Fractal Component-Based Software Engineering

Thierry Coupaye[1] and Jean-Bernard Stefani[2]

[1] France Telecom R&D
thierry.coupaye@orange-ftgroup.com
[2] INRIA
Jean-Bernard.Stefani@inrialpes.fr

Abstract. This article is a report on the 5th international workshop devoted to the Fractal component model that took place the 4th of July 2006 in Nantes, France, as an ECOOP workshop. Prior to that, the article provides some background on the Fractal project and previous Fractal workshops for readers who are not familiar with Fractal.

1 Introduction

We are witnessing a tremendous expansion in the use of software in scientific research, industry, administration and more and more in every day life. With the advent of the Internet and more globally the convergence between telecommunications and computing, software has become omnipresent, critical, complex. Time-to-market of services, which rely on system engineering (operating systems, distributed systems, middleware), is becoming a strategic factor in a competitive market in which operation (deployment, administration) costs are much higher than development costs.

In this context, component-based software architectures have naturally emerged as a central focus and reached momentum in different fields of computing because Component-Based Software Engineering (CBSE) is generally recognized as one of the best way to develop, deploy and administrate increasingly complex software with good properties in terms of flexibility, reliability, scalability[3] - not to mention lower development cost and faster time-to-market through software reuse and programmers productivity improvements.

Fractal is an advanced component model and associated on-growing programming and management support devised initially by France Telecom and INRIA since 2001. Most developments are framed by the Fractal project inside the ObjectWeb open source middleware consortium. The Fractal project targets the development of a reflective component technology for the construction of highly adaptable and reconfigurable distributed systems.

[3] As stated in the conclusions of the 7th International Symposium on CBSE (Edinburgh, Scotland, 2004): "Components are a way to impose design constraints that as structural invariants yields some useful properties".

2 The Fractal Ecosystem

2.1 Component Model

The Fractal component model relies on some classical concepts in CBSE: *components* are runtime entities that conforms to the model, *interfaces* are the only interaction points between components that express dependencies between components in terms of *required/client* and *provided/server* interfaces, *bindings* are communication channels between component interfaces that can be primitive, i.e. local to an address space or composite, i.e. made of components and bindings for distribution or security purposes.

Fractal also exhibits more original concepts. A component is the composition of a *membrane* and a *content*. The membrane exercices an *arbitrary reflexive control* over its content (including interception of messages, modification of message parameters, etc.). A membrane is composed of a set of *controllers* that may or may not export control interfaces accessible from outside the considered component. The model is *recursive (hierarchical) with sharing* at arbitrary levels. The recursion stops with base components that have an empty content. Base components encapsulate entities in an underlying programming language. A component can be shared by multiple enclosing components. Finally, the model is programming *language independent* and *open*: everything is optional and extensible[4] in the model, which only defines some "standard" API for controlling bindings between components, the hierarchical structure of a component system or the components life-cycle (creation, start, stop, etc).

The Fractal component model enforces a limited number of very structuring architectural principles. Components are runtime entities conformant to a model and do have to exist at runtime per se for management purposes. There is a clear separation between interfaces and implementations which allow for transparent modifications of implementations without changing the structure of the system. Bindings are programmatically controllable: bindings/dependencies are not "hidden in code" but systematically externalized so as to be manipulated by (external) programs. Fractal systems exhibits a recursive structure with composite components that can overlap, which naturally enforces encapsulation and easily models resource sharing. Components exercise arbitrary reflexive control over their content: each component is a management domain of its own. Altogether, these principles make Fractal systems self-similar (hence the name of the model): architecture is expressed homogeneously at arbitrary level of abstraction in terms of bindings an reflexive containment relationships.

2.2 Implementations

There exist currently 8 implementations (platforms)[5] providing support for Fractal components programming in 8 programming languages:

[4] This openness leads to the need for conformance levels and conformance test suites so as to compare distinct implementations of the model.

[5] Julia, AOKell, ProActive and THINK are available in the ObjectWeb code base. FracNet, FractTalk and Flone are available as open source on specific web sites.

- *Julia* was historically (2002) the first Fractal implementation[6], provided by France Telecom. Since its second version, Julia makes use of AOP-like techniques based on interceptors and controllers built as a composition of mixins. It comes with a library of mixins and interceptors mixed at loadtime (Julia relies very much on loadtime bytecode transformation as the main underlying technique thanks to the ASM Java bytecode Manipulation Framework). The design of Julia cared very much for performance: the goal was to prove that component-based systems were not doomed to be inefficient compared to plain Java. Julia allows for intra-components and inter-components optimizations which altogether exhibit very acceptable performance.
- *THINK* is a C implementation of Fractal, provided by France Telecom and INRIA Sardes, with a growing participation of STMicroelectronics and CEA, geared at operating and especially embedded systems development. Using THINK, OS architects can build OS kernels conforming to any kernel architecture: exo-kernel, micro-kernel... Minimal kernels can be built on bare hardware and basic functions such as scheduler and memory policies can be easily redefined or even not included. This helps achieve speed-ups and low memory footprints over standard general-purpose operating systems. THINK is also suggested for prototyping when using a complete OS would be a too heavy solution. It can also be used when implementing application-specific kernels, especially when targeting small platforms embedding microcontrollers. THINK comes along with KORTEX, a library of already existing system components, implementing various functions (memory management, schedulers, file systems, etc.) on various targets (e.g. ARM, PPC, x86).
- *ProActive* is a distributed and asynchronous implementation of Fractal targetting grid computing, developed by INRIA Oasis with a participation of France Telecom. It is a grid middleware for parallel, distributed, and concurrent computing, also featuring mobility and security in a uniform framework. It mixes the active object paradigm for concurrent programming (objects executing their own asynchronous activity in a thread) and the component paradigm for deployment and management.
- *AOKell* is a Java implementation by INRIA Jacquard and France Telecom similar to Julia, but based on standard AOP technologies (static weaving with AspectJ in AOKell v1 and loadtime weaving with Spoon in AOKell v2) instead of mixins. Also AOKell v2 is the first Fractal implementation that supports component-based membranes: Fractal component controllers can themselves be implemented as Fractal components. AOKell offers similar performance to Julia.
- *FractNet* is a .Net implementation of the Fractal component model developed by the LSR laboratory. It is essentially a port of AOKell on .Net, in which AspectDNG is used as an alternative aspect weaver to AspectJ or Spoon. FractNet provides for Fractal component programming in J#, C#, VB.Net and Cobol.Net languages.

[6] And sometimes considered for this reason as "the reference implementation" in Java.

- *Flone* is a Java implementation of the Fractal component model developed by INRIA Sardes for teaching purposes. Flone is not a full-fledge implementation of Fractal: it offers simplified APIs that globally reduce the openness and use of reflection of the general Fractal model so as to make teaching of component-based programming easier for students.
- *FracTalk* is an experimental SmallTalk implementation of the Fractal component model developed at Ecole des Mines de Douai. FracTalk focuses very much on dynamicity in component-based programming thanks the intrinsic dynamic nature of the SmallTalk language.
- *Plasma* is a C++ experimental implementation of Fractal developed at INRIA Sardes (with a participation of Microsoft Research) dedicated to the construction of self-adaptable multimedia applications.

2.3 Languages & Tools

A large number of R&D activities are being conducted inside the Fractal community around languages and tools, with the overall ambition to provide a complete environment covering the complete component-based software life cycle covering modelling, design, development, deployment and (self-)management. A representative but not exhaustive list of such activities is the following:

- development of formal foundations for the Fractal model, typically by means of calculi, essentially by INRIA Sardes,
- development of basic and higher levels (e.g. transactional) mechanisms for trusted dynamic reconfigurations, by France Telecom, INRIA Sardes and Ecole des Mines de Nantes (EMN),
- support for configuration, development of ADL support and associated tool chain, by INRIA Sardes, Jacquard, France Telecom, ST Micoelectronics,
- support for packaging and deployment, by INRIA Jacquard, Sardes Oasis, IMAG LSR laboratory, ENST Bretagne,
- development of navigation and management tools, by INRIA Jacquard and France Telecom,
- development of architectures that mix components and aspects (AOP), at the component (applicative) level and at the membrane (technical) level, by INRIA, France Telecom, ICS/Charles University Prague,
- development of specification models, languages and associated tools for static and dynamic checking of component behaviour, involving ICS/Charles University Prague, I3S/U. Nice, France Telecom, Valoria/U. Bretagne Sud,
- development of security architectures (access control, authentication, isolation), by France Telecom,
- development of QoS management architectures and mechanisms, for instance in THINK-based embedded systems, by France Telecom, or multimedia services with Plasma, by INRIA Sardes,
- development of semi-formal modelling and design methodologies (UML, MDA), models and tools, by CEA, Charles University Prague and others,
- ...

The most mature among these works are typically incorporated as new modules into the Fractal code base. Examples of such modules are the following:

- Fractal RMI is a set of Fractal components that provide a binding factory to create synchronous distributed bindings between Fractal components (la Java RMI). These components are based on a re-engineering process of the Jonathan framework.
- Fractal ADL (Architecture Description Languages) is a language for defining Fractal configurations (components assemblies) and an associated retargetable parsing tool with different back-ends for instantiating these configurations on different implementations (Julia, AOKell, THINK, etc.). Fractal ADL is a modular (XML modules defined by DTDs) and extensible language to describe components, interfaces, bindings, containment relationships, attributes and types - which is classical for an ADL - but also to describe implementations and especially membrane constructions that are specific to each Fractal implementation, deployment information, behaviour and QoS contracts or any other architectural concern. Fractal ADL can be considered as the favourite entry point to Fractal components programming (its offers a much higher level of abstraction than the bare Fractal APIs) that embeds concepts of the Fractal component model[7].
- FractalGUI is a graphical editor for Fractal component configurations which allows for component design with boxes and arrows. Fractal GUI can import/export Fractal configurations from/to Fractal ADL files.
- FScript is a scripting language used to describe architectural reconfigurations of Fractal components. FScript includes a special notation called FPath (loosely inspired by XPath) to query, i.e. navigate and select elements from Fractal architectures (components, interfaces...) according to some properties (e.g. which components are connected to this particular component? how many components are bound to this particular component?). FPath is used inside FScript to select the elements to reconfigure, but can be used by itself as a query language for Fractal.
- Fractal Explorer is a "graphical" (in fact a multi-textual windows system) management console that allows for navigation, introspection and reconfiguration of running Fractal systems in Java.
- Fractal JMX is a set of Fractal components that allows for automatic, declarative and non-intrusive exposition of Fractal components into JMX servers with filtering and renaming capabilities. Fractal JMX allows administrators to see a Fractal system as if it was a plain Java system instrumented "by hand" for JMX management: Fractal components are mapped to MBeans that are accessible by program or with a JMX console through a JMX server.

[7] It is worth noticing that Fractal ADL is not (yet) a complete component-oriented language (in the Turing sense), hence the need for execution support in host programming languages a.k.a. "implementations".

2.4 Component Library & Real Life Usage

Fractal has essentially been used so far to build middleware and operating system components. The current library of components engineered with Fractal that are currently available inside ObjectWeb include:

- DREAM, a framework (i.e. a set of components) for building different types (group communications, message passing, event-reaction, publish-subscribe) of asynchronous communication systems (management of messages, queues, channels, protocols, multiplexers, routers, etc.)
- GOTM, a framework for building transaction management systems (management of transactions demarcation, distributed commit, concurrency, recovery, resources/contexts, etc.)
- Perseus, a framework for building persistence management systems (management of persistency, caching, concurrency, logging, pools, etc.),
- Speedo, an implementation of the JDO (Java Data Object) standard for persistence of Java objects. Speedo embeds Perseus,
- CLIF, a framework for performance testing, load injection and monitoring (management of blades, probes, injectors, data aggregators, etc.)
- JOnAS, a J2EE compliant application server. JOnAS embeds Speedo (hence Perseus, Fractal, Julia, ASM),
- Petals, an implementation of Java Business Integration (JBI) platform, i.e. an Enterprise Software Bus.

Some of these components that embed Fractal technology are used operationally, for instance JOnAS, Speedo and CLIF by France Telecom: JOnAS is widely used by France Telecom[8] for its service platforms, information systems and networks by more than 100 applications including vocal services including VoIP, enterprise web portals, phone directories, clients management, billing management, salesman management, lines and incidents management.

3 Organization of the Workshop

3.1 History of Fractal workshops

The Fractal CBSE workshop at ECOOP 2006 was the 5th in the series[9].

The first workshop was held in January 2003 as an associated event of an ObjectWeb architecture meeting. The attendance was of about 35 people. 15 talks were given, organized in 5 sessions. The first session was a feedback session about the use of Fractal in Jonathan (a flexible ORB), JORAM (a JMS-compliant MOM) and ProActive (a distributed computing environment based on active objects). The second session was dedicated to Fractal implementation,s

[8] See http://jonas.objectweb.org/success.html for a more comprehensive list of operational usage of JOnAS.

[9] All programs and talks from Fractal CBSE workshops are available on the Fractal project web site at http://fractal.objectweb.org.

namely Julia and THINK. The third sessions was devoted to configuration tools, namely Kilim and Fractal GUI. The fourth session was dedicated to management and deployment, especially JMX management with Fractal JMX and connection with J2EE management and OSGi. The last session presented a conceptual comparison of Fractal and other component models.

The second workshop was held in March 2004 as an associated event of an ObjectWeb architecture meeting and ITEA Osmose project meeting. The attendance was of about 30 people. 10 talks were given, organized in 3 sessions. The first session was dedicated to tutorials on the Fractal model and Java tools (Fractal ADL, Fractal GUI, Fractal Explorer). The second session was dedicated to feedback from practical usage of Fractal in the Dream communication framework, the CLIF framework for load injection and performance evaluation and the GoTM open transaction monitor. The third session was dedicated to work in progress: components for grid computing with Fractal and ProActive, components and aspects, convergence of the Fractal and SOFA component models.

The third workshop was held in June 2005, again as an associated event of an ObjectWeb architecture meeting. The attendance was of about 20 people. It was mostly dedicated to discussions about components and aspects around AOKell (aspect-oriented programming of Fractal component membranes), FAC (Fractal Aspect Components: reification of aspects as components), and "microcontrollers". Another talk was given about the development of a formal and dynamic ADL.

The fourth workshop was held in November 2005 as a satellite of the ACM /IFIP/USENIX Middleware conference. The attendance was of more than 50 people. 8 talks about work in progress were given, framed by an introduction to Fractal and the Fractal project, and a final discussion about the evolution of the Fractal project. The technical talks described the recent developments concerning the Fractal ADL tool chain, the Fractal RMI ORB, the AOKell and ProActive implementations, reliability of Fractal components thanks to contracts (ConFract), behaviour protocols and model checking, with an original talk from the Nokia research center about dynamic and automatic configuration of components.

3.2 Call for proposals

The call for proposals, that was publicized on several mailing-lists (ObjectWeb, DBWorld, seworld, ACM SIGOPS France...), contained:

- a description and rationale for component-based architecture and its interest for the ECOOP conference;
- the expected audience: the Fractal community inside the ObjectWeb community hopefully enlarged thanks to ECOOP;
- the definition of scope of expected proposals: implementation and conformance test suites, model extensions, languages and tools, practical usage and feedback;
- and finally a description of the submission and selection process.

The submission and selection processes were rather light. Submissions were asked to contain 2 to 4 pages describing the work to be presented during the workshop. No full-length articles were asked for submission[10].

3.3 Selection and call for participation

More than 20 propositions were received, evaluated and discussed by the workshop organisers. Among them, 11 were selected for regular talks during the workshop. The selection was based on several individual criteria (technical maturity, originality, novelty) and also globally so as to cover a wide spectrum of activities around the Fractal component model and to make an interesting program with potential vivid discussions among participants. Most other proposals were very relevant but unfortunately could not fit in a one-day workshop, and were proposed to give place to poster presentations during breaks and lunch.

The final call for participation repeated the general items of the call for proposals and gave the detailed program with the list of regular talks and posters.

4 Tenue of the Workshop

The workshop took place the 3rd of July 2006. It was organized around 11 talks (typically 20 mn talk + 10 mn discussion) grouped in 5 sessions: Implementation and Basic Tools, Higher Languages and Tools, UML and MDA Design, Verification and Predictable Assembly, and Applications. 3 poster sessions also took place during coffee breaks and lunch. A final free discussion involving the around 30 participants closed the workshop.

4.1 Presentations and Discussions

The first morning session was devoted to implementations and basic tools for Fractal component programming.

L. Seinturier presented a joint work between INRIA Jacquard (L. Seinturier, N. Pessemier) and IMAG LSR laboratory (D. Donsez, C. Escoffier) towards a reference model for implementing the Fractal specifications in Java and the .Net platform. This preparatory work, fuelled by the development of the AOKell Fractal implementation and its port on the .Net platform, and a comparative analysis of the Julia implementation, advocates for a greater interoperability between Fractal implementations. The purpose of a Fractal implementation is to support the Fractal APIs and to offer mechanisms to compose control aspects inside membranes. Of course, all Fractal implementations support the Fractal API (with possible different conformance levels however) but offer generally different and incompatible mechanisms for building membranes. The aim of this line of work is to define some "Service Provider Interfaces" (SPI) that would

[10] A post-workshop editing and publishing activity to produce post-workshops proceedings was planned however.

embody programming conventions; implementations should follow these conventions so as to build assembly of, for instance, Julia and OAKell components and hopefully mix controllers/interceptors from different implementations. This line of work was acknowledged by the audience as very useful and important, and probably strongly connected to necessary efforts towards the definition of compliance test suites and benchmarks for Fractal implementations.

E. Özcan presented a status of the work in progress around THINK by STMicrolectronics (E. Özcan, M. Leclerc), France Telecom (J. Polakovic) and INRIA Sardes (J.-B. Stefani). The talk focused on recent developments of the ADL toolchain for THINK (Fractal ADL Factory) so as to make it more modular (finer-grained), extensible and retargetable, i.e. able to consider different back-ends corresponding to different hardware platforms. The talk concluded by listing other recent R&D activities and additions to the Kortex component library such as support for multi-processor platforms and support for customizable multimedia applications. The following discussion was not so much technical but concerned the collaborative management of the THINK code base. The THINK code base was historically managed by a few individuals from France Telecom and INRIA, with a quite clear direction and minimal collaborative decision making. Now, the growing implication of STMicrolectronics and others raises the question of how to choose between alternative propositions, e.g. concerning the design of the ADL tool chain for THINK, who is authorized to commit in the code base, who is authorized to create branches, etc.

The second session was devoted to higher languages and tools.

P.-C. David presented the work he did on FScript with T. Ledoux at Ecole des Mines de Nantes and France Telecom. FScript is a scripting language that allows for expressing reconfigurations of Fractal systems much more concisely, thanks to a higher level of abstraction than the bare Fractal APIs. FScript also includes FPath, a sublanguage/subsystem for navigation/query in Fractal architectures. It only comes with a Java backend for the time being but works are ongoing, e.g. at France Telecom, to use FScript to express reconfigurations in the THINK platform. One focus of the talk was the ACID-like transactional properties of FScript that would allow for *safe* reconfigurations. The vivid discussion following the talk revealed that this important but complex matter would/should require more developments.

R. Rouvoy presented the work on attribute-oriented programming around Fraclet with N. Pessemier, R. Pawlack and P. Merle at INRIA Jacquard. Fraclet is an annotation framework for Fractal components in Java. The motivation for this work is that component programming can be considered as verbose - and hence time consuming - by developers because the components code has to respect some conventions and provide meta-information as required by the Fractal model. Fraclet is composed of a library of annotations and plugins to generate[11] automatically various artifacts required by the Fractal component model (a.k.a. callbacks). Annotations provide a way to describe the component meta-

[11] Fraclet and attribute-oriented programming in general takes its roots in generative programming and aspect-oriented programming.

information directly in the source code of the content Java class. Fraclet plugins generate either Fractal component glue (use of Fractal APIs) or FractalADL definitions. Two implementations of the Fraclet annotation framework exist: Fraclet XDoc and Fraclet Annotation. Fraclet XDoc uses the XDoclet generation engine to produce the various artifacts required by the Fractal component model. Fraclet Annotation uses the Spoon transformation tool to enhance the handwritten program code with the non-functional properties of the component model. The talk emphasised two benefits of the approach. First, a reduction in development time and in the size of the components code produced "by hand". Second, a better support for software deployment and evolution: the presence in components code of architecture/deployment concerns facilitates the co-evolution of business and architecture/deployment code. This second benefit appeared as arguable from an industrial point of view: mixing, within the same file, business and deployment concerns might not appear as such a pleasant idea for software administrators. Also, a massive use of annotations is quite questionable with respect to code analysis and dependability in general. Most participants to the workshops were rather programmers than industrials and appeared quite enthusiastic about annotations and Fraclet anyway!

The third session was devoted to component modelling and more specifically to UML and MDA design.

V. Mencl presented a study with M. Polak at Charles University, Prague. They used their comparative analysis of UML 2.0 components and Fractal components to discuss possible mappings of Fractal concepts in UML. They actually proposed one specific mapping and instrumented it as a plug-in for the Enterprise Architect platform which is able to generate the Fractal ADL component descriptions, Java interfaces and a *skeleton* of the actual Java code of components. In the after-talk discussion, some possible future extensions were mentioned such as to *reverse engineer* UML models from Fractal ADL descriptions or runtime capture and representation in UML of a running Fractal system.

F. Loiret presented a study with D. Servat at CEA/LIST and L. Seinturier at INRIA Jacquard about modelling real-time Fractal components. This early work includes the definition of a EMF (Eclipse Modelling Framework) meta-model of Fractal IDL and ADL description, as well as the development of an Eclipse plug-in for actual generation of Fractal components targetting the THINK platform. The perspectives that were discussed include an extension of the meta-model to describe components behaviour and a reverse engineering tool chain to extract behaviour from the code of components.

The general discussion at the end of this modelling session acknowledged that there is probably not a unique direct mapping between UML and Fractal, especially because of specificities of Fractal such as component sharing and reflection (components controllers and membranes). However, thanks to UML/MDA (meta)modelling capabilities, different UML (meta)models could be defined to tackle Fractal specificities. People/teams interested by this line of work inside the Fractal community were encouraged to discuss further and hopefully to con-

verge towards a common meta model (or at least to assess if one such a common model would make sense).

In the afternoon, the fourth session was devoted to verification tools and predictable assembly.

J. Kofron presented the work on behaviour protocols by J. Adamek, T. Bures, P. Jesek, V. Mencl, P. Parizek and F. Plasil at Charles University, Prague. Behaviour protocols are basically a formalism that allows for the specification of the expected behaviour of components in terms of legal sequences of operation invocations on components interfaces. A static behaviour protocol checker has been developed in the context of the SOFA component models for several years by Charles University. Recently, behaviour protocols have been ported on the Fractal platform through a partnership between Charles University and France Telecom. The result is a static checker and a dynamic checker that include Java code analysis of primitive components with the JavaPathFinder (JPF) model checker.

E. Madelaine presented a case-study of verification of distributed components behaviour with L. Henrio and A. Cansado at INRIA Oasis/I3S/U. Nice. The case study application itself has been defined in a partnership between Charles University and France Telecom to experiment behaviour protocols (cf. previous paragraph). E. Madelaine and al. used this application to experiment with their own verification formalism, *parameterized networks* with their supporting verification platform Vercors. This formal approach allows for model-checking of components behaviour (typically deadlock and reachability checking). The work also mentioned the proposition of a new Fractal ADL module (defined in collaboration with Charles University) for attaching behaviour specification and associated verification tools in architecture descriptions.

D. Deveaux presented a work with P. Collet, respectively at Valoria/U. Bretagne Sud and I3S/U. Nice, on contract-based built-in testing. The approach leverages previous works on built-in testing of Java classes by Valoria and ConFract, a contracting system for Fractal by I3S and France Telecom. It proposes to instrument each component under test (CUT) with, for instance, ConFract contracts which embody the particular testing information of this component and a test controller that would generate a test bed component encapsulating (containing) each CUT. A prototype is currently under development. Some questions arose from the audience concerning the adherence to ConFract and if the approach was only applicable during the design phase or whether it would be used in a deployed system. D. deveaux explained that the system would exhibit low dependancy to the contracting system (alternative contract systems may be used instead of ConFract) and would not be limited to unit testing, but could also handle admission, integration and regression test thanks to the dynamic configuration management capabilities in Fractal.

The fifth and last session was devoted to applications in real life of the Fractal technology.

N. Rivierre presented the work around JMXPrism with T. Coupaye at France Telecom. JMXPrism is a mediation layer that stands between the systems to be

managed through JMX and management consoles or applications. JMXPrism provides a unique access point (embedding a JMX server) for managers that allows for the definition and management of *logical views* on the managed systems. JMXPrism prevents managers to access directly the managed systems and allows for filtering, renaming, etc. JMXPrims is implemented in Fractal which makes it very dynamic, allowing views and other components of a JMXPrims server to be changed very easily. JMXPrism embeds Fractal JMX, which was released some time ago as open source in the Fractal code base, and which allows for a declarative and non-intrusive exposition of Fractal components in JMX. JMXPrism has been used inside France Telecom to build a toy autonomic prototype controlling the creation of threads correlated to memory consumption. It has also been used more operationally in a grid information system project in partnership with Fujitsu to provide an homogenous view of resources in cluster on which resource sharing control was exercised to arbitrate two concurrently running applications: a visio-conference application exhibiting real-time QoS constraints and a batch-oriented scientific computing application.

G. Huang presented the last work with L. Lan, J. Yang and H. Mei at Peking University, Beijing, China on next generation J2EE servers. The work advocates for a combined use of reflective (applicative) components as embodied in Fractal or the ABC tool chain from Peking University and reflective middleware (especially EJB container) in future J2EE servers. Experiments are been conducted in PKUAS, an J2EE-compliant J2EE application server developed at Peking University. The talk raises up the engaged collaboration between ObjectWeb and OrientWare[12], a Chinese open source middleware consortium, as a suitable context for this line of work.

4.2 Final Discussion

The open discussion session was launched by a short talk by D. Caromel from INRIA Oasis, who reported on the Grid Component Model (GCM). GCM is an component model dedicated to grid systems that is being defined by the IST CoreGrid[13] network of excellence (NoE) along with the IST STREP project GridCOMP which is in charge of implementing, tooling and experimenting GCM. Fractal is considered as the basis for GCM and also as the main candidate to emerge as the standard component model for grid computing, at least in Europe. The talk recalled for some changes in the Fractal APIs that would be suitable for grid environments and that were discussed in previous Fractal workshop (e.g. multicast interfaces) but, more importantly, advocates for a close synergy between ObjectWeb/Fractal and CoreGrid/GridCOMP, i.e. a support of Fractal in CoreGrid and symmetrically a commitment from the Fractal community. This point was largely acknowledged as an important matter for the visibility and future of Fractal.

[12] http://www.orientware.org
[13] http://www.coregrid.net/

The discussion on the expected synergy between ObjectWeb/Fractal and CoreGrid/GridCOMP raised up a more general discussion about the evolution of the Fractal project. Some time ago was announced an evolution towards "Fractal v3". Some points were discussed during the previous Fractal workshop (November 2005), namely: i) evolution of the Fractal model specification (e.g. removal of some semantic ambiguities, changes and additions required for grid computing), including evolution in the organisation of the work on the specification with editors, editing committee and contributors, ii) evolution in the management of the Fractal code bases (e.g. cartography/matrix of (in)compatibilities between implementations and tools, conformance test suites) and iii) evolution of the Fractal web site (e.g. bibliography, success stories) and more generally of the management of the Fractal community (e.g. more structured workshops with CFP, program committee, proceedings; working groups inside the Fractal project). Since then, some elements contributed to "Fractal v3" in a quite informal way e.g. reflections on interoperability between Fractal implementations (cf. work by Seinturier and al. in the previous section), organisation of the two last workshops as satellite events of Middleware and ECOOP conferences (including CFP, PC and hopefully post-proceeding for Fractal Workshop at ECOOP), additions to the Fractal code base(s) (e.g. AOKell, FScript) and web site. The discussion at ECOOP, as well as previous informal discussions in particular on the Fractal mailing-list, revealed that some people were perhaps expecting a quicker evolution. Again, after a lively discussion, the workshop organisers pointed out that there might have been a misunderstanding and that what was intended by "Fractal v3" does not boild down to just a evolution of the specification of the model itself but refers to a collective effort with implication of many individuals that are part of the Fractal community so as to tackle the different issues at stake (implementations engineering and interoperability, conformance test suites, tools, common code base for uses cases and demonstrators, management of commit in code bases, web site, etc.).

From the evolution of Fractal, the discussion then jumped to the standardisation of Fractal. Several participants advocated for a more volunteer approach of the Fractal community towards standardization organisms. A vivid discussion took place to assess which standard committees/organizations would be most appropriate (ISO, IUT, Sun JCP, OMG...). Some others pointed out that standardization in the middleware area is a huge effort and the return on this investment not always remunerating. Most participants agreed that the group or institution they represent would not have much resources for such activities anyway. The question of standardization activities around Fractal remains largely open.

Acknowledgments We would like to thank the ECOOP conference and workshop organizers for their assistance in the preparation of this workshop. We would like to thank the Fractal community for its vitality, for having proposed so many talks at this workshop even though this was the 3rd Fractal workshop in less than a year. Thanks to A. Lefebvre for his careful reading and comments.