Note that our point in this paper is not to discuss the characteristics of synchronization schemes with growing complexity but in showing how our framework helps at reusing and customizing various levels of synchronization schemes, in order to produce (and implement) such refinements and variants. (You may again look at Fig. 2 which summarizes the hierarchy of synchronization schemes/classes that have been partially developed in this section.)

6 Evaluation, Related and Future Work

6.1 Evaluation of Current Framework

As we shown, implementations of various languages models, constructs and synchronization schemes have been rather easily integrated within a single framework. The benefits are in the possibilities to reuse and refine such models, constructs and schemes and to actually compare, apply and test them on real programs developed from the standard Smalltalk-80 environment. It is also possible to select and apply various synchronization schemes (and generally speaking various OOCP models) *locally* to various *parts* and *computational contexts* of a whole program/system.

Meanwhile, this experience in developing a hierarchy raises the general methodological issue of how to best extend such a taxonomy of classes while maintaining it highly modular.[13] Currently this is the sole responsibility of the designer and implementor. As we have now some significant number of classes within the three main components taxonomies, it will be interesting to see how semi-automatic hierarchy management/reclassification mechanisms (such as [CAS 92]) may help us to manage the evolution and refinement of the taxonomies.

6.2 Evaluation of Evolution

Note that the initial version of Actalk in 1988 was much smaller than current framework. The 88 kernel included only two component classes (activity and behavior were not separated yet) and much less methods. In 1994 we developed a large number of class extensions of the kernel to implement numerous OOCP computation models, languages and synchronization schemes. Current Actalk framework emerged through the identification of the stable parts (kernel components and their basic methods) and their parametrization (parameter methods). This complexified the architecture albeit increased the possibilities. By comparing initial platform and current framework, we note that the initial platform got the larger use by numerous teams who used the kernel as a basis for implementing systems based on active objects. Thus, by using terms defined by Christopher Alexander [ALE 75] and highlighted by Richard Gabriel

[13] We therefore use hierarchic parameterization of the architecture, i.e. decomposing further parameter methods as needed within the hierarchy, as discussed in [BRI 94].

[GAB 96], we could say that although the 94 framework provided more extensive and systematic modeling abilities, *habitability* and *piecemeal growth* was better achieved for the first - lighter - version of Actalk.

6.3 Related Work

The pros of building object-oriented (and more specifically Smalltalk-based) generic platforms for classifying various programming constructs has also been demonstrated by other platforms, such as Simtalk (for modeling various simulation schemes [BÉZ 87]) as well as others (Classtalk, Prototalk...). A generic scheduler has also been developed, as part of the Actalk project, by Loïc Lescaudron to classify and parameterize various scheduling policies [LBB 91, BRI 93].

Alternatives to represent various OOCP designs are some more formal approaches, as the object calculus proposed by Oscar Nierstrasz [NIE 93]. Note that our pragmatic approach allows experiments with actual programs within a sophisticated programming environment (Smalltalk-80 based) to help at developing and monitoring them [LBB 91].

The component-based architecture of an Actalk active object is close to the component-based meta-architecture of CodA, designed by Jeff McAffer [MCA 95]. CodA provides finer grain components and more refined interface between them, but at the cost of more complexity.

There are also other alternatives to components and methods decomposition, as for instance with parameterizing the invocation of some methods via arguments, as advocated by the Hermes/ST architecture [FHR 93]. This is also related to the issue of having a minimal kernel and extending it with more complex protocols and mechanisms versus having a bigger kernel offering more protocols not necessarily all used at the kernel level (see also previous discussion in Sect. 6.2). In the case of Actalk, the kernel has been designed a while ago to be simple and efficient, as its initial motivation was mainly didactic [BRI 89]. The scope of Actalk has also been restricted as it does not address distribution aspects (as for CodA) or fault-tolerance aspects (as for Hermes/ST) but concentrates on OOCP language design.

6.4 Future Work

Besides the areas and directions for future work already mentioned, an important and general issue is the combination of various aspects/descriptions of a computational behavior. The decomposition of the Actalk architecture in orthogonal components, and their further decomposition in parameter methods, help at such combination. Meanwhile, it reaches some limitations when combining differents versions of a same component (in the context of this paper, the activity/synchronization component). The programmer may have then to rely on some amount of explicit combination. A finer grain decomposition of components, as proposed by CodA [MCA 95], brings more independence and modularity, but it also still relies on a single component to cover activity and

synchronization concerns. Our belief is that it is difficult anyway to further decompose a complex aspect, such as activity/synchronization, in fully orthogonal pieces. (In other words, we ultimately reach some atoms or even quarks.) Therefore, we believe that we cannot avoid the general problem of composing non fully orthogonal components, and that we should develop some rationale and methodology for doing so. Some starting propositions may for instance be found in [McA 95].

7 Conclusion

In this paper we have described the design of a framework for object-oriented concurrent programming, its architecture, and examples of use. We implemented successive refinements and combinations of synchronization schemes in order to show the expressive power of our platform. We finally evaluated our experience, related it to other works, and pointed future areas of investigation.

Latest version of Actalk (including some documentation) is available through anonymous ftp/http:
"`ftp://ftp.lip6.fr/lip6/softs/actalk/actalk.html`"
or: "`ftp://ftp.yl.is.s.u-tokyo.ac.jp/pub/actalk/actalk.html`".

References

[Agh 86] G. Agha, *Actors: a Model of Concurrent Computation in Distributed Systems, Series in Artificial Intelligence*, MIT Press, 1986.

[AWY 93] G. Agha, P. Wegner, and A. Yonezawa (editors), *Research Directions in Concurrent Object-Oriented Programming*, Mit Press, 1993.

[Ale 75] C. Alexander, "The Oregon Experiment," Oxford University Press, 1975.

[AS 83] G.R. Andrews and F.B. Schneider, "Concepts and Notations for Concurrent Programming," *ACM Computing Surveys*, Vol. 15, n° 1, March 1983, pages 3–43.

[Béz 87] J. Bézivin, "Some Experiments in Object-Oriented Simulation," *Proc. of OOPSLA '87*, Special Issue of ACM SIGPLAN Notices, Vol. 22, n° 12, December 1987, pages 394–405.

[BH 94] S. Billet-Coat and D. Hérin-Aimé, "A Multi-Agent Architecture for an Evolving Expert System Module," in *5th International Conference on Database and Expert Systems Applications (DEXA '94)*, edited by D. Karagiannis, LNCS, No 856, Springer-Verlag, September 1994, pages 581–590,

[BFS 91] T. Bouron, J. Ferber, and F. Samuel, "MAGES: a Multi-Agent Testbed for Heterogeneous Agents," *Decentralized Artificial Intelligence*, Vol. II, edited by Y. Demazeau and J.-P. Muller, North Holland, 1991.

[Bri 89] J.-P. Briot, "Actalk: a Testbed for Classifying and Designing Actor Languages in the Smalltalk-80 Environment," *Proc. of ECOOP '89*, edited by S. Cook, Cambridge University Press, July 1989, pages 109–129.

[Bri 93] J.-P. Briot, "Object-oriented design of a generic scheduler," *Object-Oriented Computing II* (Proc. of JSSST WOOC'93), edited by A. Yonezawa, S. Matsuoka, and W. Kato, Lecture Notes/Software Science, n° 6, Kindai-Kagaku-Sha, Japan, 1994, pages 73–81.

[Bri 94] J.-P. Briot, "Modélisation et Classification de Langages de Programmation Concurrente à Objets : l'Expérience Actalk," *LITP Research Report*, n° 94/59, Institut Blaise Pascal, France, October 1994. (Also in *Proc. of Conference "Langages et Modèles à Objets (LMO '94)"*, INRIA/IMAG/PRC-IA, Grenoble, France, October 1994, pages 153–165.)

[Bri 96] J.-P. Briot, "An Experiment in Classification and Specialization of Synchronization Schemes," *Object Technologies for Advanced Software* (Proc. of ISOTAS'96), edited by K. Futatsugi and S. Matsuoka, LNCS, n° 1049, Springer-Verlag, March 1996, pages 227–249.

[BG 96] J.-P. Briot and R. Guerraoui, "On the Use of Smalltalk for Concurrent and Distributed Programming," *Informatik/Informatique*, Swiss Professional Informatics Societies, Switzerland, Special Issue on "Smalltalk (Languages, Tools, Environments)," edited by R. Guerraoui, February 1996.

[BGL 98] J.-P. Briot, R. Guerraoui, and K.-P. Löhr, "Concurrency and Distribution in Object-Oriented Programming," *ACM Computing Surveys*, Vol. 30, n° 3, September 1998, pages 291–329.

[BGY 96] J.-P. Briot, J.-M. Geib, and A. Yonezawa (editors), *Object-Based Parallel and Distributed Computation*, LNCS, Springer-Verlag, 1996.

[Cas 92] E. Casais, "An Incremental Class Reorganization Approach," *Proc. of ECOOP'92*, edited by O. Lehrmann Madsen, LNCS, n° 615, Springer-Verlag, June 1992, pages 133–152.

[FHR 93] M. Fazzolare, B.G. Humm, and R.D. Ranson, "Object-Oriented Extendibility in Hermes/ST, a Transactional Distributed Programming Environment," *Proc. of the ECOOP'93 Workshop on Object-*

Based Distributed Programming, edited by R. Guerraoui, O. Nierstrasz, and M. Riveill, LNCS, n° 791, Springer-Verlag, 1994, pages 241–261.

[GAB 96] R.P. GABRIEL, "Patterns of Software - Tales from the Software Community," Oxford University Press, 1996.

[GHJ+95] E. GAMMA, R. HELM, R. JOHNSON, and J. VLISSIDES, *Design Patterns: Elements of Reusable Object-Oriented Design*, Addison-Wesley, 1995.

[GHÉ 94] K. GHÉDIRA, "A Distributed Approach to Partial Constraint Satisfaction Problems", *Proc. of the 6th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'94)*, LNCS, n° 1069, Springer-Verlag, 1996.

[GS 91] S. GIROUX and A. SENTENI, "ReActalk, a Reflective Version of Actalk," *OOPSLA'91 Workshop on Reflection and Metalevel Architectures*, Xerox PARC Technical Report, October 1991.

[GB 98] Z. GUESSOUM and J.-P. BRIOT, "Building Agents as an Extension of Active Objects - Application to the Simulation of Economic Models," *Proceedings of the Conférence Africaine sur la Recherche en Informatique (CARI'98)*, INRIA, October 1998.

[KG 94] M. KANG and D.D. GRANT, "A Kernel Mechanism for Simulating an Object-Oriented Process Sensitive Software Engineering Environment," *Proc. of TOOLS Pacific'94 (TOOLS 15)*, ISE - Prentice Hall, Fall 1994, pages 57–67.

[LBB 91] L. LESCAUDRON, J.-P. BRIOT, and M. BOUABSA, "Prototyping Programming Environments for Object-Oriented Concurrent Languages: a Smalltalk-Based Experience," *Proc. of TOOLS USA '91*, ISE - Prentice Hall, August 1991, page 449–462.

[LES 92] L. LESCAUDRON, "Prototypage d'Environnements de Programmation pour les Langages à Objets Concurrents : une Réalisation en Smalltalk-80 pour Actalk," *Thèse d'Université*, TH93.11, LITP, Université Paris VI - CNRS, France, May 1992 (in french).

[MY 93] S. MATSUOKA and A. YONEZAWA, "Analysis of Inheritance Anomaly in Object-Oriented Concurrent Programming Languages," *Research Directions in Concurrent Object-Oriented Programming*, edited by G. Agha, P. Wegner, and A. Yonezawa, Mit Press, 1993, pages 107–150.

[MCA 95] J. MCAFFER, "Meta-level Programming with CodA," *Proc. of ECOOP'95*, edited by W. Olthoff, LNCS, n° 952, Springer-Verlag, August 1995, pages 190–214.

[MCH 94] C. MCHALE, S. BAKER, B. WALSH, and A. DONNELLY, "Synchronization Variables," *Technical Report*, TCD-CS-94-01, Dept. of Computer Science, University of Dublin, January 1994.

[NH 96] P. NEUHAUS and U. HAHN, "Restricted Parallelism in Object-Oriented Parsing," *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, Vol. 1, Kyoto, Japan, August 1996, pages 379–385.

[NIE 93] O. NIERSTRASZ, "Composing Active Objects," in [AWY 93], pages 151–171.

[RV 77] P. ROBERT and J.-P. VERJUS, "Towards Autonomous Descriptions of Synchronization Modules," *Proc. of IFIP'77*, edited by B. Gilchrist, North-Holland, pages 981–986, August 1977.

[SG 96] M. SHAW and D. GARLAN, *Software Architectures: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[THO 94] L. THOMAS, "Inheritance Anomaly in True Concurrent Object Oriented Languages: A Proposal," in *Proc. of IEEE TENCON'94*, August 1994, pages 541–545.

[YT 87] A. YONEZAWA and M. TOKORO (editors), *Object-Oriented Concurrent Programming, Computer Systems Series*, MIT Press, 1987.