Jean-Marie Jacquet
Gian Pietro Picco (Eds.)

Coordination Models and Languages

LNCS 3454

7th International Conference, COORDINATION 2005
Namur, Belgium, April 2005
Proceedings

Springer

Jean-Marie Jacquet Gian Pietro Picco (Eds.)

Coordination Models and Languages

7th International Conference, COORDINATION 2005
Namur, Belgium, April 20-23, 2005
Proceedings

Springer

Volume Editors

Jean-Marie Jacquet
University of Namur
Institute of Informatics
Rue Grandgagnage 21, 5000 Namur, Belgium
E-mail: jmj@info.fundp.ac.be

Gian Pietro Picco
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32, 20133 Milan, Italy
E-mail: picco@elet.polimi.it

Library of Congress Control Number: 2005923585

CR Subject Classification (1998): D.2.4, D.2, C.2.4, D.1.3, F.1.2, I.2.11

ISSN 0302-9743
ISBN-10 3-540-25630-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25630-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11417019 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3454

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

Modern information systems rely increasingly on combining concurrent, distributed, mobile, reconfigurable and heterogenous components. New models, architectures, languages, and verification techniques are therefore necessary to cope with the complexity induced by the demands of today's software development. Coordination languages have emerged as a successful approach, providing abstractions that cleanly separate behavior from communication and therefore increasing modularity, simplifying reasoning, and ultimately enhancing software development.

This volume contains the proceedings of the 7th International Conference on Coordination Models and Languages (Coordination 2005), held at the Institute of Informatics of the University of Namur, Belgium, on April 20–23, 2005. The previous conferences in this series took place in Cesena (Italy), Berlin (Germany), Amsterdam (The Netherlands), Limassol (Cyprus), York (UK), and Pisa (Italy). Building upon the success of these events, Coordination 2005 provided a forum for the community of researchers interested in models, languages, and implementation techniques for coordination and component-based software, as well as applications that exploit them.

The conference attracted 88 submissions from authors all over the world. The Program Committee, consisting of 20 of the most distinguished researchers in the coordination research area, selected 19 papers for presentation on the basis of originality, quality, and relevance to the topics of the conference. Each submission was refereed by three reviewers — four in the case of papers written by a member of the Program Committee. As with previous editions, the paper submission and selection processes were managed entirely electronically. This was accomplished using ConfMan (`www.ifi.uni.no/confman/ABOUT-ConfMan/`), a free Web-based conference management system, and with the invaluable help of Paolo Costa, who installed and customized the system, ensuring its smooth operation.

We are grateful to all the Program Committee members who devoted much effort and time to read and discuss the papers. Moreover, we gratefully acknowledge the help of additional external reviewers, listed later, who reviewed submissions in their areas of expertise.

Finally, we would like to thank the authors of all the submitted papers and the conference attendees, for keeping this research community lively and interactive, and ultimately ensuring the success of this conference series.

February 2005

Jean-Marie Jacquet
Gian Pietro Picco

Organization

Program Co-chairs

Jean-Marie Jacquet	University of Namur, Belgium
Gian Pietro Picco	Politecnico di Milano, Italy

Program Committee

Farhad Arbab	CWI, Amsterdam, The Netherlands
Luca Cardelli	Microsoft Research, Cambridge, UK
Gianluigi Ferrari	University of Pisa, Italy
Paola Inverardi	University of L'Aquila, Italy
Toby Lehman	IBM Almaden Research Center, USA
Ronaldo Menezes	Florida Institute of Technology, USA
Amy L. Murphy	University of Lugano, Switzerland
Andrea Omicini	University of Bologna, Italy
George Papadopoulos	University of Cyprus, Cyprus
Ernesto Pimentel	University of Malaga, Spain
Rosario Pugliese	University of Florence, Italy
Antonio Porto	New University of Lisbon, Portugal
Carolyn Talcott	SRI International, USA
Sebastian Uchitel	Imperial College London, UK
Jan Vitek	Purdue University, USA
Michel Wermelinger	New University of Lisbon, Portugal and The Open University, UK
Herbert Wiklicky	Imperial College London, UK
Alexander Wolf	University of Lugano, Switzerland and University of Colorado, Boulder, USA
Alan Wood	University of York, UK
Gianluigi Zavattaro	University of Bologna, Italy

Steering Committee

Farhad Arbab	CWI, Amsterdam, The Netherlands
Paolo Ciancarini	University of Bologna, Italy
Rocco De Nicola	University of Florence, Italy
Chris Hankin	Imperial College London, UK

George Papadopoulos	University of Cyprus, Cyprus
Antonio Porto	New University of Lisbon, Portugal
Gruia-Catalin Roman	Washington University in Saint Louis, USA
Robert Tolksdorf	Free University of Berlin, Germany
Alexander Wolf	University of Colorado, Boulder, USA

Referees

Marco Aldinucci
Silvia Amaro
Paolo Bellavista
Michele Boreale
Roberto Bruni
Marzia Buscemi
Nadia Busi
Sonia Campa
Carlos Canal
Phil Chan
David G. Clarke
Giovanni Conforti
Maria del Mar Gallardo
Enrico Denti
David F. de Oliveira Costa
Nikolay Diakov
Manuel Diaz
Alessandra Di Pierro
Davide Di Ruscio
Daniele Falassi
Richard Ford
Luca Gardelli
Mauro Gaspari
Stefania Gnesi
Daniele Gorla
Claudio Guidi
Chris Hankin
Dan Hirsch
Jeremy Jacob
Ivan Lanese
Ruggero Lanotte
Alessandro Lapadula
Alberto Lluch-Lafuente
Michele Loreti
Roberto Lucchi
Fabio Mancinelli
Hernan Melgratti
Nicola Mezzetti
Michela Milano
Alberto Montresor
Gianluca Moro
Henry Muccini
Patrizio Pellicione
Alfonso Pierantonio
Cees Pierik
Alessandro Ricci
Bartolome Rubio
Juan Guillen Scholten
Laura Semini
Marius Silaghi
Igor Siveroni
Giuseppe Sollazzo
Ryan Stansifer
Emilio Tuosto
Izura Udzir
Leon W.N. van der Torre
Mirko Viroli
Andrew Wilkinson
Peter Zoeteweij

Lecture Notes in Computer Science

For information about Vols. 1–3355

please contact your bookseller or Springer

Vol. 3459: R. Kimmel, N. Sochen, J. Weickert (Eds.), Scale Space and PDE Methods in Computer Vision. XI, 634 pages. 2005.

Vol. 3456: H. Rust, Operational Semantics for Timed Systems. XII, 223 pages. 2005.

Vol. 3455: H. Treharne, S. King, M. Henson, S. Schneider (Eds.), ZB 2005: Formal Specification and Development in Z and B. XV, 493 pages. 2005.

Vol. 3454: J.-M. Jacquet, G.P. Picco (Eds.), Coordination Models and Languages. X, 299 pages. 2005.

Vol. 3453: L. Zhou, B.C. Ooi, X. Meng (Eds.), Database Systems for Advanced Applications. XXVII, 929 pages. 2005.

Vol. 3452: F. Baader, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XI, 562 pages. 2005. (Subseries LNAI).

Vol. 3450: D. Hutter, M. Ullmann (Eds.), Security in Pervasive Computing. XI, 239 pages. 2005.

Vol. 3449: F. Rothlauf, J. Branke, S. Cagnoni, D.W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G.D. Smith, G. Squillero (Eds.), Applications on Evolutionary Computing. XX, 631 pages. 2005.

Vol. 3448: G.R. Raidl, J. Gottlieb (Eds.), Evolutionary Computation in Combinatorial Optimization. XI, 271 pages. 2005.

Vol. 3447: M. Keijzer, A. Tettamanzi, P. Collet, J.v. Hemert, M. Tomassini (Eds.), Genetic Programming. XIII, 382 pages. 2005.

Vol. 3444: M. Sagiv (Ed.), Programming Languages and Systems. XIII, 439 pages. 2005.

Vol. 3443: R. Bodik (Ed.), Compiler Construction. XI, 305 pages. 2005.

Vol. 3442: M. Cerioli (Ed.), Fundamental Approaches to Software Engineering. XIII, 373 pages. 2005.

Vol. 3441: V. Sassone (Ed.), Foundations of Software Science and Computational Structures. XVIII, 521 pages. 2005.

Vol. 3440: N. Halbwachs, L.D. Zuck (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XVII, 588 pages. 2005.

Vol. 3439: R.H. Deng, F. Bao, H. Pang, J. Zhou (Eds.), Information Security Practice and Experience. XII, 424 pages. 2005.

Vol. 3436: B. Bouyssounouse, J. Sifakis (Eds.), Embedded Systems Design. XV, 492 pages. 2005.

Vol. 3434: L. Brun, M. Vento (Eds.), Graph-Based Representations in Pattern Recognition. XII, 384 pages. 2005.

Vol. 3433: S. Bhalla (Ed.), Databases in Networked Information Systems. VII, 319 pages. 2005.

Vol. 3432: M. Beigl, P. Lukowicz (Eds.), Systems Aspects in Organic and Pervasive Computing - ARCS 2005. X, 265 pages. 2005.

Vol. 3431: C. Dovrolis (Ed.), Passive and Active Network Measurement. XII, 374 pages. 2005.

Vol. 3429: E. Andres, G. Damiand, P. Lienhardt (Eds.), Discrete Geometry for Computer Imagery. X, 428 pages. 2005.

Vol. 3427: G. Kotsis, O. Spaniol, Wireless Systems and Mobility in Next Generation Internet. VIII, 249 pages. 2005.

Vol. 3423: J.L. Fiadeiro, P.D. Mosses, F. Orejas (Eds.), Recent Trends in Algebraic Development Techniques. VIII, 271 pages. 2005.

Vol. 3422: R.T. Mittermeir (Ed.), From Computer Literacy to Informatics Fundamentals. X, 203 pages. 2005.

Vol. 3421: P. Lorenz, P. Dini (Eds.), Networking - ICN 2005, Part II. XXXV, 1153 pages. 2005.

Vol. 3420: P. Lorenz, P. Dini (Eds.), Networking - ICN 2005, Part I. XXXV, 933 pages. 2005.

Vol. 3419: B. Faltings, A. Petcu, F. Fages, F. Rossi (Eds.), Constraint Satisfaction and Constraint Logic Programming. X, 217 pages. 2005. (Subseries LNAI).

Vol. 3418: U. Brandes, T. Erlebach (Eds.), Network Analysis. XII, 471 pages. 2005.

Vol. 3416: M. Böhlen, J. Gamper, W. Polasek, M.A. Wimmer (Eds.), E-Government: Towards Electronic Democracy. XIII, 311 pages. 2005. (Subseries LNAI).

Vol. 3415: P. Davidsson, B. Logan, K. Takadama (Eds.), Multi-Agent and Multi-Agent-Based Simulation. X, 265 pages. 2005. (Subseries LNAI).

Vol. 3414: M. Morari, L. Thiele (Eds.), Hybrid Systems: Computation and Control. XII, 684 pages. 2005.

Vol. 3412: X. Franch, D. Port (Eds.), COTS-Based Software Systems. XVI, 312 pages. 2005.

Vol. 3411: S.H. Myaeng, M. Zhou, K.-F. Wong, H.-J. Zhang (Eds.), Information Retrieval Technology. XIII, 337 pages. 2005.

Vol. 3410: C.A. Coello Coello, A. Hernández Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization. XVI, 912 pages. 2005.

Vol. 3409: N. Guelfi, G. Reggio, A. Romanovsky (Eds.), Scientific Engineering of Distributed Java Applications. X, 127 pages. 2005.

Vol. 3408: D.E. Losada, J.M. Fernández-Luna (Eds.), Advances in Information Retrieval. XVII, 572 pages. 2005.

Vol. 3407: Z. Liu, K. Araki (Eds.), Theoretical Aspects of Computing - ICTAC 2004. XIV, 562 pages. 2005.

Vol. 3406: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. XVII, 829 pages. 2005.

Vol. 3404: V. Diekert, B. Durand (Eds.), STACS 2005. XVI, 706 pages. 2005.

Vol. 3403: B. Ganter, R. Godin (Eds.), Formal Concept Analysis. XI, 419 pages. 2005. (Subseries LNAI).

Vol. 3401: Z. Li, L.G. Vulkov, J. Waśniewski (Eds.), Numerical Analysis and Its Applications. XIII, 630 pages. 2005.

Vol. 3399: Y. Zhang, K. Tanaka, J.X. Yu, S. Wang, M. Li (Eds.), Web Technologies Research and Development - APWeb 2005. XXII, 1082 pages. 2005.

Vol. 3398: D.-K. Baik (Ed.), Systems Modeling and Simulation: Theory and Applications. XIV, 733 pages. 2005. (Subseries LNAI).

Vol. 3397: T.G. Kim (Ed.), Artificial Intelligence and Simulation. XV, 711 pages. 2005. (Subseries LNAI).

Vol. 3396: R.M. van Eijk, M.-P. Huget, F. Dignum (Eds.), Agent Communication. X, 261 pages. 2005. (Subseries LNAI).

Vol. 3395: J. Grabowski, B. Nielsen (Eds.), Formal Approaches to Software Testing. X, 225 pages. 2005.

Vol. 3394: D. Kudenko, D. Kazakov, E. Alonso (Eds.), Adaptive Agents and Multi-Agent Systems III. VIII, 313 pages. 2005. (Subseries LNAI).

Vol. 3393: H.-J. Kreowski, U. Montanari, F. Orejas, G. Rozenberg, G. Taentzer (Eds.), Formal Methods in Software and Systems Modeling. XXVII, 413 pages. 2005.

Vol. 3392: D. Seipel, M. Hanus, U. Geske, O. Bartenstein (Eds.), Applications of Declarative Programming and Knowledge Management. X, 309 pages. 2005. (Subseries LNAI).

Vol. 3391: C. Kim (Ed.), Information Networking. XVII, 936 pages. 2005.

Vol. 3390: R. Choren, A. Garcia, C. Lucena, A. Romanovsky (Eds.), Software Engineering for Multi-Agent Systems III. XII, 291 pages. 2005.

Vol. 3389: P. Van Roy (Ed.), Multiparadigm Programming in Mozart/OZ. XV, 329 pages. 2005.

Vol. 3388: J. Lagergren (Ed.), Comparative Genomics. VII, 133 pages. 2005. (Subseries LNBI).

Vol. 3387: J. Cardoso, A. Sheth (Eds.), Semantic Web Services and Web Process Composition. VIII, 147 pages. 2005.

Vol. 3386: S. Vaudenay (Ed.), Public Key Cryptography - PKC 2005. IX, 436 pages. 2005.

Vol. 3385: R. Cousot (Ed.), Verification, Model Checking, and Abstract Interpretation. XII, 483 pages. 2005.

Vol. 3383: J. Pach (Ed.), Graph Drawing. XII, 536 pages. 2005.

Vol. 3382: J. Odell, P. Giorgini, J.P. Müller (Eds.), Agent-Oriented Software Engineering V. X, 239 pages. 2005.

Vol. 3381: P. Vojtáš, M. Bieliková, B. Charron-Bost, O. Sýkora (Eds.), SOFSEM 2005: Theory and Practice of Computer Science. XV, 448 pages. 2005.

Vol. 3380: C. Priami, Transactions on Computational Systems Biology I. IX, 111 pages. 2005. (Subseries LNBI).

Vol. 3379: M. Hemmje, C. Niederee, T. Risse (Eds.), From Integrated Publication and Information Systems to Information and Knowledge Environments. XXIV, 321 pages. 2005.

Vol. 3378: J. Kilian (Ed.), Theory of Cryptography. XII, 621 pages. 2005.

Vol. 3377: B. Goethals, A. Siebes (Eds.), Knowledge Discovery in Inductive Databases. VII, 190 pages. 2005.

Vol. 3376: A. Menezes (Ed.), Topics in Cryptology – CT-RSA 2005. X, 385 pages. 2005.

Vol. 3375: M.A. Marsan, G. Bianchi, M. Listanti, M. Meo (Eds.), Quality of Service in Multiservice IP Networks. XIII, 656 pages. 2005.

Vol. 3374: D. Weyns, H.V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems. X, 279 pages. 2005. (Subseries LNAI).

Vol. 3372: C. Bussler, V. Tannen, I. Fundulaki (Eds.), Semantic Web and Databases. X, 227 pages. 2005.

Vol. 3371: M.W. Barley, N. Kasabov (Eds.), Intelligent Agents and Multi-Agent Systems. X, 329 pages. 2005. (Subseries LNAI).

Vol. 3370: A. Konagaya, K. Satou (Eds.), Grid Computing in Life Science. X, 188 pages. 2005. (Subseries LNBI).

Vol. 3369: V.R. Benjamins, P. Casanovas, J. Breuker, A. Gangemi (Eds.), Law and the Semantic Web. XII, 249 pages. 2005. (Subseries LNAI).

Vol. 3368: L. Paletta, J.K. Tsotsos, E. Rome, G.W. Humphreys (Eds.), Attention and Performance in Computational Vision. VIII, 231 pages. 2005.

Vol. 3367: W.S. Ng, B.C. Ooi, A. Ouksel, C. Sartori (Eds.), Databases, Information Systems, and Peer-to-Peer Computing. X, 231 pages. 2005.

Vol. 3366: I. Rahwan, P. Moraitis, C. Reed (Eds.), Argumentation in Multi-Agent Systems. XII, 263 pages. 2005. (Subseries LNAI).

Vol. 3365: G. Mauri, G. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 415 pages. 2005.

Vol. 3363: T. Eiter, L. Libkin (Eds.), Database Theory - ICDT 2005. XI, 413 pages. 2004.

Vol. 3362: G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet, T. Muntean (Eds.), Construction and Analysis of Safe, Secure, and Interoperable Smart Devices. IX, 257 pages. 2005.

Vol. 3361: S. Bengio, H. Bourlard (Eds.), Machine Learning for Multimodal Interaction. XII, 362 pages. 2005.

Vol. 3360: S. Spaccapietra, E. Bertino, S. Jajodia, R. King, D. McLeod, M.E. Orlowska, L. Strous (Eds.), Journal on Data Semantics II. XI, 223 pages. 2005.

Vol. 3359: G. Grieser, Y. Tanaka (Eds.), Intuitive Human Interfaces for Organizing and Accessing Intellectual Assets. XIV, 257 pages. 2005. (Subseries LNAI).

Vol. 3358: J. Cao, L.T. Yang, M. Guo, F. Lau (Eds.), Parallel and Distributed Processing and Applications. XXIV, 1058 pages. 2004.

Vol. 3357: H. Handschuh, M.A. Hasan (Eds.), Selected Areas in Cryptography. XI, 354 pages. 2004.

Vol. 3356: G. Das, V.P. Gulati (Eds.), Intelligent Information Technology. XII, 428 pages. 2004.

Table of Contents

A Case Study of Web Services Orchestration

Manuel Mazzara[1] and Sergio Govoni[2]

[1] Department of Computer Science, University of Bologna, Italy
mazzara@cs.unibo.it
[2] Imaging Science and Information Systems Center, Georgetown University, Washington, DC, USA
govoni@isis.imac.georgetown.edu

Abstract. Recently the term Web Services Orchestration has been introduced to address composition and coordination of Web Services. Several languages to describe orchestration for business processes have been presented and many of them use concepts such as long-running transactions and compensations to cope with error handling. WS-BPEL is currently the best suited in this field. However, its complexity hinders rigorous treatment. In this paper we address the notion of orchestration from a formal point of view, with particular attention to transactions and compensations. In particular, we discuss $\texttt{web}\pi_\infty$, an untimed subcalculus of $\texttt{web}\pi$ [15] which is a simple and conservative extension of the π-calculus. We introduce it as a theoretical and foundational model for Web Services coordination. We simplify some semantical and pragmatical aspects, in particular regarding temporization, gaining a better understanding of the fundamental issues. To discuss the usefulness of the language we consider a case study: we formalize an e-commerce transactional scenario drawing on a case presented in our previous work [12].

1 Introduction

The aim of Web Services is to ease and to automate business process collaborations across enterprise boundaries. The core Web Services standards, WSDL [11] and UDDI [26], cover calling services over the Internet and finding them, but they are not enough. Creating collaborative processes requires an additional layer on top of the Web Services protocol stack: this way we can achieve Web Services composition and orchestration. In particular, orchestration is the description of interactions and messages flow between services in the context of a business process [23]. Orchestration is not a new concept; in the past it has been called workflow [28].

1.1 The State of the Art in Orchestration

Three specifications have been introduced to cover orchestration: Web Services Business Process Execution Language (WS-BPEL or BPEL for short) [1] which is the successor of Microsoft XLANG [25, 5] and IBM WSFL [16], together

J.-M. Jacquet and G.P. Picco (Eds.): COORDINATION 2005, LNCS 3454, pp. 1–16, 2005.

with WS-Coordination (WS-C) [29] and WS-Transaction (WS-T) [30]. BPEL is a workflow-like definition language that allows to describe sophisticated business processes; WS-Coordination and WS-Transaction complement it to provide mechanisms for defining specific standard protocols to be used by transaction processing systems, workflow systems, or other applications that wish to coordinate multiple services. Together, these specifications address connectivity issues that arise when Web Services run on several platforms across organizations.

1.2 Transactions in Web Services

A common business scenario involves multiple parties and different organizations over a time frame. Negotiations, commitments, shipments and errors happen. A business transaction between a manufacturer and its suppliers ends successfully only when parts are delivered to their final destination, and this could be days or weeks after the initial placement of the order.

A transaction completes successfully (*commits*) or it fails (*aborts*) undoing (*roll-backing*) all its past actions. Web services transactions [17] are long-running transactions. As such, they pose several problems. It is not feasible to turn an entire long-running transaction into an ACID transaction, since maintaining isolation for a long time poses performance issues [31]. Roll-backing is also an issue. Undoing many actions after a long time from the start of a transaction entails trashing what could be a vast amount of work.

Since in our scenario a traditional roll-back is not feasible, Web Services orchestration environments provide a *compensation* mechanism which can be executed when the effects of a transaction must be cancelled. What a compensation policy does depends on the application. For example, a customer orders a book from an on-line retailer. The following day, that customer gets a copy of the book elsewhere, then requests the store to withdraw the order. As a compensation, the store can cancel the order, or charge a fee. In any case, in the end the application has reached a state that it considers equivalent to what it was before the transaction started.

The notions of orchestration and compensation require a formal definition. In this paper, we address orchestration with particular attention to web transactions. We introduce $\texttt{web}\pi_{\infty}$, a subcalculus of $\texttt{web}\pi$ [15] that does not model time, as a simple extension of the π-calculus. As a case study, we discuss and formalize an e-commerce transactional scenario building on a previous one, which we presented in an earlier work [12] using a different algebra, the Event Calculus, which we introduced in [18]. The Event Calculus needed some improvement to make it more readable and easier to use for modelling real-world scenarios. This paper is a step in that direction.

1.3 Related Work

In this paper we mainly refer to BPEL, the most likely candidate to become a standard among workflow-based composition languages. Other languages have

been introduced, among them WS-CDL [14], which claims to be in some relation with the fusion calculus [22].

Other papers discuss formal semantics of compensable activities in this context. [13] is mainly inspired by XLANG; the calculus in [9] is inspired by BP-Beans [10]; the πt-calculus [8] focuses on BizTalk; [6] deals with short-lived transactions in BizTalk; [7] also presents the formal semantics for a hierarchy of transactional calculi with increasing expressiveness.

Some authors believe that time should be introduced both at the model level and at the protocols and implementation levels [15, 3, 2, 4]. XLANG, for instance, provides a notion of *timed transaction* as a special case of long running activity. BPEL uses timers to achieve a similar behavior. This is a very appropriate feature when programming business services which cannot wait forever for the other parties reply.

1.4 Outline

This work is organized as follows. In Section 2 we explain our formal approach to orchestration: extending the π-calculus to include transactions. In Section 3 we discuss this extension with its syntax and semantics, while in Section 4 we discuss an e-commerce transactional scenario to show the strength of the language. Section 5 draws a conclusion.

2 A Formal Approach to Web Services Orchestration

Business process orchestration has to meet several requirements, including providing a way to manage exceptions and transactional integrity [23]. Orchestration languages for Web Services should have the following interesting operations: sequence, parallel, conditional, send to/receive from other Web Services on typed WSDL ports, invocation of Web Services, error handling.

BPEL covers all these aspects. Its current specification, however, is rather involved. A major issue is error handling. BPEL provides three different mechanisms for coping with abnormal situations: *fault handling*, *compensation handling* and *event handling*. [1] Documentation shows ambiguities, in particular when interactions between these mechanisms are required. Therefore it is difficult to use the language, and we want to address this issue.

Our goal is to define a clear model with the smallest set of operators which implement the operations discussed above, and simple to use for application designers. We build on the π-calculus [21, 20, 24], a well known process algebra. It is simple and appropriate for orchestration purposes. It includes: a parallel operator allowing explicit concurrency; a restriction operator allowing compositionality and explicit resource creation; a recursion or a process definition operator allowing Turing completeness; a sequence operator allowing causal relationship

[1] The BPEL event handling mechanism was not designed for error handling only. However, here we use it for this purpose.

between activities; an inaction operator which is just a ground term for inductive definition on sequencing; message passing and in particular name passing operators allowing communication and link mobility.

There is an open debate on the use of π-calculus versus Petri nets in the context of Web Services composition [27]. The main reason here for using the π-calculus for formalization is that the so called *Web Services composition languages*, like XLANG, BPEL and WS-CDL claim to be based on it, and they should therefore allow rigorous mathematical treatment. However, no interesting relation with process algebras has really been proved for any of them, nor an effective tool for analysis and reasoning, either theoretical or software based, has been released. Therefore, we see a gap that needs to be filled, and we want to address the problem of composing services starting directly from the π-calculus.

By itself the π-calculus does not support any transactional mechanism. Programming complex business processes with failure handling in term of message passing only is not reasonable; also, the Web Services environment requires that several operations have transactional properties and be treated as a single logical unit of work when performed within a single business transaction. Below we consider a simple extension of the π-calculus that covers transactions.

3 The Orchestration Calculus $\mathtt{web}\pi_\infty$

The syntax of $\mathtt{web}\pi_\infty$ *processes* relies on countable sets of *names*, ranged over by $x, y, z, u, \cdots$. Tuples of names are written $\widetilde{u}$.

$$
\begin{array}{lll}
P ::= & & \\
 & \mathbf{0} & \text{(nil)} \\
 & \mid \overline{x}\,\langle \widetilde{u} \rangle & \text{(output)} \\
 & \mid x(\widetilde{u}).P & \text{(input)} \\
 & \mid (x)P & \text{(restriction)} \\
 & \mid P \mid P & \text{(parallel composition)} \\
 & \mid A(\widetilde{u}) & \text{(process invocation)} \\
 & \mid \langle\!| P \; ; \; P |\!\rangle_x & \text{(transaction)}
\end{array}
$$

We are assuming a set of process constants, ranged over by A, in order to support process definition. A defining equation for a process identifier A is of the form

$$A(\widetilde{u}) \stackrel{def}{=} P$$

where each occurrence of A in P has to be guarded, i.e. it is underneath an input prefix. It holds $fn(P) \subseteq \{\widetilde{u}\}$ and $\widetilde{u}$ is composed by pairwise distinct names.

A process can be the inert process $\mathbf{0}$, an output $\overline{x}\,\langle \widetilde{u} \rangle$ sent on a name x that carries a tuple of names $\widetilde{u}$, an input $x(\widetilde{u}).P$ that consumes a message $\overline{x}\,\langle \widetilde{w} \rangle$ and behaves like $P\{\widetilde{w}/\widetilde{u}\}$, a restriction $(x)P$ that behaves as P except that inputs and messages on x are prohibited, a parallel composition of processes, a process invocation $A(\widetilde{u})$ or a transaction $\langle\!| P \; ; \; R |\!\rangle_x$ that behaves as the *body* P until a transaction abort message $\overline{x}\,\langle\rangle$ is received, then it behaves as the *compensation* Q.

Names x in outputs and inputs are called *subjects* of outputs and inputs respectively. It is worth noticing that the syntax of $\mathtt{web}\pi_\infty$ processes simply extends the asynchronous π-calculus with the transaction process.

The input $x(\widetilde{u}).P$ and restriction $(x)P$ are binders of names $\widetilde{u}$ and x respectively. The scope of these binders is the processes P. We use the standard notions of α-equivalence, *free* and *bound names* of processes, noted $\mathtt{fn}(P)$, $\mathtt{bn}(P)$ respectively. In particular

$\mathtt{fn}(\{\!|P \ ; \ R|\!\}_x) = \mathtt{fn}(P) \cup \mathtt{fn}(R) \cup \{x\}$ and α-equivalence equates $(x)(\{\!|P \ ; \ Q|\!\}_x)$ with $(z)(\{\!|P\{^z/_x\} \ ; \ Q\{^z/_x\}|\!\}_z)$;

In the following we let $\tau.P$ be the process $(z)(\overline{z}\,\langle\rangle \mid z().P)$ where $z \notin \mathtt{fn}(P)$. $\mathtt{web}\pi_\infty$ processes considered in this paper are always *well-formed* according to the following:

Definition 1 (Well-formedness). *Received names cannot be used as subjects of inputs. Formally, in $x(\widetilde{u}).P$ free subjects of inputs in P do not belong to names $\widetilde{u}$.*

This property avoids a situation where different services receive information on the same channel, which is a nonsense in the service oriented paradigm.

3.1 Semantics of the Language

We give the semantics for the language in two steps, following the approach of Milner [19], separating the laws which govern the static relations between processes from the laws which rule their interactions. The first step is defining a static structural congruence relation over syntactic processes. A structural congruence relation for processes equates all agents we do not want to distinguish. It is introduced as a small collection of axioms that allow minor manipulation on the processes' structure. This relation is intended to express some intrinsic meanings of the operators, for example the fact that parallel is commutative. The second step is defining the way in which processes evolve dynamically by means of an operational semantics. This way we simplify the statement of the semantics just closing with respect to $\equiv$, i.e. closing under process order manipulation induced by structural congruence.

Definition 2. *The* structural congruence $\equiv$ *is the least congruence closed with respect to α-renaming, satisfying the abelian monoid laws for parallel (associativity, commutativity and $\mathbf{0}$ as identity), and the following axioms:*

1. *The scope laws:*

$$(u)\mathbf{0} \equiv \mathbf{0}, \qquad (u)(v)P \equiv (v)(u)P,$$
$$P \mid (u)Q \equiv (u)(P \mid Q), \quad \text{if } u \notin \mathtt{fn}(P)$$
$$\{\!|(z)P \ ; \ Q|\!\}_x \equiv (z)\{\!|P \ ; \ Q|\!\}_x, \quad \text{if } z \notin \{x\} \cup \mathtt{fn}(Q)$$

2. *The invocation law:*

$$A(\tilde{v}) \equiv P\{\tilde{v}/\tilde{u}\} \qquad \text{if } A(\tilde{u}) \stackrel{def}{=} P$$

3. *The transaction laws:*

$$\langle\!|\mathbf{0}\ ;\ Q|\!\rangle_x \equiv \mathbf{0}$$
$$\langle\!|\langle\!|P\ ;\ Q|\!\rangle_y \mid R\ ;\ R'|\!\rangle_x \equiv \langle\!|P\ ;\ Q|\!\rangle_y \mid \langle\!|R\ ;\ R'|\!\rangle_x$$

4. *The floating law:*

$$\langle\!|\overline{z}\,\langle\widetilde{u}\rangle \mid P\ ;\ Q|\!\rangle_x \equiv \overline{z}\,\langle\widetilde{u}\rangle \mid \langle\!|P\ ;\ Q|\!\rangle_x$$

The scope and invocation laws are standard. Let us discuss transaction and floating laws, which are unusual. The law $\langle\!|\mathbf{0}\ ;\ Q|\!\rangle_x \equiv \mathbf{0}$ defines committed transactions, namely transactions with $\mathbf{0}$ as body. These transactions, being committed, are equivalent to $\mathbf{0}$ and, therefore, cannot fail anymore. The law $\langle\!|\langle\!|P\ ;\ Q|\!\rangle_y \mid R\ ;\ R'|\!\rangle_x \equiv \langle\!|P\ ;\ Q|\!\rangle_y \mid \langle\!|R\ ;\ R'|\!\rangle_x$ moves transactions outside parent transactions, thus flattening the nesting of transactions. Notwithstanding this flattening, parent transactions may still affect children transactions by means of transaction names. The law $\langle\!|\overline{z}\,\langle\widetilde{u}\rangle \mid P\ ;\ R|\!\rangle_x \equiv \overline{z}\,\langle\widetilde{u}\rangle \mid \langle\!|P\ ;\ R|\!\rangle_x$ floats messages outside transactions; it models that messages are particles that independently move towards their inputs. The intended semantics is the following: if a process emits a message, this message traverses the surrounding transaction boundaries, until it reaches the corresponding input. In case an outer transaction fails, recovery actions for this message may be detailed inside the compensation processes. The dynamic behavior of processes is defined by the reduction relation.

Definition 3. *The* reduction relation $\rightarrow$ *is the least relation satisfying the following axioms and closed with respect to* $\equiv$, $(x)\_\ ,\ \_\ |\_$ *and* $\langle\!|\ \_\ ;\ Q|\!\rangle_x$*:*

$$\overline{x}\,\langle\widetilde{v}\rangle \mid x(\widetilde{u}).P \ \rightarrow\ P\{\widetilde{v}/\widetilde{u}\} \qquad \text{(COM)}$$

$$\overline{x} \mid \langle\!|\textstyle\prod_{i\in I} x_i(\widetilde{u_i}).P_i\ ;\ Q|\!\rangle_x \ \rightarrow\ Q \qquad (I \neq \emptyset) \qquad \text{(FAIL)}$$

Rule (com) is standard in process calculi and models input-output interaction. Rule (fail) models transaction failures: when a transaction abort (a message on a transaction name) is emitted, the corresponding transaction is terminated by garbage collecting the threads (the input processes) in its body and activating the compensation. On the contrary, aborts are not possible if the transaction is already terminated, namely every thread in the body has completed its job.

4 A Case Study

In this section, we discuss an implementation in $\mathtt{web}\pi_\infty$ of a classical e-business scenario: a customer attempts to buy a set of items from some providers, using a coordination service exposed by a web portal. Actors involved in this e-business scenario are a *customer*, a *web portal* and a set of *item providers*.