Ann. Telecommun. (2009) 64:1–4
DOI 10.1007/s12243-009-0086-1

Component-based architecture: the Fractal initiative

Gordon Blair · Thierry Coupaye · Jean-Bernard Stefani

1 The Fractal initiative

Component-based software engineering (CBSE) [32] and *software architecture* [30] have become mature and very active fields of study.[1] Both can be traced back to an early vision of systematically produced software [24] and are primarily concerned with the construction of software systems by composition or assembly of software components, with well-defined interfaces and explicit dependencies. Nowadays, it is generally recognized that component-based software engineering and software architecture approaches are crucial to the development, deployment, management, and maintenance of large, dependable software systems [6].

Several component models and associated architecture description languages (i.e., notations for defining components and component assemblages) have been defined in the past 15 years.[2] In large part, this diversity stems from differences in application domains targeted by the different models, as well as differences in choices for component interconnection and composition capabilities (much as there exists today a wealth of models and languages for concurrency).

In the face of this diversity, the *Fractal component model* [10]—a programming language-independent component model for the construction of highly configurable software systems introduced by France Telecom and INRIA in 2001—was designed from the outset to allow arbitrary forms of component interconnection and component composition and to allow other component models to be derived as Fractal personalities. To achieve this, the Fractal model combines ideas from three main sources: (1) CBSE and software architecture, (2) reflective systems [18], and (3) configurable and adaptable distributed systems, (notably, the OpenORB [7] system and its associated OpenCOM component model [12]).

From CBSE and software architecture, Fractal inherits basic concepts for the modular construction of software systems: software components, interfaces, and explicit connections (called *bindings*) between them. Notably, these concepts allow for the (recursive) construction and management of hierarchical component architectures possibly with sharing (a subcomponent can be part of several parent components) this last feature being an originality of Fractal. From reflective systems, Fractal inherits the idea that components can exhibit meta-level behavior reifying, through appropriate interfaces (called *controller* interfaces), their internal structure, and behavior. From configurable

[1] As witnessed, e.g., by their significant presence at major software engineering conferences such as ICSE (Int. Conf. on Soft. Eng.), FSE (Int. Conf. on Foundations of Soft. Eng.), and the existence of several dedicated conference series, including CBSE (Int. Symp. on Component-Based Soft. Eng.), and GPCE (Int. Conf. Generative Programming and Component Engineering).

G. Blair
Computing Department, Lancaster University,
InfoLab21, South Drive,
Lancaster LA1 4WA, UK

T. Coupaye (✉)
Orange Labs,
28 chemin du Vieux Chêne,
38243 Meylan, France
e-mail: thierry.coupaye@orange-ftgroup.com

J.-B. Stefani
INRIA Rhône-Alpes,
655 avenue de l 'Europe, Montbonnot St Martin,
38334 St Ismier Cedex, France

[2] See, e.g., [19] for an analysis of recent component models, [27] for a recent comparison of software component models based on a common design problem, [9, 25] for a survey of architecture description languages, and [20, 23] for a survey of formal foundations of component-based software.

and adaptable distributed systems, Fractal inherits a general naming and binding framework for component interconnection across networks of heterogeneous machines and the idea that reflection can be used successfully to implement nonfunctional services.

Since its inception in 2001, the Fractal model has benefited from the development of several implementations in different programming languages and environments (notably, Java, C, C++, Smalltalk, .Net) and of several tools to assist programmers and systems architects in their construction of Fractal systems, including support for an extensible architecture description language, an architecture browser and management console, etc. Most of these implementations and tools are accessible as open source software on the Fractal Project web site [1], which is part of the OW2 (formerly ObjectWeb) open source middleware consortium [2]. Among others, four results are worth mentioning here: the combination of component-based software engineering and aspect-oriented programming with Fractal [29]; the development of FPath, a navigation and query language for Fractal architectures, and of FScript, a domain-specific language for reconfiguring Fractal systems, which are presented in this special issue; the development of grid component model (GCM), an important Fractal personality for grid computing, which is presented in this special issue; the development of an extensible toolchain for heterogeneous architecture descriptions based on the Fractal ADL [21].

Fractal has also been used successfully for the development of several configurable software systems, as well as for building automated distributed systems management capabilities. Example systems built with Fractal include: the Think framework for the construction of reconfigurable operating system kernels [15, 26]; the PEtALS enterprise service bus [3]; the Perseus framework for the construction of persistency services and Speedo, an implementation of the JDO (Java Data Object) standard, which embeds Perseus [5]; the GOTM framework for the construction of transaction management services [28]; the Dream framework for the construction of communication subsystems and message-oriented middleware [22]; the CLIF framework for distributed load testing presented in this special issue. Systems management capabilities developed with Fractal include in particular: deployment and configuration management capabilities [4, 11, 14, 16, 17], self-repair capabilities [31], and overload management capabilities [8]. Again, most of these systems and software frameworks are available as open source software hosted by the OW2 consortium [2].

2 This special issue

This special issue is an outgrowth of the fifth Fractal workshop, which was held under the auspices of the ECOOP 2006 conference [13] and the result of an open call for proposals on the topic of this issue. Each paper was peer-reviewed by an international panel of experts in CBSE, software architecture, formal methods, operating systems, and distributed systems. The papers presented in this issue constitute a good cross-section of the research carried out in and out of the Fractal community and are a testament to the vitality of the CBSE field and of Fractal-based CBSE in particular. The papers fall roughly into three categories: (1) refinements and extensions to the Fractal model, (2) tool support for Fractal-based CBSE, and (3) applications of the Fractal model.

Under the first category *refinements and extensions of the Fractal model*, one finds the paper by Françoise Baude, Denis Caromel, Cédric Dalmasso, Marco Danelutto, Vladimir Getov, Ludovic Henrio, and Christian Pérez on the GCM. This work is the result of collaborative work that took place within the EC-funded IST CoreGrid Network of Excellence and is currently under consideration by the European Telecommunications Standards Institute to establish an international standard for component-based programming of grid computing applications.

The paper shows how to extend the Fractal model in order to support deployment on a grid, to support classical many-to-many communications found typically required by grid computing applications, and to add support for autonomic computations.

Under the category *tool support for Fractal-based CBSE*, one finds the paper by Barros et al. on behavioral models for distributed Fractal components, the paper by David et al. on FPath and FScript, the paper by Rouvoy and Merle on Fraclet, and the paper by Salmi et al. on performance evaluation of Fractal components.

The paper by Tomás Barros, Rabéa Ameur-Boulifa, Antonio Cansado, Ludovic Henrio and Eric Madelaine presents a new formal behavioral model, called parameterized networks of synchronized automata or pNets, which is expressive and compact enough to be used as an internal format for software tools intended for the verification of Fractal and GCM-based systems. The paper shows how to model active objects (used in the GCM/Proactive implementation), Fractal and GCM components, and shows in particular how to deal with meta-level behavior associated with Fractal components, a crucial requirement when considering control and management issues in reconfigurable systems.

The paper by Pierre-Charles David, Thomas Ledoux, Marc Léger, and Thierry Coupaye presents the FPath and FScript languages, together with transactional support required by reconfiguration operations described using the FScript language. This development is an important milestone for providing safe reconfiguration in distributed Fractal architectures. FScript (which embeds FPath) offers programmers and architects the benefits of a domain

specific language for concisely and safely expressing complex reconfiguration operations. FPath can be seen as a "pointcut language" for Fractal architectures, which can be helpful, more generally, in the description of complex meta-level operations on Fractal architectures.

The paper by Romain Rouvoy and Philippe Merle introduces Fraclet, an annotation framework which provides a simple way to support component-based programming in Java. In contrast to existing annotation frameworks for component models such as Enterprise Java Beans or the SCA components, Fraclet can be adapted to different models and their supporting environment using a generative approach. This is demonstrated by its application to Fractal and to the OpenCOM component models (also offering an interesting overview and comparison of these two advanced component models).

The paper by Nabila Salmi, Patrice Moreaux and Malika Ioualalen targets a missing element in the existing set of tools developed around the Fractal model, namely performance analysis for Fractal (and more generally component-based) systems. The paper shows how to model Fractal components using the classical model of Stochastic Well-formed Nets, a special class of high-level Petri nets useful for performance analysis, and proposes a structured analysis method to compute performance indices for a Fractal system, such as response times.

Under the category *applications of the Fractal model*, one finds the paper by Dillenseger on CLIF, and the paper by Lacoste et al. on kernel-level access control.

The paper by Bruno Dillenseger on CLIF describes a Fractal-based framework for distributed load testing. The CLIF framework addresses a recurrent performance evaluation concern from managers of large distributed systems. Testing, e.g., the behavior of large distributed Web application servers under various access patterns and load conditions requires comprehensive and distributed load testing facilities, which constitute themselves a nontrivial distributed system to set up and configure. The CLIF framework nicely demonstrates how to leverage the Fractal model in the construction of such configurable load testing facilities.

The paper by Marc Lacoste, Tahar Jarboui, and Ruan He presents an access control architecture for component-based operating system, called CRACKER, which has been implemented with Think. The CRACKER architecture enforces access control to operating system resources by protecting them with reference monitors (that perform security checks) implemented as component controller behavior, thus exploiting Fractal components as sandboxes. The architecture is remarkable for its flexibility and its clean separation of security check mechanisms from policy aspects, encapsulated in Fractal components. The paper also shows how flexible OS kernel authorization management can be implemented with a very acceptable system performance overhead.

Acknowledgments We would like to thank all the authors for submitting papers for consideration to this special issue. We would also like to thank the anonymous reviewers for their conscientious reviewing work and support that greatly help in the selection of the final set of seven papers. Each paper has been evaluated by at least three reviewers both inside and outside the Fractal community including reviewers from the Annals of Telecommunications scientific committee. Finally, we hope you, the reader, will enjoy reading this selection of papers as much as we did.

References

1. Fractal Project. URL: http://fractal.objectweb.org
2. OW2 Open Source Middleware Consortium. URL: http://www.ow2.org
3. PEtALS, Open source ESB. URL: http://petals.ow2.org
4. Abdellatif T, Kornas J, Stefani JB (2005) J2EE Packaging, deployment and reconfiguration using a general component model. In Component Deployment, 3rd International Working Conference, CD 2005, volume 3798 of Lecture Notes in Computer Science. Springer
5. Alia M, Chassande-Barrioz S, Dechamboux P, Hamon C, Lefebvre A (2004) A middleware framework for the persistence and querying of java objects. In 18th European Conference Object-Oriented Programming (ECOOP 2004), volume 3086 of Lecture Notes in Computer Science. Springer
6. Bass L, Clements P, Kazman R (2003) Software architecture in practice. SEI Series in Software Engineering, 2nd edn. Addison-Wesley, Toronto
7. Blair G, Coulson G, Andersen A, Blair L, Clarke M, Costa F, Duran-Limon H, Fitzpatrick T, Johnston L, Moreira R, Parlavantzas N, Saikoski K. The design and implementation of open ORB v2. IEEE Distributed Systems Online Journal, 2(6), November 2001
8. Bouchenak S, De Palma N, Hagimont D, Taton C (2006) Autonomic management of clustered applications. In IEEE International Conference on Cluster Computing (CLUSTER). IEEE
9. Bradbury J, Cordy J, Dingel J, Wermelinger M (2004) A survey of self-management in dynamic software architecture specifications. In Proceedings of the 1st ACM SIGSOFT Workshop on Self-Managed Systems, WOSS 2004. ACM
10. Bruneton E, Coupaye T, Leclercq M, Quema V, Stefani JB (2006) The fractal component model and its support in java. Softw Pract Exp 36:11–12
11. Caromel D, di Costanzo A, Delbé C (2007) Peer-to-Peer and fault-tolerance: towards deployment-based technical services. Future Gener Comput Syst 23:7
12. Coulson G, Blair GS, Grace P, Taïani F, Joolia A, Lee K, Ueyama J, Sivaharan T (2008) A generic component model for building systems software. ACM Trans Comput Syst 26:1
13. Coupaye T, Stefani JB (2007) Fractal component-based software engineering—report of Fractal CBSE Workshop at ECOOP'06. In ECOOP 2006 Workshop Reader, volume 4379 of Lecture Notes in Computer Science. Springer
14. David PC, Léger M, Grall H, Ledoux T, Coupaye T (2008) A multi-stage approach for reliable dynamic reconfigurations of component-based systems. In Distributed Applications and Interoperable Systems, 8th IFIP WG 6.1 International Conference, DAIS 2008, volume 5053 of Lecture Notes in Computer Science. Springer
15. Fassino J-P, Stefani J-B, Lawall J, Muller G (2002) THINK: a software framework for component-based operating system kernels. In USENIX Annual Technical Conference

16. Flissi A, Dubus J, Dolet N, Merle P (2008) Deploying on the Grid with DeployWare. In 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008). IEEE Computer Society
17. Flissi A, Merle P (2006) A generic deployment framework for grid computing and distributed applications. In OTM Confederated International Conferences, volume 4276 of Lecture Notes in Computer Science. Springer
18. Kiczales G, des Rivières J, Bobrow D (1991) The art of the metaobject protocol. MIT, Cambridge
19. Lau KK, Wang Z (2007) Software component models. IEEE Trans Softw Eng 33:10
20. Leavens G, Sitaraman M (2000) Foundations of component-based systems. Cambridge University Press, Cambridge
21. Leclercq M, Özcan AE, Quéma V, Stefani JB (2007) Supporting heterogeneous architecture descriptions in an extensible toolset. In 29th International Conference on Software Engineering (ICSE). IEEE Computer Society
22. Leclercq M, Quéma V, Stefani JB (2005) DREAM: a component framework for the construction of resource-aware, configurable MOMs. IEEE Distributed Syst Online 6:9
23. Liu Z, He J (2006) Mathematical Frameworks for Component Software—Models for Analysis and Synthesis. World Scientific, Mountain View
24. McIlroy MD (1968) Mass produced software components. In Naur P, Randell B (eds) Software engineering, report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968. Full report available at: http://homepages.cs.ncl.ac.uk/brian.randell/NATO
25. Medvidovic N, Taylor RN (2000) A classification and comparison framework for software architecture description languages. IEEE Trans Softw Eng 26:1
26. Polakovic J, Stefani JB (2008) Architecting reconfigurable component-based operating systems. J Systems Archit 54:6
27. Rausch A, Reussner R, Mirandola R, Plasil F (eds) (2008) The common component modeling example—comparing software component models, volume 5153 of Lecture Notes in Computer Science. Springer
28. Rouvoy R, Merle P (2007) Using microcomponents and design patterns to build evolutionary transaction services. Electr Notes Theor Comput Sci, 166
29. Seinturier L, Pessemier N, Duchien L, Coupaye T (2006) A component model engineering with components and aspects. In 9th Inter. Symp. on Component-Based Software Engineering (CBSE), number 4063 in Lecture Notes in Computer Science. Springer
30. Shaw M, Garlan D (1996) Software architecture: perspectives on an emerging discipline. Prentice Hall, Upper Saddle River
31. Sicard S, Boyer F, De Palma N (2008) Using components for architecturebased management: the self-repair case. In 30th International Conference on Software Engineering (ICSE 2008). ACM
32. Szyperski C (2002) Component software, 2nd edn. Addison-Wesley, Toronto