Tutorial

LNCS 4710

Chris W. George
Zhiming Liu
Jim Woodcock (Eds.)

Domain Modeling and the Duration Calculus

International Training School
Shanghai, China, September 2007
Advanced Lectures

Springer

Chris W. George
Zhiming Liu
Jim Woodcock (Eds.)

Tutorial

LNCS 4710

Domain Modeling and the Duration Calculus

International Training School
Shanghai, China, September 2007
Advanced Lectures

Springer

Lecture Notes in Computer Science 4710

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

Chris W. George Zhiming Liu
Jim Woodcock (Eds.)

Domain Modeling and the Duration Calculus

International Training School
Shanghai, China, September 17-21, 2007
Advanced Lectures

Springer

Volume Editors

Chris W. George
Zhiming Liu
United Nations University
International Institute for Software Technology
P.O.Box 3058, Macau SAR, China
E-mail:{cwg, z.liu}@iist.unu.edu

Jim Woodcock
University of York
Department of Computer Science
Heslington, York YO10 5DD, UK
E-mail: jim@cs.york.ac.uk

Library of Congress Control Number: Applied for

CR Subject Classification (1998): F.3, D.2.11, D.2.4, D.2.2, F.2.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-74963-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-74963-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12124022 06/3180 5 4 3 2 1 0

Preface

This volume contains a record of the lectures given at the , held during the 17th–21st September 2007 in Shanghai. The was organised by East China Normal University, UNU-IIST, and the University of York as part of the celebrations of the 70th birthdays of Dines Bjørner and Zhou Chaochen. There were two associated events:

- . Papers presented at a Symposium held in Macao on 24th & 25th September 2007. LNCS volume 4700. Springer 2007.
- . Held in Macao during 26th–28th September 2007. LNCS volume 4711. Springer 2007.

The is aimed at postgraduate students, researchers, academics, and industrial software engineers who are interested in the state of the art in these topics. No previous knowledge of the topics involved is assumed. Two of the courses are in the area of domain engineering (and in formal, abstract modelling in general) and two are in the area of duration calculus; the fifth links the two areas. The five courses are taught by experts in these fields from Europe and Asia.

We are happy to acknowledge sponsorship from the following organisations:

- China International Talent Exchange Foundation
- East China Normal University
- United Nations University International Institute for Software Technology
- University of York

The proceedings were managed and assembled using the EasyChair conference management system.

Contributors

Alan Burns is a professor of computer science at the University of York. His research interests are in real-time systems, including the assessment of real-time programming languages, distributed operating systems, the formal specification of scheduling algorithms and implementation strategies, and the design of dependable user interfaces to real-time applications.

Dang Van Hung is a research fellow of UNU-IIST. He received a doctoral-level degree in computer science in 1988 from the Computer and Automation Research

Institute, Hungarian Academy of Sciences. His research interests include formal techniques of programming, concurrent and distributed computing, and design techniques for real-time systems.

Chris George is the Associate Director of the United Nations International Institute for Software Technology (UNU-IIST) in Macao. He is one of the main contributors to RAISE, particularly the RAISE method, and that remains his main research interest. Before coming to UNU-IIST he worked for companies in the UK and Denmark.

Michael Reichhardt Hansen is an associate professor at the Technical University of Denmark. His research interests include duration calculus, interval logic, and formal methods. He is one of the authors of the book [illegible] with Zhou Chaochen.

Cliff Jones was a professor at the University of Manchester, worked in industry at Harlequin for a period, and is now a professor of computing science at Newcastle University. He is Editor-in-Chief of the [illegible]. He undertook the DPhil at Oxford University Computing Laboratory under Prof. Sir Tony Hoare frs, awarded in 1981. He worked with Dines Bjørner and others on the Vienna Development Method (VDM) at IBM in Vienna. He is a Fellow of the Royal Academy of Engineering.

Lecture Courses

Course 1: Delivering Real-Time Behaviour. This series of lectures is given by Alan Burns, and it focuses on how to engineer systems so that they can meet their timing requirements. Four separate, but related, issues are addressed.

1. A time band model that caters for the broad set of granularities found in a typical complex system.
2. The delay and deadline statements that allow timing requirements to be specified.
3. Scheduling analysis that enables a set of concurrent deadlines to be verified.
4. Timing analysis that enables sequential code to be inspected to determine its worst case behaviour.

These four topics—together with a number of other techniques and tools described in the course—allow real-time behaviour to be delivered.

Course 2: Applicative Modelling with RAISE. This course—given by Chris George—provides an introduction to the RAISE Specification Language and to the RAISE method. The course concentrates on the applicative style of RAISE, the style most commonly used initially in development. It also describes two examples. The first is a simple communication system that allows the transmission of messages with the possibility of higher priority messages overtaking others. The example illustrates the use of abstract initial specification to capture

vital properties, and of more detailed concrete specification to describe a model having those properties. The second example is a control system of a lift and illustrates the use of model checking to gain confidence in a RAISE model.

Course 3: A Theory of Duration Calculus with Application. This course is given jointly by Dang Van Hung and Michael Hansen. It presents selected central elements in the theory of the and gives examples of applications. The lectures cover syntax, semantics, and a proof system for the basic logic. Results on decidability, undecidability, and model-checking are also presented. Some extensions of the basic calculus are described; in particular, hybrid duration calculus and duration calculus with iterations. The concepts are illustrated by a case study: the bi-phase mark protocol. References are provided for further study.

Course 4: Understanding Programming Language Concepts via Operational Semantics. Cliff Jones's lectures cover five topics.

1. **History of Verification.** This is based on his paper [Jon03]; this lecture adds more on semantics.
2. **Rely/Guarantee Method.** The most accessible reference for this is [Jon96] but the origins lie a long way back [Jon81,Jon83a,Jon83b] (see the extensive list of publications on various forms of rely/guarantee conditions at `homepages.cs.ncl.ac.uk/cliff.jones/home.formal`).
3. **Deriving Specifications.** This lecture is described in the accompanying volume [JHJ07]; there is an earlier conference paper [HJJ03]).
4. **Semantics of Programming Languages.** This lecture is published in this volume. Chris George covers the idea of abstract modelling in general; Cliff Jones focuses on the application of this idea to programming languages.
5. **Soundness of Rely/Guarantee Proof Rules.** This final lecture justifies a set of proof rules like those introduced in Lecture 2 based on a semantics like that in Lecture 4. The proof is published in [CJ07]. This material links to "Refining Atomicity" [JLRW05,BJ05,Jon05,Jon07].

July 2007 J. C. P. W.

References

[BJ05] Burton, J.I., Jones, C.B.: Investigating atomicity and observability. Journal of Universal Computer Science 11(5), 661–686 (2005)

[CJ07] Coleman, J.W., Jones, C.B.: Guaranteeing the soundness of rely/guarantee rules (revised). Journal of Logic and Computation (in press, 2007)

[HJJ03] Hayes, I., Jackson, M., Jones, C.: Determining the specification of a control system from that of its environment. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 154–169. Springer, Heidelberg (2003)

[Jon81] Jones, C.B.: Development Methods for Computer Programs including a Notion of Interference. PhD thesis, Oxford University, June 1981 Printed as: Programming Research Group, Technical Monograph 25 (1981)

[Jon83a] Jones, C.B.: Specification and design of (parallel) programs. In: Proceedings of IFIP 1983, pp. 321–332. North-Holland, Amsterdam (1983)

[Jon83b] Jones, C.B.: Tentative steps toward a development method for interfering programs. ACM Transactions on Programming Languages and Systems 5(4), 596–619 (1983)

[Jon96] Jones, C.B.: Accommodating interference in the formal design of concurrent object-based programs. Formal Methods in System Design 8(2), 105–122 (1996)

[Jon01] Jones, C.B.: On the search for tractable ways of reasoning about programs. Technical Report CS-TR-740, Newcastle University, Superceded by (2001)

[Jon03] Jones, C.B.: The early search for tractable ways of reasoning about programs. IEEE, Annals of the History of Computing 25(2), 26–49 (2003)

[Jon05] Jones, C.B.: An approach to splitting atoms safely. In: Electronic Notes in Theoretical Computer Science, MFPS XXI, 21st Annual Conference of Mathematical Foundations of Programming Semantics, pp. 35–52 (2005)

[Jon07] Jones, C.B.: Splitting atoms safely. Theoretical Computer Science 357, 109–119 (2007)

[JHJ07] Jones, C., Hayes, I., Jackson, M.A.: Specifying systems that connect to the physical world. In: Essays in Honour of Dines Bjørner and Zhou Chaochen on the Occasion of the 70th Birthdays. Papers presented at a Symposium held in Macao on 24th & 25th September 2007. LNCS, vol. 4700, Springer, Heidelberg (2007)

[JLRW05] Jones, C.B., Lomet, D., Romanovsky, A., Weikum, G.: The atomicity manifesto (2005)

Coordinating Committee

Chris George	UNU-IIST
He Jifeng	East China Normal University
Zhiming Liu	UNU-IIST
Geguang Pu	East China Normal University
Jim Woodcock	University of York
Yong Zhou	East China Normal University

Table of Contents

Delivering Real-Time Behaviour

Alan Burns and Andy Wellings

Real-Time Systems Research Group
Department of Computer Science
University of York, UK
{burns,andy}@cs.york.ac.uk

Abstract. This paper focuses on how we can engineer systems so that they can meet their timing requirements. Four separate, but related, issues are addressed: a time band model that caters for the broad set of granularities found in a typical complex system, the delay and deadline statements that allow timing requirements to be specified, scheduling analysis that enables a set of concurrent deadlines to be verified and timing analysis that enables sequential code to be inspected to determine its worst case behaviour. These four topics together with a number of other techniques and tool described in the paper allow real-time behaviour to be delivered.

1 Introduction

In the construction of real-time systems it is vital to ensure that timing requirements are satisfied by the system under development. To do this requires a number of different techniques that must be integrated into an engineering process[13]. In this paper we support the rigorous verification of timing requirements by proposing an engineering process and populating it with existing/modified methods such as model checking, schedulability analysis and timing analysis. The development of large computer-based systems, with embedded components, imposes a number of significant challenges, both technical and organisational. Their complexity makes all stages of their development (requirements analysis, specification, design, implementation, deployment and maintenance/evolution) subject to failure and costly re-working. Even the production of an unambiguous behavioural description of an existing system is far from straightforward.

The process discussed here by which real-time behaviour is delivered comes from the synergy of many existing methods and proposals. It is not entirely formal but is strongly influenced by the need to engineer real systems with industrial strength tools and methods. The key dimensions of the process are:

1. Time bands – to situate the proposed system in a finite set of distinct time scales.
2. Delay and Deadline Primitives – to capture timing requirements in each band.
3. Scheduling analysis – to manage the resources needed at each band to ensure the system makes appropriate progress (i.e. meets its deadlines).
4. Timing analysis – to ensure activities defined within a single band have a bounded resource requirement.

These four dimensions are supported by

C. George, Z. Liu, and J. Woodcock (Eds.): Domain Modeling, LNCS 4710, pp. 1–50, 2007.

- A modelling and verification formalism based on a restricted use of Timed Automata in which timing requirements within an automaton are represented by *delay* and *deadline* conditions.
- A program model that utilises common pattern to implement the require behaviour – typical patterns being periodic and sporadic processes, consumer/ producer relations and shared objects. The program model can be realised in languages such as Spark [26].

One characteristic of computer-based systems is that they are required to function at many different time scales (from microseconds or less to hours or more). Time is clearly a crucial notion in the specification (or behavioural description) of computer-based systems, but it is usually represented, in modeling schemes for example, as a single flat physical phenomenon. Such an abstraction fails to support the structural properties of the system, forces different temporal notions on to the same flat description, and fails to support the separation of concerns that the different time scales of the system facilitate. Just as the functional properties of a system can be modeled at different levels of abstraction or detail, so too should its temporal properties be representable in different, but provably consistent, time scales.

To make better use of 'time', with the aim of producing more dependable embedded systems, we propose a framework that explicitly identifies a number of distinct *time bands* in which the system under study is situated [11,10]. Within each time band, timing requirements are represented by delay and deadline primitives. Delay ensures the technical system does not 'get ahead' of its environment; deadlines ensure the system does not get too far behind. The key role of the implementation (as well as obvious functional correctness) is to satisfy the deadline constraints. To examine these constraints, the sequential code must be amenable to timing analysis and the concurrent system amenable to scheduling analysis.

The four dimensions identified above are addresses in the four main sections of this paper. First, time bands are motivated and then described. Next timing requirements within each bands are considered using delays and deadlines. Then schediling analysis is outlined and finally a brief review of timing analysis is given. Conclusions are provided in section 6.

2 Time Bands

The aim of this section of the paper is to motivate a modeling framework in which a multi-banded representation of time is advocated. Much of this material is necessarily focused on an informal description of the framework. A brief discussion on the formalisation of the framework in provided in a later section (2.8).

The framework enables the temporal properties of existing systems to be described and the requirements for new or modified systems to be specified. The concept of time band comes from the work of Newell [43] in his attempts to describe human cognition. Newell focuses on hierarchical structures within the brain and notes that different time scales are relevant to the different layers of his hierarchy. By contrast, we put the notion of a time band at the centre of our framework. It can then be used within any

organisational scheme or architectural form — for they all lead to systems that exhibit a wide variety of dynamic behaviours.

2.1 Informal Description of the Framework

The domain of any large computer-based system exhibits dynamic behaviour on many different levels. The computational components have circuits that have nanosecond speeds, faster electronic subcomponents and slower functional units. Communication on a fast bus is at the microsecond level but may be tens of milliseconds on slow or wide-area media. Human time scales as described above move from the 1ms neuron firing time to simple cognitive actions that range from 100ms to 10 seconds or more. Higher rational actions take minutes and even hours. At the organisational and social level, time scales range from a few minutes, through days, months and even years. Perhaps for some environmentally sensitive systems, consequences of failure may endure for centuries. To move from nanoseconds to centuries requires a framework with considerable descriptive and analytical power.

Most formulations that attempt to identify time granularity do so by mapping all activities to the finest granularity in the system. This results in cumbersome formulae, and fails to recognise the distinct role time is taking in the structuring of the system. An exception is the work of Corsetti *et al*[21,16]; they identify "*a finite set of disjoint and differently grained temporal domains*". Their framework is not as extensive as the one developed here, but they do show how the notion of temporal domains can be embedded into a logical specification language. We are not aware of any other work that uses the existence of distinct time scales as the basis of system modeling.

2.2 Definition of a Band

A band is represented by a granularity (expressed as a unit of time that has meaning within the band) and a precision that is a measure of the accuracy of the time frame defined by the band. The precision of a band defines the tolerance over the requirements for two or more events to occur simultaneously. System activities are placed in some band B if they engage in significant events at the time scale represented by B. They have dynamics that give rise to changes that are observable or meaningful in band B's granularity. So, for example, at the nanosecond band, gates are firing; at the 10 millisecond band, human neural circuits are firing, significant computational functions are completing and an amount of data communication will occur. At the five minute band, work shifts are changing, meetings are starting, etc. For any system there will be a highest and lowest band that gives a temporal system boundary — although there will always be the potential for larger and smaller bands. Note that at higher bands the physical system boundary may well be extended to include wider (and slower) entities such as legislative constraints or supply chain changes.

Time has both discrete and continuous characteristics within the framework. Both are needed to capture the essential properties of complex systems; the term *hybrid system* is often used to indicate this dual need. A time band defines a temporal frame of reference (e.g., a clock that *ticks* at the granularity of the band) into which discrete actions can easily be placed. But continuous entities can also be placed in this band if they exhibit significant observable events on this time scale. For these entities, time is continuous

but significant events occur at a frequency of no more than (but close to) once per 'tick' of the band's abstract clock.

By definition, all activities within band B have similar dynamics. Within any modeling framework there is considerable advantage in assuming that actions are instantaneous. They represent behaviours that are atomic; the combined behaviour of a number of concurrent yet atomic actions is easy to assert as there is no interference between behaviours. However in real-time embedded systems it is also necessary to consider the real duration of actions. Within a band, *activities* have duration whilst *events* are instantaneous — "take no time in the band of interest". Many activities will have a repetitive cyclic behaviour with either a fixed periodicity or a varying pace. Other activities will be event-triggered. Activities are performed by agents (human or technical). In some bands all agents will be artificial, at others all human, and at others both will be evident. The relationship between the human agent and the time band will obviously depend on the band and will bring in studies from areas such as the psychology of time [27,28,47] and the sociology of time [39]. Embedded software will populate a number of bands, the execution time of a single instruction will denote one band, the completion of distinct unit funtions are best described at another band, and complete schedulable tasks will typically be mapped to yet another band.

In the specification of a system, an event may cause a response 'immediately' – meaning that at this band the response is within the granularity of the band. This helps eliminate the problem of over specifying requirements that is known to lead to implementation difficulties [33]. For example, the requirement 'when the fridge door opens the light must come on immediately' apparently give no scope for an implementation to incorporate the necessary delays of switches, circuitry and the light's own latency. By making the term 'immediate' band specific, it enables a finer granularity band to include the necessary delays, latencies and processing time that are needed to support the immediate behaviour at the higher band.

Events that are instantaneous at band B map to activities that have duration at some lower band with a finer granularity – we will denote this lower band as C. A key property of a band is the precision it defines for its time scale. This allows two events to be simultaneous ("at the same time") in band B even if they are separated in time in band C. This definition of precision enables the framework to be used effectively for requirements specification. A temporal requirement such as a deadline is band-specific; similarly the definition of a timing failure. For example, being one second late may be a crucial failure in a computing device, whereas on a human scale being one second late for a meeting is meaningless. The duration of an activity is also 'imprecise' (within the band). Stating that a job will take three months is assumed to mean plus or minus a couple of days. Of course the precision of band B can only be explored in a lower band.

From a focus on band B, two adjacent bands are identified. The slower (broader) band (A) can be taken to be unchanging (constant) for most issues of concern to B (or at least any activity in band A will only exhibit a single state change during any activity within band B). At the other extreme, behaviours in (the finer) band C are assumed to be instantaneous. The actual differences in granularity between A, B and C are not precisely defined (and indeed may depend on the bands themselves) but will typically be in the range 1/10th to 1/100th. When bands map on to hierarchies (structural or control)