Channel-based Coordination Models and Languages for Component Composition: A Survey

Patricia Derler

***Abstract*—This paper surveys the field of channel-based coordination models for component composition. Composition of systems out of components can be done by interaction on a component level or by coordinating components from the outside. Coordination from the outside, also called exogenous coordination, imposes many advantages to software systems like easy reusability and maintainability or the ability to dynamically plug in new components into a system. Exogenous coordination can be done by using channels to connect components. Components passively exchange data items via channels, the communication protocol is given by the topology of the connections and the behavior of the channels. Examples for channel behaviors are synchronous or asynchronous, lossy or blocking channels. This paper discusses coordination models and languages dealing with channel-based component composition.**

***Index Terms*—channel-based, component composition, coordination model, coordination language**

I. Introduction

Building software systems out of independent components to encapsulate functionality is a best practice in software development. Important goals of component based design of software is maintainability and reusability. But it is not enough to build software systems out of independent components, the way those components interact also has a big impact on the quality of software, especially on the reusability of components in a different context. A component can interact with another component by initiating communication itself through method call semantics. Components have to know about other components interfaces in order to use them which result in tight coupling of components. In tightly coupled systems the extraction of a component from its context and the reuse in a different context is very difficult. A coordination of components from the outside can help in avoiding tight coupling of components. Control from the outside is also called exogenous coordination, the opposite of endogenous coordination which means that the components coordinate themselves. In exogenous coordination models, the control is never given to the components. Components are only passive entities in a system exchanging data. By controlling components from the outside, it is not necessary to know about components and their interfaces in order to use them. Exogenous coordination can be done by writing glue code that manages the interaction between components. Glue code is typically written in scripting languages. The amount of glue code is directly proportional to the size of the system which, in large systems, soon becomes hard to maintain and reuse. To avoid having monolithic glue code, the code for interaction can also be built compositionally out of components. Reusable glue code components can be constructed out of channels and connectors.

Paris-Lodron University of Salzburg, Austria
email: patricia.derler@cs.uni-salzburg.at

This paper is organized as follows: In Section II, the notion of coordination is explained and III introduces the concept of (mobile) channels for coordination. Section IV describes models for channel-based component composition and section V discusses languages for channel-based component composition. Section VI shows some examples where channel-based component composition was used. The paper concludes with a brief summary in section VII.

II. What is Meant by Coordination

Arbab states in [1] that coordination is the problem of ensuring proper communication among code pieces of active entities in an application. In his research, he focuses on the model of cooperation which has to be chosen to enable communication by means of choosing communication primitives that must be used. Coordination is the study of topologies of interactions and the realization of protocols to ensure the well-behavedness of a system. The goal of coordination models and languages is to find a solution to the problem of managing the interaction among concurrent programs. There are multiple models and languages for coordination which can be classified as data-oriented or control-oriented, exogenous or endogenous. Data-oriented applications are concerned with what happens to the data and usually model data in a substantial shared body. Examples for a data-oriented application are databases. Control-oriented applications describe a set of activities that produce, consume and transform data; important is the flow of control. Work-flow organizations are examples for control-oriented applications. Coordination must be considered in all parts of the software development process, i.e. in design, development, debugging, maintenance and reuse. This often leads to an early integration of the communication protocol and makes it hard to change the communication protocol at a later date. The coordination protocol should be made explicit to avoid errors, facilitate changes, enable reuse as well as the adoption of different coordination models. Surveys on coordination models and languages can be found at [14], [15], [16] or [3].

III. Coordination via Channels

An efficient way of describing and implementing interaction between components can be done via channels [27]. Channels allow anonymous point-to-point communication among components. Mobile channels allow the dynamic reconfiguration of channels in a system without affecting the components or having components noticing these changes.

Scholten [28] presents a coordination framework called MoCha for asynchronous mobile channels in distributed systems. [29] presents an exogenous coordination calculus for MoCha based on mobile channels which is built upon the π-calculus, a model for describing concurrent computation as systems of communication agents [30].

Andrade and Fiadeiro [32] present a mathematical pattern for component coordination based on the notion on channels. Channels are described as programs which do not perform any computation but enable the communication between two components by establishing a connection.

IV. Models for Channel-based Component Composition

The oldest channel-based models for component composition are dataflow models, Petri-nets [5] and Kahn networks [6]. Dataflow models are used largely to visualize the data flow in systems. The idea of 'coordination from the outside' is a basic principle of data flow models. The focus in data flow models is on the coordination of nodes by input and output operations. Broy and Stefanescu [24] research the algebraic structure of dataflow networks by basing the semantics of dataflow networks on stream processing functions. This research describes the algebraic calculus that combines laws of graph isomorphism and laws of semantic characteristics of dataflow nodes and studies deterministic and nondeterministic cases. A generalization of data-flow networks for describing dynamically reconfigurable or mobile networks is given in [17] and [18] using the model of stream functions.

Petri-nets [5] are graphs consisting of nodes and directed arcs connecting these nodes. *Place nodes* contain tokens and *transition nodes* process (also called fire) these tokens and put tokens into other place nodes. Petri nets are nondeterministic because in multiple enabled transitions there is no specification on which transition fires first. Petri nets are used in software design but also analysis and diagnostics and in workflow management. In [12], Guillen-Scholten et al. discuss systems communicating through mobile channels using Petri-nets as a modeling language.

Kahn [6] networks represent channel histories as streams and processes as continuous functions on streams. A network is a system of recursive stream equations and network behavior is the least fixed point of network equations. Processes communicate via unbounded FIFO channels by writing and reading tokens to and from channels. Kahn networks are deterministic, for the same input tokens, the system always produces the same set of output tokens. Kahn networks are used for modeling distributed systems and signal processing systems.

The Idealized Worker Idealized Manager (IWIM) Model [2] picks up the idea of Kahn networks and extends it. IWIM is a generic model of communication. Important concepts are compositionality (inherited from the data-flow model), anonymous communication and the separation of computation concerns from communication concerns. This separation describes exogenous coordination. The name IWIM comes from the concept of a weak dependence of workers on their environment. Each process is an individual worker. The weak dependence allows more sophisticated exogenous coordination of active entities in a system. IWIM describes processes, events, ports and channels. Processes are black boxes with well defined ports through which data can be exchanged. Ports are connected via channels and events are broadcasts of information to the system. IWIM introduces different kinds of channels which differs from Kahn networks where only one channel kind, namely FIFO channels, are used. Communication is supported by events sent via ports. IWIM is specialized on processes and also deals with process instance creation and installation. Manifold [7], a coordination language built upon the IWIM model, will be discussed in section V.

In [23], Arbab et al. describe a formal model for component-based systems. A formal, logic-based component interface description language is introduced that conveys the observable semantics of components. Components are black boxes that communicate via unbounded FIFO buffers which are called channels. Channels can be dynamically relocated. The component interface only shows observable behavior of a component, i.e. the channels it is connected to, blocking invariants to describe possible deadlock behavior, preconditions and postconditions which describe the contents of the buffers for the initial external channels and when the system terminates. This interface specification leads to the possibility of reasoning about the correctness of an entire system which extends the usual notion of partial correctness by excluding deadlocks.

Reo [9] builds upon the IWIM model of coordination and the coordination language Manifold. Reo comes from the Greek word $\rho\varepsilon\omega$ pronounced 'rhe-oh' and means flow as water in streams or channels. Reo is an exogenous coordination language based on a calculus of channel composition. This model introduces a variety of channels like synchronous channels, asynchronous channels, drains, spouts and lossy channels. In Reo, complex connectors are built out of simple channels or other connectors. Connectors are reusable and build networks consisting of channels and nodes. Communication in Reo is not deterministic but fairness can be guaranteed. Reo is not only a coordination model for process models but can be used for any kind of active entity inside a component. Examples are threads, agents or actors. Reo is more general than dataflow models, Kahn-networks and Petrinets which can be seen as specialized channel-based models. A formal description of Reo is given in [25] in a coalgebraic semantics. Reo connectors are modelled as relations or timed data streams consisting of twin pairs of separate data and time streams. A whole toolset is evolving around Reo for modeling, simulation, model checking, animation and code generation of Reo circuits.

V. Languages for Cannel-based Component Composition

Examples for coordination languages are Linda [4], a data-oriented language, which [13] claims to be one of the best known. Linda introduces the notion of a shared tuple space which is a centrally managed space containing all pieces of information that processes want to communicate.

Manifold ([7], [8]) is a control oriented coordination language and an incarnation of the IWIM model described in IV. Manifold supports dy-

namic reconfiguration of Kahn network topologies and explicitly supports connectors. All communication in Manifold is asynchronous. A compiler, a run-time system library, utility programs and libraries support in writing a Manifold application. A Manifold application consists of processes running on a network of heterogeneous hosts. Processes may be written in different programming languages and they don't need to know about Manifold. Visifold [26] is a visual programming environment for Manifold.

The $\sigma\pi$ coordination language [13] is inspired by Manifold. $\sigma\pi$ is a core language for specifying dynamic networks of components. A program is seen as a number of components each consisting of a collection of separable, reusable classes. Objects as instances of a class execute in parallel with other objects. $\sigma\pi$ enables anonymous communication via synchronous or asynchronous, mobile channels. Interaction is controlled by the objects executing on the system which means that $\sigma\pi$ does not impose exogenous coordination. The separation of coordination and computation is nevertheless easily reached by specifying each classes external channels in the component interface. Port-managers implement these interfaces and as a result only have access to the ports of a component and can not affect the computation. A coordinator describes the communication between port-managers by creating components and their associated port-managers, links components and transmits data items in channels between port-managers. The coordinator can be seen as the coordination protocol.

VI. Applications of Channel-based Component Composition

[13] lists a few areas, where coordination languages have been applied, including the parallelization of computation intensive sequential programs in the fields of simulation of fluid dynamics systems, matching of DNA strings, molecular synthesis, parallel and distributed simulation, monitoring of medical data, computer graphics, analysis of financial data integrated into decision support systems and game playing (chess). This section mentions a few concrete applications for channel-based component composition.

The following list contains applications of channel-based models to show the broad usage from modeling of software systems to modeling biological organisms. A big benefit of modeling interaction between components with channels is the intuitive graphical flow-diagram representation.

A. Enhancing Component Interfaces to Support Component Composition

In [10] Amaro, Pimentel and Roldan address the problem of interoperability of components at the protocol level by specifying the interaction behavior of software components. The approach enhances component interfaces by extending interface description languages with a description of an abstract component interaction protocol (i.e. the interactive behavior of a component). This protocol enables compatibility checks (e.g. when two components can interact without deadlocking) dependent on the connector considered for composition of components. Substitutability of components can be analyzed with respect to preserving a "safe" behavior of the system.

B. Quality of Service in Service-oriented Architectures

Meng describes in [22] an approach of how to assure quality of service requirements in Service-oriented Applications. QoS values are non-functional requirements of users to a component or to the behavior of a system consisting of interconnected components. QoS values can be specified in an algebraic model. Constraint automata can be used to compute QoS values for connectors. To meet the QoS requirements, connectors from different providers can be selected such that their composition satisfies the non-functional requirements.

C. Software Adaption

The discipline of software adaption [19] deals with topics related with managing the entities of a system to properly communicate with each other. Eterovic et al. [20] describe an approach for software adaption using coordination models and aspect-oriented techniques. An aspect-oriented architecture description language (Ao-

Rapide) is used to separate adaption aspects at the software architecture level. Separating the aspects is done using the coordination model Reo by wrapping functional components and linking them via Reo connectors.

D. Improving the Deployment Process with Respect to QoS

The deployment process of software components for large, distributed applications usually has many constraints and requirements. It is very difficult to do the deployment manually, automated tools and techniques are needed. If QoS parameters like performance or reliability have to be taken into consideration, a straight-forward deployment is not possible. [21] presents a graph-based approach for software deployment enabling the planning with respect to the communication resources. Those communication resources are channels required by components and communication resources available on the hosts in the target environment. Component-based applications and distributed environments are modeled as graphs and the deployment is defined as a mapping of the application graph to the target environment graph. The approach pays attention to behavior, cost, speed and security of the interconnections among components of the application which have significant effects on the applications QoS. The paper provides an example that models the composition of Web Services using Reo showing how application and target environment are modelled and how the mapping function is calculated.

E. Modelling Coordination in Biological Systems

Models of biological systems from cells to organisms are used for understanding effects and side-effects of influences on a system. An example where models for biological systems are used is drug research. Models capture the causality, dependence, conflicts and competition between entities. The main advantage of modeling biological systems is that models are predictable and behavior is modeled in terms of boundary conditions. Coordination in terms of biology means communication between cells which can be rather complex. Regulatory gene networks can be modelled using coordination languages. In [11], Clarke, Costa and Arbab show an application of Reo which provides a model for describing and reasoning about the behavour of biological systems. Benefits of using Reo for modeling biological systems are the formal techniques, the algebraic behavior and the abstraction from molecular actions provided by Reo.

F. Multi-Agent Systems

Dastani et al. present in [31] the application of a channel-based exogenous coordination language on multi-agent systems using Reo as a coordination model. The paper describes an approach for modeling and verifying the organizational structure of multi-agent systems consisting of individual agents or other multi-agent systems. Noteworthy is the possibility of dynamic reconfiguration of organizational structures of multi-agent systems.

VII. Conclusion

Component composition and coordination of components is an area with a lot of ongoing research. Especially for distributed systems which are prone to topological changes, exogenous coordination via channels turns out to be a promissing approach for managing the complexity of the specification and implementation of an independent coordination protocol.

This paper gives a brief overview of the work done in the field of channel-based coordination models and languages for component composition and mentions areas where this approach was successfully applied in order to improve quality, understandability and maintainability of heterogeneous systems.

References

[1] Arbab, F. (1998) What Do You Mean, Coordination?. *Bulletin of the Dutch Association for Theoretical Computer Science*, NVTI 11-22.

[2] Arbab, F. (1996) The IWIM model for coordination of concurrent activities. *Coordination Languages and Models* (April 1996), P. Ciancarini and C. Hankin, Eds., vol. 1061 of Lecture Notes in Computer Science, Springer- Verlag, pp. 34-56.

[3] Ciancarini, P. (1996) Coordination models and languages as software integrators. ACM Comput. Surv. 28, 2 (Jun. 1996), 300-302.

[4] Carriero, N., and Gelernter, D. (1989) LINDA in context. *Communications of the ACM* 32 (1989), 444-458.

[5] C.A. Petri (1996) Nets, Time and Space. Theoretical Computer Science, 153:348.

[6] Kahn, C. (1974) The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information Processing '74: Proceedings of the IFIP Congress*, 471-475. North-Holland, New York, NY, 1974.

[7] Bonsangue, M., Arbab, F., de Bakker, J., Rutien, J., Scutella, A., and Zavattaro, G. (2000) A transition system semantics for the control-driven coordination language manifold. *Theoretical Computer Science* 2004 3-47.

[8] Arbab, F. (1996) Manifold version 2: Language reference manual. Technical report, CWI, Amsterdam, The Netherlands, 1996. Available on-line at the URL: `http://www.cwi.nl/ftp/manifold/refman.ps.Z`.

[9] Arbab, F. (2004) Reo: a channel-based coordination model for component composition. *Mathematical. Structures in Comp. Science*, 14.3, 2004, 329-366, Cambridge University Press.

[10] Amaro, S., Pimentel, E., Roldan, A., Reo Based Interaction Model. *Electronic Notes in Theoretical Computer Science*, FACS 2005.

[11] Clarke, D., Costa, D., Arbab, F. (2004) Modelling Coordination in Biological Systems. ISoLA 2004: 9-25.

[12] Guiellen-Scholten, J., Arbab, F., de Boer, F., Bonsangue, M. (2005) Modeling the Exogenous Coordination of Mobile Channel-based Systems with Petri Nets. *Electronic Notes in Theoretical Computer Science*, Volume 154, Issue1, 2006, FOCLASA 2005.

[13] Arbab, F., de Boer, F. and Bonsangue, M. (2000a) A coordination language for mobile components. *Proc. ACMSAC'00.*

[14] Arbab, F. (1998) Coordination and its Relevance. In Proceedings of the 9th international Workshop on Database and Expert Systems Applications (August 26 - 28, 1998). DEXA. IEEE Computer Society, Washington, DC, 529.

[15] Gelernter, D. and Carriero, N., (1992) Coordination languages and their significance. Commun. ACM 35, 2 (Feb. 1992).

[16] Papadopoulos, G. A., Arbab, F. (1998) Coordination models and languages, Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-369X.

[17] Broy, M. (1995) Equations for describing dynamic nets of communicating systems. *5th COMPASSworkshop. Springer-Verlag Lecture Notes in Computer Science* 906 170-187.

[18] Grosu, R., Stoelen, K. (1996) A model for mobile point-to-point data-flow networks without channel sharing. *Springer-Verlag Lecture Notes in Computer Science* 1101 504-519.

[19] Yellin, D. M., Strom, R. E. (1997) Protocol Specications and Component Adaptors. ACM Transaction on Programming Languages and Systems, 19(2).

[20] Eterovic, Y., Murillo, J. M., Palma K. (2004) Managing components adaptation using aspect oriented techniques. In *First International Workshop on Coordination and Adaptation Techniques for Software Entities* (WCAT04, held in conjunction with ECOOP 2004), June 2004.

[21] Heydarnoori, A. (2006) Caspian: A QoS-Aware Deployment Approach for Channel-based Component-based Applications, Technical Report.

[22] Meng, S. (2007) QCCS: A Formal Model to Enforce QoS Requirements in Service Composition, tase, pp. 389-400, *First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering* (TASE '07).

[23] Arbab, F., de Boer, F. S., and Bonsangue, M. M. (2000) A Logical Interface Description Language for Components. In *Proceedings of the 4th international Conference on Coordination Languages and Models* (September 11 - 13, 2000). A. Porto and G. Roman, Eds. Lecture Notes In Computer Science, vol. 1906. Springer-Verlag, London, 249-266.

[24] Broy, M. and Stefanescu, G. (2001) The algebra of stream processing functions. Theor. Comput. Sci. 258, 1-2 (May. 2001), 99-129.

[25] Arbab, F., Rutten, J., (2002) A coinductive calculus of component connectors, Technical Report SEN-R0216, CWI, Amsterdam, September 2002.

[26] Bouvry, P. and Arbab, F. (1996) VISIFOLD: A Visual Environment for a Coordination Language. In *Proceedings of the First international Conference on Coordination Languages and Models* (April 15 - 17, 1996). P. Ciancarini and C. Hankin, Eds. Lecture Notes In Computer Science, vol. 1061. Springer-Verlag, London, 403-406.

[27] Guillen-Scholten, J., Arbab, F., de Boer, F., and Bonsangue, M. (2006) A Component Coordination Model Based on Mobile Channels. Fundam. Inf. 73, 4 (Dec. 2006), 561-582.

[28] Guillen-Scholten, J.G. (2001) MoCha: A model for distributed Mobile Channels. Master's thesis, Leiden University.

[29] Guillen-Scholten, J., Arbab, F., de Boer, F., and Bonsangue, M. (2005) MoCha-pi, an exogenous coordination calculus based on mobile channels. In *Proceedings of the 2005 ACM Symposium on Applied Computing* (Santa Fe, New Mexico, March 13 - 17, 2005). L. M. Liebrock, Ed. SAC '05. ACM Press, New York, NY, 436-442.

[30] Milner, R. (1999) Communication and Mobile Systems: The Pi Calculus. *Cambridge University Press.*

[31] Dastani, M., Arbab, F., and de Boer, F. (2005) Coordination and composition in multi-agent systems. In *Proceedings of the Fourth international Joint Conference on Autonomous Agents and Multiagent Systems* (The Netherlands, July 25 - 29, 2005). AAMAS '05. ACM Press, New York, NY, 439-446.

[32] Andrade, L., Fiadeiro, J. (2001) Coordination patterns for component-based systems, In *Proceedings of the V Brazilian Symposium on Programming Languages* SBLP'2001.