Seventh OOPSLA Workshop on Behavioral Semantics of OO Business and System Specifications

Haim Kilov
Merrill Lynch
Operations Services and Technology
World Financial Center
New York, NY 10080-6105, USA
Haim_Kilov@ml.com

Bernhard Rumpe
Institut für Informatik,
Technische Universität München,
80333 Munich, Germany
Bernhard.Rumpe@in.tum.de

Ian Simmonds
IBM T J Watson Research Center
30 Saw Mill River
Hawthorne, NY 10532, USA
simmonds@us.ibm.com

The Seventh OOPSLA workshop on behavioral semantics took place on Monday, October 19th, 1998. With 17 accepted papers of high quality, written by 41 authors, quite a few new and consolidated ideas were presented and discussed. As in previous years some prominent guests visited and actively participated in the workshop. Continuing the tradition of the six successful OOPSLA workshops (for the last two see [5,7] and two equally successful ECOOP workshops [6,8] on behavioral semantics, the workshop brought together theoreticians and practitioners to report on their experience with making semantics precise and explicit in various OO specifications. We consider that this year's workshop was another success.

The proceedings [5] of this year's workshop are available through the organizers. Contributions based upon the first four OOPSLA workshops appear in the book "Object-Oriented Behavioral Specifications" [3]. Results of the workshop and its participants are also reflected in relevant national and international (ISO) standards [1,2] (e.g., in Open Distributed Processing) and in various OMG documents [4], to which many workshop participants continue to contribute.

RULES

1.The words given to be linked together constitute a 'Doublet', the interposed words are the 'Links', and the entire series a 'Chain'. The object is to complete the Chain with the least possible number of Links.
2.Each word in the Chain must be formed from the preceding word by changing one letter in it, and one only. The substituted letter must occupy the same place, in the word so formed, which the discarded letter occupied in the preceding word, and all the other letters must retain their places.
3.When three or more words are given to be made into a Chain, the first and last constitute a 'Doublet'. The others are called 'Set Links', and must be introduced into the Chain in the order in which they are given. A Chain of this kind must not contain any word twice over.
4.No word is admissible as a Link unless it (or, if it be an inflection, a word from which it comes) is to be found in the following Glossary. Comparatives and superlatives of adjectives and adverbs, when regularly formed, are regarded as 'inflections' of the positive form, and are not given separately, e.g., the word 'new' being given, it is to be understood that 'newer' and 'newest' are also admissible. But nouns formed from verbs (as 'reader' from 'read') are not so regarded, and may not be used as Links unless they are to be found in the Glossary.

Lewis Carroll. *Doublets*.

Like in any established engineering discipline, business and system specifications sometimes are, and always should be, technical documents used to describe and understand businesses and the computer systems that support the rules of the business. Specifications have to express this understanding in a clear, precise, concise, and explicit way, in order to act as common ground between business domain experts, analysts and software developers. They also provide the basis for reuse of concepts and constructs ("patterns") common to all, or a large number of, businesses, and in doing so save intellectual effort, time and money. They introduce precision much earlier than in coding, and in a manner that allows business people – and not programmers – to supply and validate business rules.

An example of a set of business rules (for the *Doublets* game) provided by Lewis Carroll is reproduced above. These rules are clear, precise and elegant. As to conciseness, perhaps the reader can determine where these rules could have been improved. Nevertheless, if all business rules were formulated as clearly, precisely, concisely, and explicitly (and appropriately structured!) as this, perhaps our workshops would no longer be necessary! But this still has not happened.

Adequate specification approaches substantially ease the discovery of business requirements in discussion with business customers. They also support the clear separation of concerns -- between problem specification and solution design -- known since Adam Smith as division of labor. Different audiences are

interested in different aspects of "common business components" and so require different specifications.
Precise specification of semantics – as opposed to just signatures – is essential not only for business specifications, but also for business designs and system specifications. In particular, it is needed for appropriate handling of viewpoints which are essential for understanding large and even moderately sized systems, both business and computer ones. (800-page "flat" specifications are neither used nor read by anyone.) In order to handle the complexity of a (new or existing) large system, it must be considered, on the one hand, as a composition of separate viewpoints, and on the other hand, as an integrated whole, probably at different abstraction levels.
Many concepts and constructs used for all kinds of behavioral specifications – from business to systems – have common semantics and thus are good candidates for standardization and industry-wide usage. Various international standardization activities -- such as the ISO Reference Model of Open Distributed Processing, ISO General Relationship Model, OMG activities around the semantics of UML and other OMG submissions, (common) business objects, as well as the OMG semantics and reference model working groups -- are at different stages of addressing these issues.
It was therefore the aim of the workshop to bring together theoreticians and practitioners to report their experience with making semantics precise (perhaps even formal), clear, concise and explicit in OO business specifications, business designs, and system specifications. Papers varying from research (where category theory is starting to be used successfully) through academic (transfering theory into practice) and industrial "war stories" were all welcome. Experience in the usage of various (object-oriented) modeling approaches for these purposes was of special interest, as was experience in explicit traceability of semantics between a business specification, business design, and a system specification.

The topics of the workshop included:

- business specifications
- business architectures
- precise specification of semantics
- semantics of OO modeling approaches
- semantics-preserving refinement strategies
- viewpoint modelling
- standards
- business patterns (reusable fragments of specification)
- related tool support.

In the following, we present in alphabetical order our (biased and) drastically shortened overviews of the talks, as well as listing other papers that appear in the workshop proceedings [5]. Some material has been separated out to form the final conclusions presented at the end of this report.

Kenneth Baclawski, Scott A. DeLoach, Mieczyslaw Kokar, Jeffrey Smith: "Object-Oriented Parsing and Transformation".
Modern CASE tools and formal methods systems are more than just repositories of specification and design information. They can also be used for refinement and code generation. Refinement is the process of transforming one specification into a more detailed specification. Specifications and their refinements typically do not use the same specification language. Code generation is also a transformation, where the target language is a programming language. Although object-oriented programming languages and tools have been available for a long time, all refinement and transformational systems are still based on grammars and parse trees. The authors compare grammar-based transformation with object-oriented transformation and introduce a toolkit that automates the generation of parsers and transformers expressed in object-oriented terms. (Grammar-based refinement requires a great-deal of unnecessary effort. Direct OO refinement is easier, eliminating the useless parse and unparse steps.) A more specific objective is to apply these techniques to the problem of translating a CASE repository into logical theories of a formal methods system.
Question: Is this any different from structural reorganization within compilers? *Answer*: It's the same. *Question*: Can't you do the same thing for theorem proving? *Answer*: Yes.

Arne-Jorgen Berre, Tor Neple, Jack Hassall, John Eaton, Gary Gray, Mike Wilcock: "Applying ISO RM-ODP and UML in the Specification of Standard Interfaces to General Ledger Systems".
The authors were unable to attend the workshop.

J. A. Camara, J. C. Gouveia, L. F. Andrade: "Object-oriented implementation using state machines".
The authors define an architecture that enables the automatic synthesis of production code (C++) from a high level specification language (OBLOG) via an intermediate state machine model (TEJA). The specification language includes primitives that handle both business and architectural requirements. The authors' experience has profited from previous work related to a real-life project in the banking industry, where object-oriented models for large-scale projects were used. From an OBLOG specification, a TEJA

specification is produced, allowing TEJA to produce C++ code.

B.Cohen. "Being Served: The purposes, strengths and limitations of formal service modeling" [Paper not in proceedings].

The author's goal was to show formal service modeling, taking into account that we have perfectly respectable mathematical frameworks (no need to invent new ones). A particular motivation for the author is that despite four conferences on service interaction, inconsistencies arising as a result of this interaction still cannot be detected analytically, "by a system". (There exist more than 100 documented cases of undesired feature interaction, in telecommunications and elsewhere.) The author's approach it to compose business specifications with models of platforms to produce system specifications. An agent is either a user who sees only the service description or a provider. The provider sees this description and an implementation on the provider's platform (and promises to satisfy the specification). An enterprise is composed of the two. Thus, an agent is a category as a client, and a category as a supplier; and these categories have to be composed together. However, while category theory provides a good mathematical framework, it does not solve all problems -- in particular, it does not permit inconsistencies, although work e.g. by Fiadeiro is seeking to address these category-theoretical issues. So in the author's approach, formal service modeling is the problem of matching the value needs of a client to the value creation of a supplier.

Lucio Dinoto: "On supporting the separation of Business Rules from the Business Actions implementation".

The author defined business rules as business constraints that an organization establishes as conditions for its business actions or activities. They are to be specified "separately from the object model". Logical conditions or programming rules (such as "password must be longer than 5 characters") must not be confused with business rules. Also, business rule implementation must not be hard-coded within the implementation of business actions because it does not make the object model reusable for other contexts. Instead of this, a dynamic association is required between business action implementation and business rule execution.

Zinovy Diskin: "The arrow logic of meta specifications: a formalized graph-based framework for structuring schema repositories".

The author was unable to attend the workshop.

Janusz A. Dobrowolski, Edward Morton: "Use of the Virtual Finite State Machine for the Behavioral Specification of Complex Business Systems".

Too often, the (initial) behavioral specification of a system becomes an outdated document since the code takes precedence over the specification. To deal with this, the authors propose to generate code automatically, and state that manually accessing the generated code should be forbidden and impossible. The authors use virtual finite state machines – a notation that is programming-language-, operating-system- and database-independent – to represent "active business object" behavior. (A state of a business object is distinguished from a state of its data.) This approach has been used to generate millions of lines of code. The models are very elementary – for example, there is no nesting. Although a single model may contain hundreds of states, the authors note that models with more than 20 or 30 states tend to be hard to understand.

Question: You stated that in your approach reliability increases as complexity increases. This is counterintuitive. *Answer*: This happens since the larger the model, the more manually written code gets replaced with automatically generated code.

Marc Frappier, R. St-Denis: "Specifying Information Systems using Input-Output Traces and JSD Entities".

The authors describe a formal method based on input-output traces to specify the behavior of information systems. Input traces are described using entities defined by regular expressions and process algebra operators, in a precise style inspired by the Jackson System Development method. Outputs are specified using axioms on input sequences, taking advantage of the process structure defining the input sequences. Preconditions are taken care of by ordering Entity Structure Diagrams.

Radu Grosu, Manfred Broy, Bran Selic, Gheorge Stefanescu: "Towards a Calculus for UML-RT Specifications".

The unified modeling language (UML) is currently being tuned (by Rational Software Corporation and ObjecTime Limited) for real-time applications in the form of a new proposal -- UML for Real-Time (UML-RT). Because of the importance of UML-RT the authors are investigating its formal foundation in a joint project between ObjecTime Limited, Technische Universität München and the University of Bucharest. The paper deals with a part of this foundation, namely the theory of flow-graphs. The emphasis is on providing good semantics, which has to be both understandable and mathematically precise. An approach similar to category theory is used as a foundation of this approach.

Pavel Hruby: "Structuring Specification of Business Systems with UML".
The use of Unified Modeling Language (UML) for modeling business systems is intended to enhance communication between software developers, domain experts and other professionals with different backgrounds. However, UML does not specify how to structure information describing the business system, nor does it specify which diagrams to include in the design artifacts or what the relationships between various artifacts are. The author described a pattern of four design artifacts that can be used for description of business systems with UML. The pattern is based on artifacts that describe classifier relationships, interactions, responsibilities and state machines. The application of the pattern can easily be extended to cover information about the business system in different views and at different levels of abstraction.

Shusaku Iida, Kokichi Futatsugi, Razvan Diaconescu: "Component Based Algebraic Specifications".
The authors propose a new specification style called "component based algebraic specification" (CBAS) which supports formal component based software construction as well as software architecture and design patterns (since testing huge and complex systems is hard and expensive, verification techniques become more important). CBAS allows the specification not only a component's interfaces but also its behavior. Also, it can specify the architecture of the system built from these components. The approach is realized in CafeOBJ – an executable algebraic specification language (available for download) – for which the semantics of composition is precisely defined. CafeOBJ supports many sorted algebra, order sorted algebra, hidden algebra, and rewriting logic. In particular, it becomes possible to reuse not only code, but also proofs of behavioral equivalence. CBAS has been applied to prove the correctness of adding a component within the context of the ODP Trading function.

Haim Kilov, Allan Ash: "The Business of Automating".
Systems fail because (1) business specifications don't exist; (2) system specifications don't connect to business specifications, leading to the perception that "code then appears magically". The *realization* relationship provides traceability between business specifications, business design; system specifications; and system implementations. The (most important aspect of the) business of automating is the business of writing system specifications. The authors described a(n accounting) part of a real-life project at Merrill Lynch, in which there was a need to interface to legacy systems treated (pragmatically) as black boxes. Not surprisingly, you can do a system specification in exactly the same way as a business specification – based on the specification of the (business or system) domain. Partial specifications of (pre- and postconditions) for important operations are often sketched when the appropriate fragments of the domain are specified. A high-tech solution is not necessarily the most cost-effective for operations. It was found to be very helpful, during business design, to generate a number of low-tech, manual, or "status quo" variants for realizing some operations. (When the accounting books don't satisfy the invariant then "reconciliation" occurs, which may require manual compensation).
Question: At one far end there's a program that runs. What is the moment when a non- in principle executable spec becomes an in principle executable spec? *Answer*: the relationships between the system objects are in the system specification. Nothing is (automatically) generated. *Question*: How do you know that you've done it correctly? *Answer*: There is a step of invention that you can check by verifying that the result satisfies (eg) the business invariant. *Question*: What percentage of operations are specified in terms of pre- and postconditions? *Answer*: At the top level, most. In many cases the operation specifications follow from the invariants.

D. Muthiayen, V.S. Alagar: "A UML-based Methodology for Real-Time Reactive System Development".
In developing a modeling technique for Real-Time Reactive Systems, the authors introduce a minimal set of extensions to the UML notation. A class stereotype defines a classifier for a generic model describing reactive entities. An association stereotype describes responses to stimuli in the form of timing constraints on reactions to events. To define the behavioral semantics of objects described in the extended notation, the authors extend (one of several specific interpretations of) OCL (Object Constraint Language) with specific predicates capturing temporal properties. The semantics represents groundwork for a methodology for formal development of reactive systems of industrial scale. The authors also outlined a verification methodology founded on formal semantics and PVS (Prototype Verification System) for automated reasoning.

Bernhard Rumpe, Veronika Thurner: "Refining Business Processes".
The authors present a calculus for refinement of business process models, based on a precise definition of business processes and process nets. Business process models are a vital concept for communicating

with experts of the application domain. Depending on the roles and responsibilities of the application domain experts involved, process models are discussed on different levels of abstraction. These may range from detailed rules of process execution to strategic level interrelations of basic core processes. To ensure consistency and to allow for a flexible integration of process information on different levels of abstraction, refinement rules are introduced that allow the incremental addition to and refinement of the information in a process model, while maintaining the validity of more abstract high level processes. In particular, the authors define rules for the decomposition of single processes and logical data channels, as well as the extension of the interface and channel structure to include information that is newly gained or increases in relevance during the modeling process. *Question*: Why is this specific for business? Can't it also be used for other kinds of notation? *Answer*: the rules were adapted from other calculi the authors defined on data-flow based software architectures and on state-machines.

Vishal Sikka, Martin King: "The International Effort to Standardize Conceptual Schema Modeling".
The authors describe the standards work being done by ISO for Conceptual Schema Modeling Facilities (CSMF). The history of the ISO work goes back to 1977 and is based on the ANSI/SPARC report on Conceptual Schema. A major milestone was the availability in 1982 of the report "Concepts and Terminology for the Conceptual Schema and the Information Base", subsequently published by ISO as TR 9007. The current ISO work is towards an actual CSMF standard, and has recently seen significant progress towards a final committee draft. A major aspect of this work is related to defining the technical foundations of the standard. The authors provide a general overview of the standards process and describe the formal basis of the CSMF, as well as the set of constructs that comprise it.

Richard Sinnott, Mario Kolberg: "Business-Oriented Development of Telecommunication Services".
The development of software for distributed systems, e.g. telecommunication services, is a complex activity. Interface definition languages (IDLs) often used for these systems allow only for the specification of the syntactic aspects of the interfaces. IDLs are insufficient since they lack both behavior and abstraction – in other words, they "lose the big picture". The authors show how these issues are being addressed in the TOSCA project in its development of a service creation and validation environment. Behaviour is specified using SDL based on collections of "elementary" parameterized business patterns. Thus, these OO frameworks allow semantics to be captured and re-used. They describe "nearly finished services", in which the user needs to intervene in "points of flexibility" and specialize behavior (using graphical tools) to provide services.

Angelo E. Thalassinidis, Ira Sack: "On the Specification of Organizational Information Processing Requirements".
The authors were unable to attend the workshop.

Hei-Chia Wang, V. Karakostas: "Business Object Linking Using An Event Description Language and Business Rules".
The authors present an approach for combining business rules with an event language to enable the automatic linking of business objects in a distributed environment. Since message sending is an overspecification, object communication is based on business rules that act upon events rather than messages. Business objects contain control rules used for determining whether to accept an event request and how to handle this request in order to produce another event. Generated events are used for dynamically linking objects within a distributed system.

Conclusions

The workshop drew up the following conclusions, which were accepted by all participants:

- Different kinds of specifications (business, "system," technological) can be written using the same concepts and constructs (eg RM-ODP) and have a common foundation
- We are able to rely on existing mathematics (eg category theory)
- We should avoid reinvention, and make extensions only if needed (eg for dealing with inconsistencies)
- It is possible to reuse not only code, but specifications, proofs, and other things, leading to reusable precise patterns
- We should formulate specifications in terms of invariants ("more important and fundamental" than operations)
- Specifications have to be explicit; and mathematics can and should be used to do that
- We should try to develop systems using "higher level" constructs than those of conventional programming languages
- Practitioners (developers) are helped by mathematically precise semantics, which improves their tools and their communications with each other and with customers

- Specifications are needed in order to analyze and predict properties of the system that is specified; and to discover and understand flaws in the customer's theory of the world
- An abstract specification may be detailed

Workshop participants

Kenneth Baclawski, Northeastern University, kenb@ccs.neu.edu
Bernard Cohen, City University, London, UK, b.cohen@city.ac.uk
Lucio Dinoto, Lifia & JP Morgan, dinoto_lucio@jpmorgan.com
Janusz Dobrowolski, Lucent Technologies, jdobrowolski@lucent.com
Lenny Estrin, MetLife, lestrin@metlife.com
Marc Frappier, Universite de Sherbrooke, marc.frappier@dmi.usherb.ca
Kokichi Futatsugi, Japan Advanced Institute of Science and Technology (JAIST), kokichi@jaist.ac.jp
Joao Gouveia, Oblog Software SA, jgouveia@oblog.pt
Gunter Graw, University of Dortmund, graw@ls4.informatik.uni-dortmund.de
Radu Grosu, TU Munchen, grosu@in.tum.de
Trevor Hopkins, IBM UK OTP, trevor_hopkins@uk.ibm.com
Shusaku Iida, Japan Advanced Institute of Science and Technology (JAIST), s_iida@jaist.ac.jp
Haim Kilov, Merrill Lynch, haim_kilov@ml.com
Mitch Kokar, Northeastern University, kokar@coe.neu.edu
Chris Marshall, SES Software, chris_marshall@sesh.com
Tom Mowbray, Blueprint Technology, mowbray@www.serve.com
Darmalingum Muthiayen, Condordia University, d_muthi@cs.concordia.ca
Gianna Reggio, University of Genova, Italy; reggio@disi.unige.it
Bernhard Rumpe, TU Munich, rumpe@in.tum.de
Richard St-Denis, University de Sherbrooke, richard.st-denis@dmi.usherb.ca
Bran Selic, ObjecTime Limited, bran@objectime.com
Ian Simmonds, IBM TJ Watson Research Center, simmonds@us.ibm.com
Richard Sinnott, GMD Fokus, sinnott@fokus.gmd.de
Jeff Smith, Sanders / Northeastern University, jsmith@coe.neu.edu
Christopher Spottiswoode, Metaset, cms@metaset.co.za
Veronica Thurner, Technical University of Munich, thurner@in.tum.de
Hei-Chia Wang, University of Manchester Institute of Science and Technology (UMIST), hcwang@co.umist.ac.uk

References

1. ISO/IEC. Open Distributed Processing - Reference Model: Part 2: Foundations (IS 10746-2 / ITU-T Recommendation X.902, February 1995).
2. ISO/IEC. Information Technology - Open Systems Interconnection - Management Information Systems - Structure of Management Information - Part 7: General Relationship Model, ISO/IEC 10165-7, 1995.
3. *Object-oriented behavioral specifications*, edited by Haim Kilov and Bill Harvey, Kluwer Academic Publishers, 1996, ISBN 0-7923-9778-9.
4. OMG Semantics Working Group Green Paper. OMG Document number ormsc/97-06-10r (Haim Kilov and Kevin P. Tyson).
5. Object-Oriented Programming Languages and Applications (OOPSLA'98), *Seventh OOPSLA Workshop on Behavioral Semantics Business and System Specifications*, Vancouver, Canada, October 19th, 1998. Editors Haim Kilov, Bernhard Rumpe and Ian Simmonds; Proceedings of Munich University of Technology, TUM-I9820, August 1998
6. European Conference on Object-Oriented Programming (ECOOP'97), *Workshop on Precise Semantics for Object-Oriented Modeling Techniques*; Jyväskylä, Finland, June 9th, 1998. Editors Haim Kilov and Bernhard Rumpe; Proceedings of Munich University of Technology, TUM-I9725, Mai 1997
7. Object-Oriented Programming Languages and Applications (OOPSLA'97), *Workshop on Object-Oriented Behavioral Semantics (with an Emphasis on Semantics of Large OO Business Specifications)*, Atlanta, USA, October 5th, 1997. Editors Haim Kilov, Bernhard Rumpe and Ian Simmonds; Proceedings of Munich University of Technology, TUM-I9737, September 1997
8. European Conference on Object-Oriented Programming (ECOOP'98), Second ECOOP *Workshop on Precise Behavioral Semantics*; Brussels, Belgium, July 24th, 1998. Editors Haim Kilov and Bernhard Rumpe; Proceedings of Munich University of Technology, TUM-I9813, June 1998