Stefan Leue
Tarja Johanna Systä (Eds.)

LNCS 3466

Scenarios: Models, Transformations and Tools

International Workshop
Dagstuhl Castle, Germany, September 2003
Revised Selected Papers

Springer

Stefan Leue Tarja Johanna Systä (Eds.)

Scenarios: Models, Transformations and Tools

International Workshop
Dagstuhl Castle, Germany, September 7-12, 2003
Revised Selected Papers

Springer

Volume Editors

Stefan Leue
University of Konstanz
Department of Computer and Information Science
78457 Konstanz, Germany
E-mail: Stefan.Leue@uni-konstanz.de

Tarja Johanna Systä
Tampere University of Technology
Institute of Software Systems
33101 Tampere, Finland
E-mail: cstasy@cs.uta.fi

Library of Congress Control Number: 2005928335

CR Subject Classification (1998): F.3.1-2, C.2.4, D.2.1, D.2.4-5, D.3.1, K.6.5

ISSN 0302-9743
ISBN-10 3-540-26189-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26189-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11495628 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3466

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

Preface

Visual notations and languages continue to play a pivotal rôle in the design of complex software systems. In many cases visual notations are used to describe usage or interaction scenarios of software systems or their components. While representing scenarios using a visual notation is not the only possibility, a vast majority of scenario description languages is visual. Scenarios are used in telecommunications as Message Sequence Charts, in object-oriented system design as Sequence Diagrams, in reverse engineering as execution traces, and in requirements engineering as, for example, Use Case Maps or Life Sequence Charts. These techniques are used to capture requirements, to capture use cases in system documentation, to specify test cases, or to visualize runs of existing systems. They are often employed to represent concurrent systems that interact via message passing or method invocation. In telecommunications, for more than 15 years the International Telecommunication Union has standardized the Message Sequence Charts (MSCs) notation in its recommendation Z.120. More recently, with the emergence of UML as a predominant software design methodology, there has been special interest in the development of the sequence diagram notation. As a result, the most recent version, 2.0, of UML encompasses the Message Sequence Chart notation, including its hierarchical modeling features. Other scenario-flavored diagrams in UML 2.0 include activity diagrams and timing diagrams.

To a large extent the attractiveness of visual scenario notations stems from the ease with which these diagrams can be recognized and understood. On the other hand, the ease of use of these diagrams brings with it the danger that system specifications and designs understate the inherent system complexity and lead to incomplete system models. A research focus is therefore directed at making scenario notations amenable to formal treatment – this includes models for their formal representations, transformations between different notations and abstraction levels, and tools that support editing, analysis and synthesis for scenario notations.

The seminar on which this proceedings volume reports was entitled *Scenarios: Models, Transformations and Tools* and was held as Seminar Number 03371 during September 7–12, 2003, at Schloss Dagstuhl, Germany. It was organized as a continuation of a series of workshops that have been co-located with larger conferences such as the International Conference on Software Engineering (ICSE) and the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOSPLA) since 2000. This volume is a post-event proceedings volume and contains selected papers based on presentations given during the seminar. All included papers were thoroughly peer-reviewed in two rounds of reviewing.

The paper by Haugen, Husa, Runde and Stølen opens the first section of papers that deal with the semantics and analysis of scenario notations. The authors of this paper argue for the need to use a three-event semantics which distinguishes the sending event, the receiving event and the consumption event in timed sequence diagrams. An interactive scenario design process by which the system synthesizes a design model by learning from sets of positive and negative scenarios, represented as sequence diagrams, is described in the paper by Harel, Kugler and Weiss. An analysis tool stands at the end of their tool chain. When analyzing Scenario specifications it is important to recognize the limits of decidability. The paper by Muscholl and Peled reviews important decidability results regarding Sequence Diagrams and Message Sequence Charts, another popular visual scenario notation. It is frequently observed that the application of modeling formalisms in specific application domains requires dedicated semantics. Cremers and Mauw propose in their paper an operational semantics for Messages Sequence Charts applied in the domain of security protocols.

One objective of the Dagstuhl seminar was to entice practical work that assesses the suitability of different scenario design approaches to a common case study. Two half-days during the seminar were devoted to modeling the case study known as the *Autonomous Shuttle System* using different design approaches and tools. The paper by Giese and Klein describe this case study. Some of the subsequent papers in this volume refer to it.

We mentioned above that many but not all scenario formalisms are visual. In his paper, Dromey introduces a textual scenario description language called Design Behavior Trees and exemplifies this design notation by application to the Early Warning System case study proposed by Harel and Politi.

The paper by Diethelm, Geiger and Zündorf offers a thorough treatment of the Autonomous Shuttle System case study using the Story Driven Modeling design approach. The CASE tool Fujaba, which underlies this study, enables editing, analysis and synthesis based on a collection of scenarios. The Use Case Maps notation has recently evolved as a new visual requirements notation that focusses on expressing the causalities of events happening along use cases. In their paper, Petriu, Amyot, Woodside and Jiang illustrate the use of the Use Case Maps notation by applying it to capturing requirements for the Autonomous Shuttle System case study.

It has long been recognized that Message Sequence Charts and related scenario notations can prove helpful in software testing. The paper by Beyer and Dulz suggests the use of collections of scenarios in the synthesis of a stochastic usage model, called Markov Chain Usage Models. These models are later used as the basis for testing stochastic properties of real-time systems.

Both the formal analysis of variants of Message Sequence Chart models and the synthesis of correct executable code from these models are at the heart of the paper by Bontemps, Heymans and Schobbens. Since both problems are either computationally expensive or intractable, the authors propose sound and complete "lightweight" approximations of the original problems. The synthesis problem is also the subject of the paper by Giese, Klein and Burmester. The

authors suggest the derivation of behavior patterns from scenario specifications. The patterns will later be used for compositional system verification.

The modeling of mobile systems is addressed in the paper by Kosiuczenko. The author suggests a graphical scenario notation to represent object mobility as an extension of UML Sequence Diagrams and suggests a semi-formal interpretation for this notation.

Message Sequence Charts are frequently used at the early stages of the software design process, and it is desirable to derive executable design models from them. The MSC2SDL tool that Khendek and Zhang describe synthesizes SDL models from collections of MSC specifications. The authors illustrate their approach by using the Autonomous Shuttle System case study as a reference.

Object-oriented systems tend to be described by the services that the object instances can provide, and often assume that an object may provide different services as it plays different rôles. The paper by Krüger and Mathews illustrates the use of Scenario Diagrams in describing the different services that object instances may provide. They also show how a complete system view can be derived from this model. The authors exemplify the use of their notation by applying it to the Center TRACON Automation System (CTAS) case study, another benchmark case study for scenario-based system design.

The collection of papers included in this volume covers a major portion of the discussions that took place during the seminar. More information, including the program, transparencies of the presentations, and a summary of the outcome of the seminar, is available online under the URL http://www.dagstuhl.de/03371/

Acknowledgements. We thank Francis Bordeleau for co-organizing this seminar with us and for helping us in the initial phases of the editing of this volume. We are truly grateful to Schloss Dagstuhl and its staff for providing us with the very pleasant atmosphere that made a very productive seminar come about. The permission to use the Springer LNCS online reviewing system helped us a lot in the compilation of this volume, and we wish to thank Tiziana Margaria and Martin Karusseit for their support.

March 2005

Tarja Systä (Tampere)
Stefan Leue (Konstanz)

Organization

Seminar Organizers

F. Bordeleau
S. Leue
T. Systä

Referees

D. Amyot
Y. Bontemps
F. Bordeleau
J.P. Corriveau
H. Giese
M. Glinz
S. Graf
R. Grosu
Ø. Haugen
K. Heljanko
F. Khendek
A. Knapp
H. Kugler
S. Leue
C. Lohr
E. Mäkinen
S. Mauw
D. Peled
I. Schieferdecker
S. Somé
T. Systä
S. Uchitel
G. Weiss
M. Woodside
A. Zündorf

Lecture Notes in Computer Science

For information about Vols. 1–3449

please contact your bookseller or Springer

Vol. 3569: F. Bacchus, T. Walsh (Eds.), Theory and Applications of Satisfiability Testing, SAT 2005. XII, 492 pages. 2005.

Vol. 3556: H. Baumeister, M. Marchesi, M. Holcombe (Eds.), Extreme Programming and Agile Processes in Software Engineering. XIV, 332 pages. 2005.

Vol. 3555: T. Vardanega, A. Wellings (Eds.), Reliable Software Technology – Ada-Europe 2005. XV, 273 pages. 2005.

Vol. 3552: H. de Meer, N. Bhatti (Eds.), Quality of Service – IWQoS 2005. XV, 400 pages. 2005.

Vol. 3547: F. Bomarius, S. Komi-Sirviö (Eds.), Product Focused Software Process Improvement. XIII, 588 pages. 2005.

Vol. 3543: L. Kutvonen, N. Alonistioti (Eds.), Distributed Applications and Interoperable Systems. XI, 235 pages. 2005.

Vol. 3541: N.C. Oza, R. Polikar, J. Kittler, F. Roli (Eds.), Multiple Classifier Systems. XII, 430 pages. 2005.

Vol. 3537: A. Apostolico, M. Crochemore, K. Park (Eds.), Combinatorial Pattern Matching. XI, 444 pages. 2005.

Vol. 3536: G. Ciardo, P. Darondeau (Eds.), Applications and Theory of Petri Nets 2005. XI, 470 pages. 2005.

Vol. 3535: M. Steffen, G. Zavattaro (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 323 pages. 2005.

Vol. 3532: A. Gómez-Pérez, J. Euzenat (Eds.), The Semantic Web: Research and Applications. XV, 728 pages. 2005.

Vol. 3531: J. Ioannidis, A. Keromytis, M. Yung (Eds.), Applied Cryptography and Network Security. XI, 530 pages. 2005.

Vol. 3528: P.S. Szczepaniak, J. Kacprzyk, A. Niewiadomski (Eds.), Advances in Web Intelligence. XVII, 513 pages. 2005. (Subseries LNAI).

Vol. 3527: R. Morrison, F. Oquendo (Eds.), Software Architecture. XII, 263 pages. 2005.

Vol. 3526: S.B. Cooper, B. Löwe, L. Torenvliet (Eds.), New Computational Paradigms. XVII, 574 pages. 2005.

Vol. 3525: A.E. Abdallah, C.B. Jones, J.W. Sanders (Eds.), Communicating Sequential Processes. XIV, 321 pages. 2005.

Vol. 3524: R. Barták, M. Milano (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. XI, 320 pages. 2005.

Vol. 3523: J.S. Marques, N.P. de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, Part II. XXVI, 733 pages. 2005.

Vol. 3522: J.S. Marques, N.P. de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, Part I. XXVI, 703 pages. 2005.

Vol. 3521: N. Megiddo, Y. Xu, B. Zhu (Eds.), Algorithmic Applications in Management. XIII, 484 pages. 2005.

Vol. 3520: O. Pastor, J. Falcão e Cunha (Eds.), Advanced Information Systems Engineering. XVI, 584 pages. 2005.

Vol. 3519: H. Li, P. J. Olver, G. Sommer (Eds.), Computer Algebra and Geometric Algebra with Applications. IX, 449 pages. 2005.

Vol. 3518: T.B. Ho, D. Cheung, H. Li (Eds.), Advances in Knowledge Discovery and Data Mining. XXI, 864 pages. 2005. (Subseries LNAI).

Vol. 3517: H.S. Baird, D.P. Lopresti (Eds.), Human Interactive Proofs. IX, 143 pages. 2005.

Vol. 3516: V.S. Sunderam, G.D.v. Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part III. LXIII, 1143 pages. 2005.

Vol. 3515: V.S. Sunderam, G.D.v. Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part II. LXIII, 1101 pages. 2005.

Vol. 3514: V.S. Sunderam, G.D.v. Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part I. LXIII, 1089 pages. 2005.

Vol. 3513: A. Montoyo, R. Muñoz, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 408 pages. 2005.

Vol. 3512: J. Cabestany, A. Prieto, F. Sandoval (Eds.), Computational Intelligence and Bioinspired Systems. XXV, 1260 pages. 2005.

Vol. 3510: T. Braun, G. Carle, Y. Koucheryavy, V. Tsaoussidis (Eds.), Wired/Wireless Internet Communications. XIV, 366 pages. 2005.

Vol. 3509: M. Jünger, V. Kaibel (Eds.), Integer Programming and Combinatorial Optimization. XI, 484 pages. 2005.

Vol. 3508: P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, M. Winikoff (Eds.), Agent-Oriented Information Systems II. X, 227 pages. 2005. (Subseries LNAI).

Vol. 3507: F. Crestani, I. Ruthven (Eds.), Information Context: Nature, Impact, and Role. XIII, 253 pages. 2005.

Vol. 3506: C. Park, S. Chee (Eds.), Information Security and Cryptology – ICISC 2004. XIV, 490 pages. 2005.

Vol. 3505: V. Gorodetsky, J. Liu, V. A. Skormin (Eds.), Autonomous Intelligent Systems: Agents and Data Mining. XIII, 303 pages. 2005. (Subseries LNAI).

Vol. 3504: A.F. Frangi, P.I. Radeva, A. Santos, M. Hernandez (Eds.), Functional Imaging and Modeling of the Heart. XV, 489 pages. 2005.

Vol. 3503: S.E. Nikoletseas (Ed.), Experimental and Efficient Algorithms. XV, 624 pages. 2005.

Vol. 3502: F. Khendek, R. Dssouli (Eds.), Testing of Communicating Systems. X, 381 pages. 2005.

Vol. 3501: B. Kégl, G. Lapalme (Eds.), Advances in Artificial Intelligence. XV, 458 pages. 2005. (Subseries LNAI).

Vol. 3500: S. Miyano, J. Mesirov, S. Kasif, S. Istrail, P. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 632 pages. 2005. (Subseries LNBI).

Vol. 3499: A. Pelc, M. Raynal (Eds.), Structural Information and Communication Complexity. X, 323 pages. 2005.

Vol. 3498: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part III. L, 1077 pages. 2005.

Vol. 3497: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part II. L, 947 pages. 2005.

Vol. 3496: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part II. L, 1055 pages. 2005.

Vol. 3495: P. Kantor, G. Muresan, F. Roberts, D.D. Zeng, F.-Y. Wang, H. Chen, R.C. Merkle (Eds.), Intelligence and Security Informatics. XVIII, 674 pages. 2005.

Vol. 3494: R. Cramer (Ed.), Advances in Cryptology – EUROCRYPT 2005. XIV, 576 pages. 2005.

Vol. 3493: N. Fuhr, M. Lalmas, S. Malik, Z. Szlávik (Eds.), Advances in XML Information Retrieval. XI, 438 pages. 2005.

Vol. 3492: P. Blache, E. Stabler, J. Busquets, R. Moot (Eds.), Logical Aspects of Computational Linguistics. X, 363 pages. 2005. (Subseries LNAI).

Vol. 3489: G.T. Heineman, I. Crnkovic, H.W. Schmidt, J.A. Stafford, C. Szyperski, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 358 pages. 2005.

Vol. 3488: M.-S. Hacid, N.V. Murray, Z.W. Raś, S. Tsumoto (Eds.), Foundations of Intelligent Systems. XIII, 700 pages. 2005. (Subseries LNAI).

Vol. 3486: T. Helleseth, D. Sarwate, H.-Y. Song, K. Yang (Eds.), Sequences and Their Applications - SETA 2004. XII, 451 pages. 2005.

Vol. 3483: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part IV. XXVII, 1362 pages. 2005.

Vol. 3482: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part III. LXVI, 1340 pages. 2005.

Vol. 3481: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part II. LXIV, 1316 pages. 2005.

Vol. 3480: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part I. LXV, 1234 pages. 2005.

Vol. 3479: T. Strang, C. Linnhoff-Popien (Eds.), Location- and Context-Awareness. XII, 378 pages. 2005.

Vol. 3478: C. Jermann, A. Neumaier, D. Sam (Eds.), Global Optimization and Constraint Satisfaction. XIII, 193 pages. 2005.

Vol. 3477: P. Herrmann, V. Issarny, S. Shiu (Eds.), Trust Management. XII, 426 pages. 2005.

Vol. 3476: J. Leite, A. Omicini, P. Torroni, P. Yolum (Eds.), Declarative Agent Languages and Technologies. XII, 289 pages. 2005.

Vol. 3475: N. Guelfi (Ed.), Rapid Integration of Software Engineering Techniques. X, 145 pages. 2005.

Vol. 3474: C. Grelck, F. Huch, G.J. Michaelson, P. Trinder (Eds.), Implementation and Application of Functional Languages. X, 227 pages. 2005.

Vol. 3468: H.W. Gellersen, R. Want, A. Schmidt (Eds.), Pervasive Computing. XIII, 347 pages. 2005.

Vol. 3467: J. Giesl (Ed.), Term Rewriting and Applications. XIII, 517 pages. 2005.

Vol. 3466: S. Leue, T.J. Systä (Eds.), Scenarios: Models, Transformations and Tools. XII, 279 pages. 2005.

Vol. 3465: M. Bernardo, A. Bogliolo (Eds.), Formal Methods for Mobile Computing. VII, 271 pages. 2005.

Vol. 3464: S.A. Brueckner, G.D.M. Serugendo, A. Karageorgos, R. Nagpal (Eds.), Engineering Self-Organising Systems. XIII, 299 pages. 2005. (Subseries LNAI).

Vol. 3463: M. Dal Cin, M. Kaâniche, A. Pataricza (Eds.), Dependable Computing - EDCC 2005. XVI, 472 pages. 2005.

Vol. 3462: R. Boutaba, K.C. Almeroth, R. Puigjaner, S. Shen, J.P. Black (Eds.), NETWORKING 2005. XXX, 1483 pages. 2005.

Vol. 3461: P. Urzyczyn (Ed.), Typed Lambda Calculi and Applications. XI, 433 pages. 2005.

Vol. 3460: Ö. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A. van Moorsel, M. van Steen (Eds.), Self-star Properties in Complex Information Systems. IX, 447 pages. 2005.

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

Vol. 3458: P. Herrero, M.S. Pérez, V. Robles (Eds.), Scientific Applications of Grid Computing. X, 208 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.

Table of Contents

Scenarios: Models, Transformations and Tools

Why Timed Sequence Diagrams Require Three-Event Semantics

Øystein Haugen[1], Knut Eilif Husa[1,2], Ragnhild Kobro Runde[1], and Ketil Stølen[1,3]

[1] Department of Informatics, University of Oslo
[2] Ericsson
[3] SINTEF ICT, Norway

Abstract. STAIRS is an approach to the compositional development of sequence diagrams supporting the specification of mandatory as well as potential behavior. In order to express the necessary distinction between black-box and glass-box refinement, an extension of the semantic framework with three event messages is introduced. A concrete syntax is also proposed. The proposed extension is especially useful when describing time constraints. The resulting approach, referred to as Timed STAIRS, is formally underpinned by denotational trace semantics. A trace is a sequence of three kinds of events: events for transmission, reception and consumption. We argue that such traces give the necessary expressiveness to capture the standard UML interpretation of sequence diagrams as well as the black-box interpretation found in classical formal methods.

1 Introduction to STAIRS

Sequence diagrams have been used informally for several decades. The first standardization of sequence diagrams came in 1992 [ITU93] – often referred to as MSC-92. Later we have seen several dialects and variations. The sequence diagrams of UML 1.4 [OMG00] were comparable to those of MSC-92, while the recent UML 2.0 [OMG04] has upgraded sequence diagrams to conform well to MSC-2000 [ITU99].

Sequence diagrams show how messages are sent between objects or other instances to perform a task. They are used in a number of different situations. They are for example used by an individual designer to get a better grip of a communication scenario or by a group to achieve a common understanding of the situation. Sequence diagrams are also used during the more detailed design phase where the precise inter-process communication must be set up according to formal protocols. When testing is performed, the behavior of the system can be described as sequence diagrams and compared with those of the earlier phases.

Sequence diagrams seem to have the ability to be understood and produced by professionals of computer systems design as well as potential end-users and stakeholders of the (future) systems. Even though sequence diagrams are intuitive – a property which is always exploited, it is not always obvious how one goes

S. Leue and T.J. Systä (Eds.): Scenarios, LNCS 3466, pp. 1–25 , 2005.

about making the sequence diagrams when a certain situation is analyzed. It is also the case that intuition is not always the best guide for a precise interpretation of a complicated scenario. Therefore we have brought forth an approach for reaching a sensible and fruitful set of sequence diagrams, supported by formal reasoning. We called this approach STAIRS – Steps To Analyze Interactions with Refinement Semantics [HS03].

STAIRS distinguishes between positive and negative traces and accepts that some traces may be inconclusive meaning that they have not yet or should not be characterized as positive or negative. STAIRS views the process of developing the interactions as a process of learning through describing. From a fuzzy, rough sketch, the aim is to reach a precise and detailed description applicable for formal handling. To come from the rough and fuzzy to the precise and detailed, STAIRS distinguishes between three sub-activities: (1) supplementing, (2) narrowing and (3) detailing.

Supplementing categorizes inconclusive behavior as either positive or negative. The initial requirements concentrate on the most obvious normal situations and the most obvious exceptional ones. Supplementing supports this by allowing less obvious situations to be treated later. Narrowing reduces the allowed behavior to match the problem better. Detailing involves introducing a more detailed description without significantly altering the externally observable behavior.

STAIRS distinguishes between potential alternatives and mandatory or obligatory alternatives. A special composition operator named xalt facilitates the specification of mandatory alternatives.

Figure 1 shows our STAIRS example – an interaction overview diagram description of the making of a dinner at an ethnic restaurant.

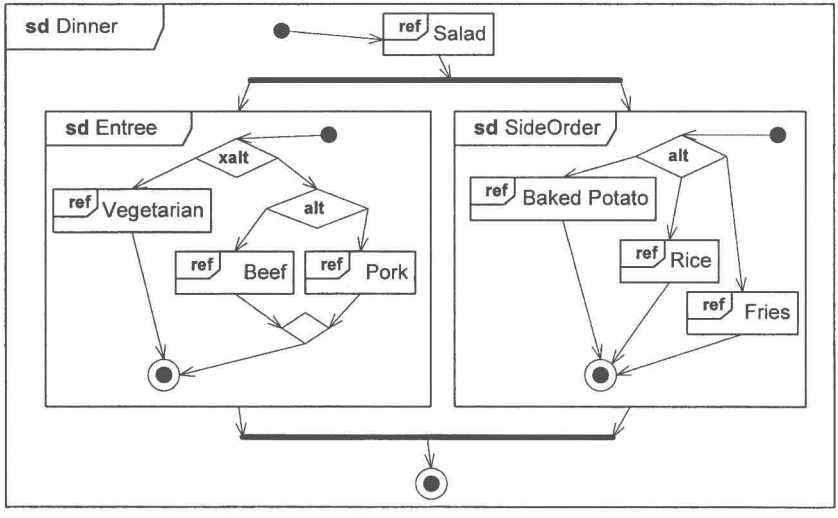

Fig. 1. Interaction overview diagram of a dinner

The dinner starts with a salad and continues with a main course that consists of an entree and a side order, which are made in parallel. For the side order there is a simple choice between three alternatives and the restaurant is not obliged to have any particular of them available. Supplementing side orders could be to offer soya beans in addition, while narrowing would mean that the restaurant could choose only to serve rice and never potatoes nor fries. It would still be consistent with the specification and a valid refinement. On the other hand, the entree has more absolute requirements. The restaurant is obliged to offer vegetarian as well as meat, but it does not have to serve both beef and pork. This means that Indian as well as Jewish restaurants are refinements (narrowing) of our dinner concept, while a pure vegetarian restaurant is not valid according to our specification.

The remainder of the paper is divided into six sections: Section 2 motivates the need for a three event semantics for sequence diagrams. Section 3 introduces the formal machinery; in particular, it defines the syntax and semantics of sequence diagrams. Section 4 defines two special interpretations of sequence diagrams, referred to as the standard and the black-box interpretation, respectively. Section 5 demonstrates the full power of Timed STAIRS as specification formalism. Section 6 introduces glass-box and black-box refinement and demonstrates the use of these notions. Section 7 provides a brief conclusion and compares Timed STAIRS to other approaches known from the literature.

2 Motivating Timed STAIRS

STAIRS works well for its purpose. However, there are certain things that cannot be expressed within the framework as presented in [HS03]. For instance time constraints and the difference between glass-box and black-box view of a system. This section motivates the need for this extra expressiveness.

Let us now look closer at the details of making the Beef in Figure 1.[1] From Figure 2 it is intuitive to assume that the working of the Cook making Beef can be explained by the following scheme: The Cook receives an order for main dish (of type Beef) and then turns on the heat and waits until the heat is adequate. Then he fetches the sirloin meat from the refrigerator before putting it on the grill. Then he fetches the sirloin from the stove (hopefully when it is adequately grilled). He then sends the steak to the customer.

We reached this explanation of the procedures of the cook from looking locally at the cook's lifeline in the Beef diagram. The input event led to one or more outputs, before he again would wait for an input. We found it natural to assume that the input event meant that the cook handled this event, consumed it and

[1] This sequence diagram is not a complete specification of Beef. The supplementing has not yet been finished. From a methodological point of view, the diagram should be "closed" with an assert when the supplementing has been finished. This to state that what is still left as inconclusive behavior should from now on be understood as negative. Otherwise, we do not get the semantics intended by Figure 1.

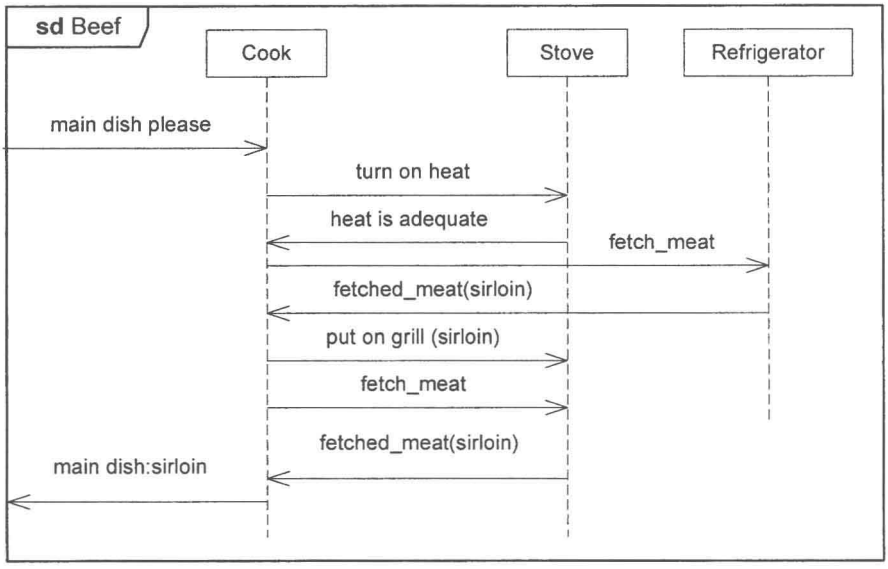

Fig. 2. Sequence diagram of Beef

acted upon it. This intuition gives rise to what we here will call the standard interpretation of sequence diagrams where an input event is seen as consumption of the event, and where the directly following output events of the trace are causally linked to the consumption. Thus, we can by considering each separate lifeline locally determine the transitions of a state machine describing the lifeline.

Our description of how the beef is made comes from a quiet day, or early in the evening when there were not so many customers and the kitchen had no problems to take care of each order immediately. Furthermore our description was probably made for one of those restaurants where the customers can look into the kitchen through glass. It was a glass-box description. We want, however, to be able to describe the situation later in the evening when the restaurant is crammed with customers and in a restaurant where there is only a black door to the kitchen. We would like to assume that even though the restaurant is full, the kitchen will handle our order immediately, but alas this is of course not the case. We can only observe the kitchen as a black-box. We observe the waiters coming through the door as messengers – orders one way and dishes the other. From these observations we could make estimates of the efficiency of the kitchen. Notice that the efficiency of the kitchen cannot be derived from when the customers placed the orders because the waiters may stop at several tables before they enter the kitchen. Comparing black-box observations of the kitchen with our glass-box one, we realize that in the glass-box description no event was attached to passing through the door. The order was sent by the customer and consumed by the chef. The passing through the door represents that the kitchen is receiving the message but not necessarily doing something with it. As long

as you are not interested in timing matters, the difference is seldom practically significant, but when time matters, the difference between when a message is received and when it is consumed is crucial. How is the kitchen organized to handle the orders in a swift and fair manner?

Motivated by this we will use three events to represent the communication of a message: the sending event, the receiving event and the consumption event, and each of these events may have a timestamp associated. We will introduce concrete syntax for sequence diagrams to capture this and the distinction is also reflected in the semantics. This will give us sufficient expressiveness to describe a black-box interpretation as well as the standard glass-box interpretation.

3 Formal Foundation

In the following we define the notion of sequence diagram. In particular, we formalize the meaning of sequence diagrams in denotational trace semantics.

3.1 Syntax of Sequence Diagrams

A message is a triple (s, tr, re) of a signal s, a transmitter tr, and a receiver re. M denotes the set of all messages. The transmitters and receivers are lifelines. L denotes the set of all lifelines.

We distinguish between three kinds of events; a transmission event tagged by an exclamation mark "!", a reception event tagged by a tilde "$\sim$", or a consumption event tagged by a question mark "?". K denotes $\{!, \sim, ?\}$.

Every event occurring in a sequence diagram is decorated with a unique timestamp. T denotes the set of timestamp tags. We use logical formulas with timestamp tags as free variables to impose constraints on the timing of events. By $\mathbb{F}(v)$ we denote the set of logical formulas whose free variables are contained in the set of timestamp tags v.

E denotes the set of all events. Formally, an event is a triple of kind, message and timestamp tag

$$E = K \times M \times T$$

We define the functions

$$k._- \in E \to K, \quad m._- \in E \to M, \quad t._- \in E \to T, \quad tr._-, re._- \in E \to L$$

to yield the kind, message, timestamp tag, transmitter and receiver of an event, respectively.

$\mathbb{N}$ denotes the set of natural numbers, while $\mathbb{N}_0$ denotes the set of natural numbers including 0.

The set of syntactically correct sequence diagrams D is defined inductively. D is the least set such that:

- $E \subset D$
- $d \in D \Rightarrow \mathsf{neg}\ d \in D \wedge \mathsf{assert}\ d \in D$