in each router of the platform, then monitored at the flow level. We put at the disposal of users real traffic emulation facilities coupled to a set of libraries and tools capable of Cisco NetFlow data export, collection and analysis. Our aim is to enable running and evaluating advanced applications for network wide traffic monitoring and optimization. The development of such applications is out of the scope of this research. We believe that the framework we are proposing can play a significant role in the systematic evaluation and experimentation of these applications’ algorithms. Among the direct candidates figure algorithms for traffic engineering and distributed anomaly detection. Furthermore, methods for placing monitors, sampling traffic, coordinating monitors, and inverting sampling traffic will find in our platform a valuable tool for experimentation.

MobitTrade
MobitTrade is the ns-3 and Android implementation of our solution in [41] for trading content between wireless devices. The application provides a utility driven trading system for efficient content dissemination on top of a disruption tolerant network. While simple tit-for-tat (TFT) mechanisms can force nodes to give one to get one, dealing with the inherent tendency of peers to
take much but give back little, they can quickly lead to deadlocks when some (or most) of interesting content must be somehow fetched across the network. To resolve this, MobiTrade proposes a trading mechanism that allows a node (merchant) to buy, store, and carry content for other nodes (its clients) so that it can later trade it for content it is personally interested in. To exploit this extra degree of freedom, MobiTrade nodes continuously profile the type of content requested and the collaboration level of encountered devices. An appropriate utility function is then used to collect an optimal inventory that maximizes the expected value of stored content for future encounters, matched to the observed mobility patterns, interest patterns, and collaboration levels of encountered nodes. See also http://planete.inria.fr/MobiTrade.
RAP Project-Team (section vide)
5. Software

5.1. CloudWeaver suite

Participants: Paulo Gonçalves, Guilherme Koslovski, Fabienne Anhalt.

The following list of softwares, whose development was initiated at RESO, constitutes the main outcome of the research work delivered by Guilherme Koslovski (PhD, July 2011) [8] and Fabienne Anhalt (PhD, July 2011) [7]. These products are also part of the technological transfer to Lyatiss (headed by Pascale Vicat-Blanc): embedded in the CloudWeaver Suite, they implement the solutions for virtual resources orchestration and infrastructure services.

- VXAlloc Dynamic allocation of virtual resources (Patent INPI:10/01626, 2010, Lyatiss, INRIA, ENS Lyon)
- VXCap Partitioning of complex physical infrastructures (Patent INPI:10/01624, 2010, Lyatiss, INRIA, ENS Lyon)
- HiperNet Automatic configuration of virtual networks, by programming virtual routers and configuring virtual links according to service requirements (APPcode: IDDN.FR.001.260010.000.S.P.2009.000.10700, 2009, Lyatiss, INRIA ENS Lyon)
- VXDL parser Interpretation and XML traduction of virtual infrastructures specifications (APPcode: IDDN.FR.001.260009.000.S.P.2009.000.10800)

Due to non-disclosure agreement between INRIA and Lyatiss, access to these softwares is now subject to patent restriction.

5.2. Queueing-systems

Participant: Thomas Begin.

This tool aims at providing a simple web based interface to promote the use of our proposed solutions to numerically solve classical queueing systems. It currently implements the solution to get the distribution for the number of customers along with customary performance parameters for a queue with multiple servers, general arrivals, exponential services and a possibly finite buffer, (i.e., $Ph/M/c/N$-like queue). The steady-state solution to this queue is based on a simple and stable recurrence [50] and was performed in collaboration with Pr. Brandwajn (UCSC). We will include new features and new models to this tool in the near future. Associated URL is: http://queueing-systems.ens-lyon.fr

5.3. ECOFEN simulation framework

Participants: Anne-Cecile Orgerie, Laurent Lefevre.

The problem when evaluating new network architectures and protocols is that large testbed platforms are really expensive and difficult to manage. That is why we have designed ECOFEN whose user’s entries are the network topology and traffic. Based on configurable measurements of different network component (routers, switches, NICs, etc.), it provides the power consumption of the overall network including the end-hosts as well as the power consumption of each equipment over time. The ECOFEN simulator supports green network leverages such as Adaptive Link Rate and on/off. The aim of ECOFEN is to compute and expose the energy consumed by a network under a given traffic. Firstly based on NS2 and now developed on NS3, this simulator has been made in collaboration with Dino Lopez-Pacheco [29].
4. Software

4.1. Introduction

SWING develops several tools supporting its research like SOCLIB and Wiplan. Moreover, SWING is an active contributor to WSnet (http://wsnet.gforge.inria.fr/) a multi-hop wireless network discrete event simulator. WSnet was created in the ARES team and it is now supported by the D-NET team of INRIA Rhône-Alpes. SWING is one of the most important contributors for the design of protocol libraries in WSnet.

4.2. SOCLIB

Participant: Tanguy Risset [correspondant].

SocLib is a library of simulation models for virtual components (IP cores) for Systems on Chip. Many simulation models are under development, SocLib currently contains simulation models for processors (Mips, ARM), memories and network on chips (Spin and DSpin developed at LIP6 laboratory. SocLib permits to simulate at cycle accurate application running on embedded computing systems such as mobile phones. Swing use this platform to prototype design techniques either for embedded software or for hardware parts of signal processing applications.

See also the web page https://www.soclib.fr/trac/dev/wiki.

4.3. Wiplan

Participants: Jean-Marie Gorce [correspondant], Guillaume Villemaud, Meiling Luo, Dmitry Umansky, Tao Wang.

Wiplan is a software including an Indoor propagation engine and a wireless LAN optimization suite, which has been registered by INSA-Lyon. The heart of this software is the propagation simulation core relying on an original method, MR-FDPF (multi-resolution frequency domain ParFlow). The discrete ParFlow equations are translated in the Fourier domain providing a wide linear system, solved in two steps taking advantage of a multi-resolution approach. The first step computes a cell-based tree structure referred to as the pyramid. In the second phase, a radiating source is simulated, taking advantage of the pre-processed pyramidal structure. Using of a full-space discrete simulator instead of classical ray-tracing techniques is a challenge due to the inherent high computation requests. However, we have shown that the use of a multi-resolution approach allows the main computation load to be restricted to a pre-processing phase. Extensive works have been done to make predictions more realistic. The network planning and optimization suite is based on a multi-criteria model relying on a Tabu solver. The development of the wiplan software is a part of the European project iPLAN (IAPP-FP7 project).

See also the web page http://wiplan.citi.insa-lyon.fr.
TREC Project-Team

5. Software

5.1. Gibbs’ Sampler

**Participant:** Chung Shue Chen.

The work on the self optimization of cellular networks based on Gibbs’ sampler (see Section 6.1.1.3) carried out in the joint laboratory with Alcatel-Lucent, led to the development of a software prototype that was presented by C. S. Chen at the INRIA Alcatel-Lucent joint laboratory seminar in March 2010 and demonstrated at the Alcatel-Lucent Bell Labs Open Days in May 2010. It was also demonstrated in the LINCS opening ceremony in April 2011.

5.2. PS12

**Participant:** Ana Bušić.

The work on perfect sampling (see Section 6.2.3) has been partially implemented in a software tool PS12, in collaboration with MESCAL team [INRIA Grenoble - Rhône-Alpes]; [https://gforge.inria.fr/projects/ps1](https://gforge.inria.fr/projects/ps1).