

WATIS²: Design and Application of an Environment Simulation System for Test Improvement of Control Software for Automatic Logistic Systems

Christian Kreiner, Christian Steger, Reinhold Weiss

Institute for Technical Informatics

Graz University of Technology

Inffeldgasse 16, A-8010 Graz, Austria

{kreiner,steger,rweiss}@iti.tu-graz.ac.at

Abstract

Control computer systems for automatic logistic systems contain reactive modules, which are usually tested on-site, in conjunction with the real environment. This leads to high test costs and unsatisfactory test coverage. Environment simulation models offer an execution environment for customizable, in-house testing. Architecture and process model of WATIS², a flexible, efficient and expandable environment simulation system for automatic logistic systems are presented. Experiments in an industrial environment show a high model reuse rate and a significant improvement of baseline software development regarding software quality and project cost, in this way justifying a systematic application of WATIS².

1. Introduction

Central control computer systems of automatic logistic systems are complex software systems implementing a wide variety of functions like database, user interface, and reactive modules controlling underlying transport facilities. After investigation of the peculiarities of software development in this domain, environment simulation promises a major improvement of the software process of the reactive software parts. The idea of environment simulation is to substitute the physical plant by a simulation model of it (Fig. 1), in this way raising test and product quality, and saving costs of on-site stays as well.

The architecture of WATIS², a flexible and efficient environment simulation system for systematic application in the software development of automatic logistic systems is presented, together with a process model for integration into the development process of the baseline projects.

The results of WATIS² application experiments in industrial projects are presented, showing an efficiency of

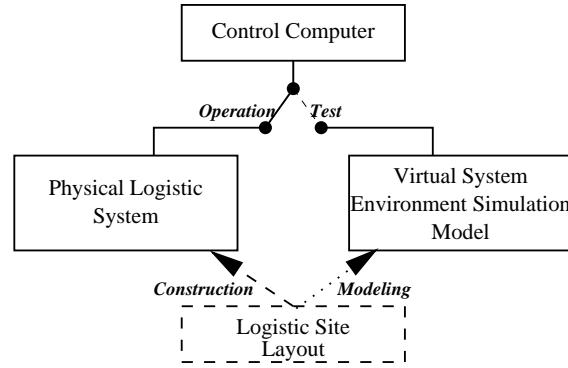


Figure 1. Environment simulation for software testing

model generation and a significant improvement of baseline project development in terms of time and quality. Taken together, the systematic application of WATIS² seems technically as well as economically justified.

2. Application domain and motivation

2.1. Automatic logistic systems

Automatic logistic systems typically consist of storage components (racks) and a set of automatic transport means like conveyors, lifting units and mobile subsystems like rail-guided stacker cranes, AGVs (automatic guided vehicles), etc. These components often contain a local PLC (programmable logic controller) which controls transport unit movements, component operation mode, local transport job activation. Additionally, these controllers provide a communication interface to a coordinating control computer system.

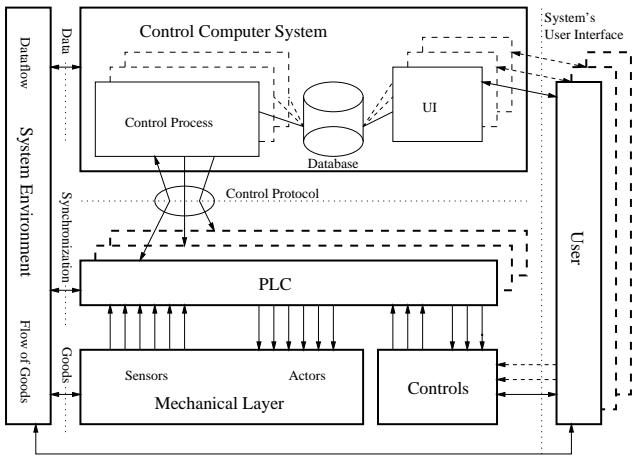


Figure 2. Control architecture of automatic logistic systems

Control computer functions include database services, user interfaces, coupling to enterprise computing systems, and, of course, coordination of the transport facilities (controlling, tracking, synchronization, reaction to failures), as well as global throughput optimization (transport scheduling, warehouse management strategies like adaptive storage allocation and priority policies).

To sum up, in the automatic logistic systems domain we find distributed, hierarchical systems, which are embedded in the customer's flows of goods and data (Fig. 2).

2.2. Software lifecycle of control computer systems

Control system software for automatic logistic systems is typically developed according to a defined lifecycle model conforming to a quality management system like ISO 9000 [12]. Variants of the well known waterfall model are frequently used.

It turns out that, due to on-site activities, test and installation phases consume a large share of the total software project budget. Close to 50 % have been reported for an application of a "standardized software system" [15] (project duration 1 year until installation, 20 % installation, plus half a year tuning and adaption, no explicit data for test). This is well at the upper margin of test and installation efforts for software in general (15-50 % [2]).

High effort for test and installation appears to be systematic for software in this application domain for various reasons [17]. These are partly related to the required software functionality:

Software Reuse. Control functions depend heavily on the site layout and customer needs. They can be regarded

uniquely for each project. Transport facilities and vehicles are often customized as well. This leads to a rather low software reuse factor of 20-30 % [7] in control computer software development which again increases the effort spent on testing.

Reactive Software Parts. The control processes are reactively coupled with the underlying automatic facilities. Testing of these software parts depends heavily on the availability of the latter. As software and transport equipment is often developed concurrently, targeting at a project deadline, testing of reactive functions often has to be postponed to a late phase in the project. Furthermore, it introduces a discontinuity to the development process, as from the test phase onward, project evolution continues on-site. Even steps back to earlier life-cycle phases, especially to implementation phase, occur on-site, in an expensive environment mostly not well suited to software development.

Load Tests. Test and tuning of throughput optimizations require the basic control of underlying facilities to be fully functional and an operating condition of the warehouse system near maximum load. Besides long test runs, presenting a high load in terms of goods flow is difficult to achieve and often impossible in preproduction phases.

Long Settling Time. Test and tuning of (probably adaptive) warehouse management strategies require even longer "test" runs and are almost certainly only possible in the productive environment: settling to a stable operating condition requires some time in the magnitude of a complete turnaround of goods in the warehouse. Even worse, the most likely situation to see in the installation phase, the initial filling of a warehouse, definitely is a transient situation.

Another set of causes arises out of the reality of project management:

Limited Testing Time. As already stated, exclusive tests in conjunction with the real facilities are only possible for a short time at a late stage in the project run. This period then is often only sufficient to test and verify protocols, tracking of goods, basic control and reactions to equipment failures, in this way leading to poor test coverage. Extensive regression testing is normally far beyond the available time budget.

Large Scale. Due to the distributed nature of logistic systems it is not easy to overlook all parts simultaneously.

Slow Physical Movements. Physical transport movements are slow compared to control computer reactions. This often implies long waiting time before an interesting test case can be watched.

Risks. Undoubtedly there are risks for goods and even workers during on-site test and installation phases.

For their high contribution to project costs, the test and installation phases are the most promising candidates for improvements regarding cost and time savings, as well as quality improvements in control computer software development.

2.3. Environment simulation

The introduction of environment simulation seems to be a pragmatic choice to improve especially the test and installation phases in logistic control software development. The idea of reactively coupling the computer system under test to an executable model of its environment (Fig. 1) to enable certain test scenarios is not new. Myers [20] reports environment simulation useful for load tests, test scenarios, which are difficult to establish in reality, and systems tests of highly safety critical systems like nuclear reactors, etc.

Regarding the characteristics of software development for automatic logistic systems, environment simulation promises improvements of test and installation by

- shifting on-site test activities to an earlier, in-house phase, in this way significantly cutting expensive on-site stays,
- better test coverage due to earlier and more intensive tests resulting in better software quality,
- faster transport movements enabling tests of optimization and long term adaptive strategies,
- easier arrangement of test situations,
- better insight to system behavior through visualization and animation.

In this logistic biased software engineering area, environment simulation has been used punctually with a variety of objectives and at different levels:

Fastabend [6] uses a production control system in conjunction with the corresponding environment simulation for staff training with hands-on and what-if experience.

Unthank and Fletcher [25] use a model of a logistic center to test a decision support system which designs the high-level interface between the logistic center and other company systems.

Ottjes and Hogedoorn [21] use simulation of a multi-AGV system to develop and test a control strategy, and they reuse their model to control and monitor the AGVs in real-time.

Prinz and Liebe [22] use environment simulation for layout optimization, control system demonstration (marketing) and testing of customer-specific extensions of a control system for a flexible production system.

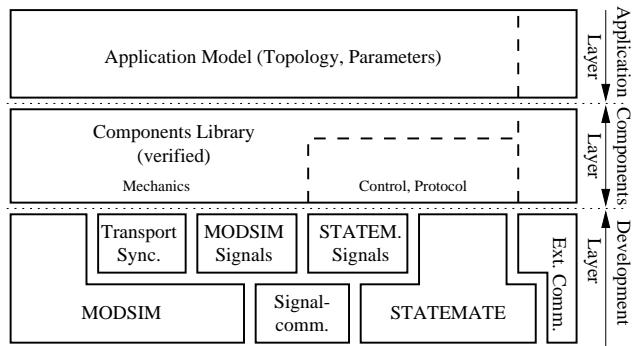


Figure 3. WATIS² modeling layers

Most of these environment simulators are highly specialized systems and were primarily used for a single project. A similar situation can be found in other areas like the development of products for the mass market (e.g. Honka's mobile phone hardware simulator [11]), and in the development of large, complex systems like nuclear reactors, etc.

Thus, flexibility and expansibility of these environment simulation systems probably were intended, but have not been main project goals. The economic aspect of environment simulation systems, in particular the modeling effort, is hardly ever addressed.

3. The environment simulation system WATIS²

The environment simulation system WATIS² (Warehouse Test and Integration Simulation System) has been designed for systematical application in and integration into the logistic control system development. In this context, not only correctness and accuracy of WATIS² models are necessary, but also an economic justification has to be given. Hence, the requirements flexibility, expansibility and modeling efficiency have to be added. Additionally, a process model has to be defined, which integrates simulation related activities and control system development.

3.1. Simulation model architecture and process

In order to meet the stated requirements, WATIS² uses a layered model architecture consisting of an application layer, a component layer, and a development layer (Fig. 3, [18]). These layers determine not only the internal architecture of WATIS² models, but also the integrated modeling process (Fig. 4, [17]) and the modeler's role in the different layers.

A "logistic domain expert" builds *application models* by combining instances of verified, parameterized model com-

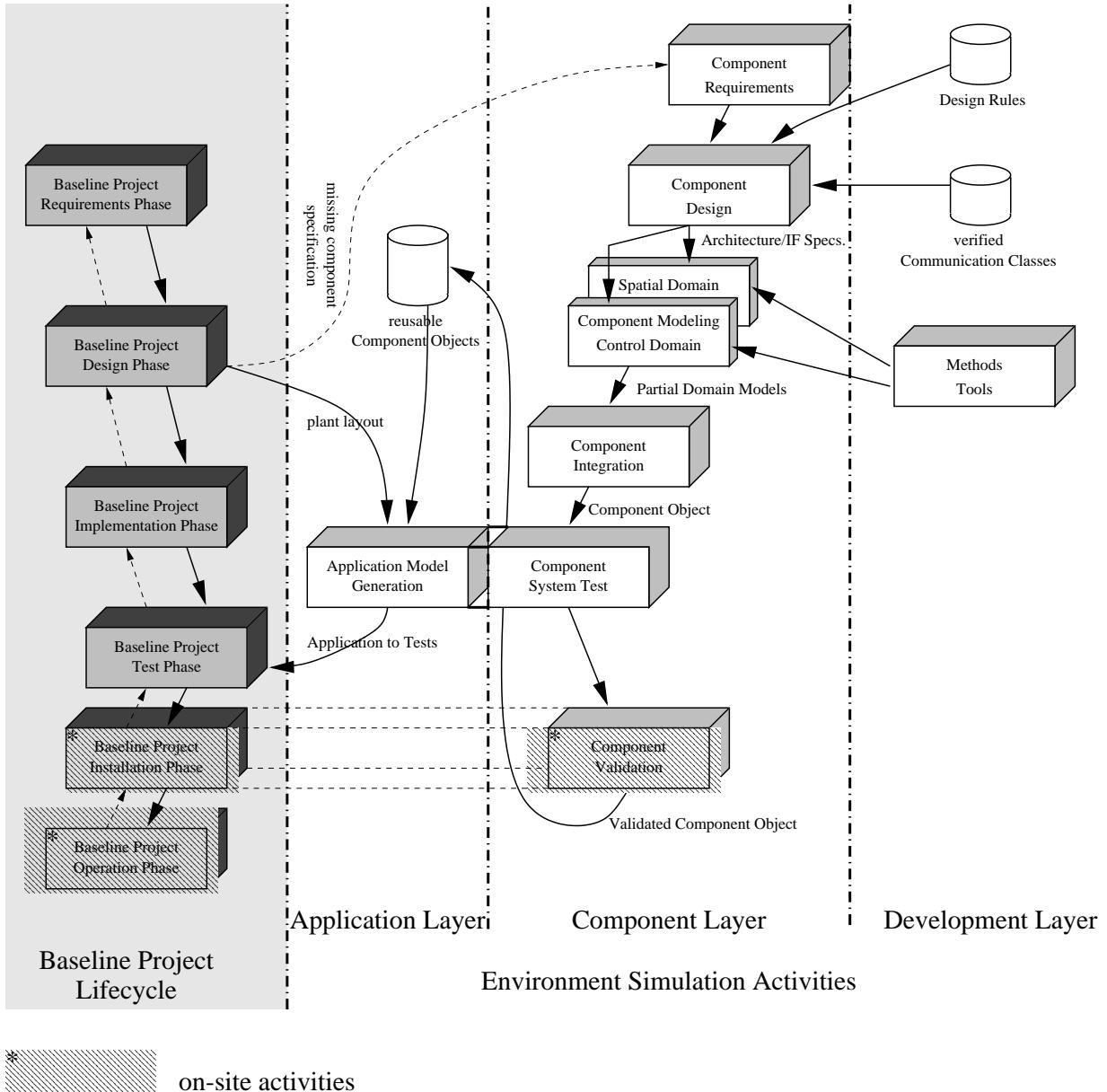


Figure 4. WATIS² integrated process model

ponent objects according to the planned site layout, here-with mimicking the construction of the physical logistic system (c.f. Fig. 1). These components model the behavior of high-level logistic domain objects like stacker cranes, conveyor segments, etc.

New *components* are developed by a "simulation expert" according to the path shown in Fig. 4, starting from the component specification. Components model transport, control and communication behavior, and include visualization and communication interface to the control computer. As the component structure (Fig. 5, notation from [23]) indi-

cates, communication objects with predefined communication patterns are used for all external and inter-component interfaces (transport synchronization, etc.). This guarantees smooth component combination in the application layer [18]. Component validation has to be postponed to the installation phase and later, because the corresponding real-life entity is only available from then on. **WATIS²** component objects are collected in a library for reuse in further application models. With this approach, the component pool can be easily extended on demand (c.f. Fig. 4).



on-site activities

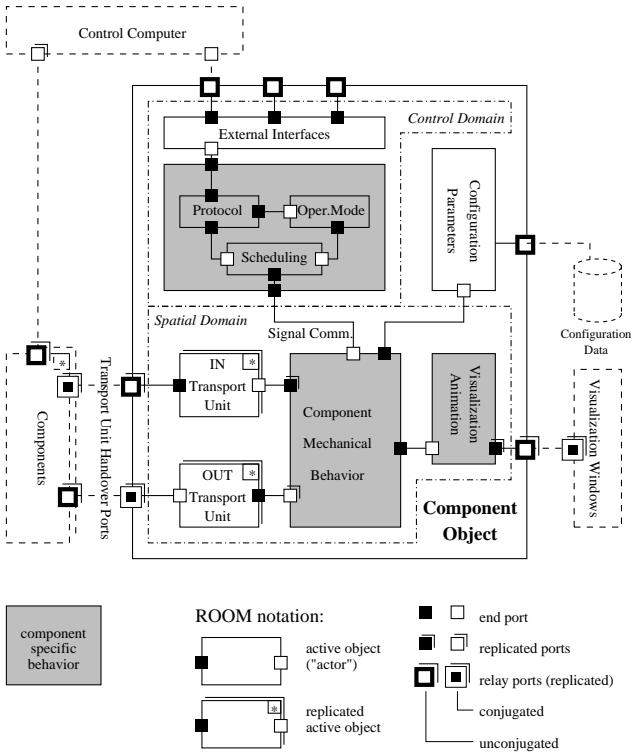


Figure 5. WATIS² component structure

The *development layer* provides infrastructure for the mentioned activities, e.g. tool support, design rules and verified communication classes for coupling components, modeling tools, as well as models and control systems.

3.2. Modeling domains and tools

In order to support modeling and verification, control aspects and transport/storage aspects are separated and modeled with different, domain-adapted methods and tools. The model parts then are connected asynchronously through a tool- and platform-independent cosimulation interface [16].

For *spatial domain* modeling the discrete event simulation language MODSIM [4] has been chosen. MODSIM can be characterized as modular, object-oriented programming and simulation language together with a simulator kernel, integrated visualization and animation.

Control domain functions include communication protocols, transport job scheduling, operation mode management, etc. A high-level graphical formalism, statecharts (an extension to finite state machines, [9]), is used for the description of behavior in this domain. The tool Statemate [8, 10] implements the statecharts specification technique and adds activitycharts for functional structuring. Static (syntax, type and consistency checking) and dynamic veri-

fication (execution, chart animation, variable tracing, racing condition detection) is supported.

4. WATIS² application and experience

First experiments have been conducted in cooperation with an industrial partner (Salomon Automation Ges.m.b.H, 8114 Stübing, Austria). In the initial phase, selected components used in automated pallet logistics – racks, stacker cranes and conveyors – have been modeled as WATIS² components. These components have then been used for building five different application models so far.

4.1. Modeled components

Modeled *rack* elements offer a number of parameters for rack structure, physical dimensions and coordinates. Rack elements act as storage for pallet objects which in turn have content and size attributes. Snapshots during simulation runs are possible, so that certain situations can be restored later easily.

Stacker cranes feature structural parameters determining number of lifts, number of forks per lift and pallet positions per fork, where each position can hold one transport unit. These subsystems can be further specified by static and dynamic mechanical parameters like coordinates, size, speed, acceleration,etc.

Tracks for the (rail-guided) stacker cranes are formed from a combination of linear segments, bends and switches.

Conveyor components comprise a catalog of linear segments, crossings, turntables, etc. They all use the same basic communication classes for the synchronization of transport unit hand-over (Fig. 5) to ensure that combinations are cooperating correctly.

As a very first encouraging result, in the course of capturing the protocol and scheduling behavior of the stacker cranes with the statecharts formalism, inconsistent and missing items in their specification have been identified and led to their correction.

4.2. Application models

Five WATIS² application models have been built so far to study modeling flexibility, accurateness, efficiency and impact to the control system's development:

Project A consists of an approx. 6000 pallet warehouse, 2 stacker cranes with 2 lifts each on a common track, and an I/O conveyor system. This was the first model built when its physical counterpart was already in operation. Its primary use was to gain experience with

	Application Model				Components		Reuse
Group	Plant	Conveyors	Racks/Cranes	Protocol	reused	extensions	
Type	Configuration		generated	C-code	MODSIM+ STATEMATE		estimated %
Project							
A	120	500	13000	150	10000	100	91
B	80	0	12500	150	10000	0	97
C	50	0	60000	150	10000	80	97
D	50	400	6500	150	10000	150	93
E	20	50	0	150	10000	20	97

Table 1. Model construction efforts and estimated reuse factor (approximate number of code resp. configuration lines)

the *WATIS*² framework, develop the mentioned components and validate them through comparison with the existing reality.

Project B has similar size and characteristics, but features a rail system with loops and a more complex conveyor system.

An environment simulation model was available for use on a voluntary basis during control system development.

Project C has a FIFO oriented multi-channel storage rack with different channel lengths with a total capacity of 20000 pallets with 3 separated stacker cranes.

Again, the model was voluntarily used in the baseline software project.

Project D is a 3-crane warehouse with a complex conveyor system. Environment simulation was employed in software testing in a planned way and integrated according to Fig. 4.

Project E was modeled primarily for layout planning purposes of a performance critical conveyor system. In the case of realization of this project it is intended to reuse this application model.

For a rough comparison of these projects and an estimation of the reuse factor in *WATIS*² models, the figures in Table 1 have been collected. As a common basis for this estimation, the size metric "line counts" has been adopted together with heuristic conversion rules for code lines, statecharts, and configuration lines [14].

Global plant configuration, conveyor configuration and small C-code adaptions for binary telegram converters as well as extensions to existing components were counted as application specific modeling effort, whereas existing components make up the reused portion of the models. Configuration of racks and stacker cranes is automated and therefore has not been taken into account at all when calculating

an effort-focused reuse factor. For the five models A to E, Table 1 then shows an estimated reuse factor greater than 90 % in each case.

Experience has shown that, with the use of the existing *WATIS*² component library, a complete warehouse model can be configured in 1-3 days, depending on size and complexity. Conveyors take approximately twice as long to model as racks and cranes.

4.3. Impact on control software lifecycle

To study and measure the impact of the presented method to the baseline software development process, software process metric acquisition and presentation system has been developed [13] using the Goal-Question-Metric (GQM) method of Basili and Rombach [1]. Much effort has been put into automatic raw metric collection and the metric presentation in the company's intranet. The metric set includes:

- durations of baseline project development,
- software change count per module, phase and project,
- simulation usage count and duration.

Table 2 shows a collected phase duration metrics for nine automatic logistic software projects. Tests of projects B, C, D have been supported by *WATIS*² models, projects U to Z had no environment simulation support. Whereas most of the latter typically show on-site time around 50 % of the total project duration, all of the *WATIS*² supported projects are well below 30 %, apparently due to better software maturity in the on-site phases. Shorter on-site stays can be also regarded as substantial overall savings, since environment simulation model building took a few days only - in terms of total project durations less than 3 %.

Project	Phase Duration (days)						On-Site %
	In-House Requirements + Design	Coding	Test	On-Site Installation	Support	Total	
U	11	76	4	35	34	160	43
V	32	35	63	18	404	552	76
W	94	59	32	42	175	402	54
X	142	47	56	53	212	510	52
Y	35	86	76	143	118	458	57
Z	133	21	35	42	11	242	22
B	210	44	103	10	69	436	18
C	173	94	39	36	69	411	26
D	31	102	11	42	12	198	27

Table 2. Phase durations and on-site time of logistic software development projects

5. Conclusion and further work

The software lifecycle of a control computer for automatic logistic systems has been characterized and test and installation phases have been found to contribute large amounts to the total project costs. By connecting a logistic environment simulation model to the control computer under development, significant improvements are expected. The *WATIS*² framework, its architecture and process model for the efficient and flexible generation of accurate application models has been presented. Experimental application in an industrial software development environment suggest a high model reuse of greater than 90 %, resulting in very efficient model generation. Impacts on the baseline project development include a significant reduction of expensive on-site stays due to earlier, environment simulation enabled testing and better software quality. In this way, the overhead of model generation is economically justified as well.

Further work will deal with the extension of the *WATIS*² component library, a graphical model building tool, and refinement and extension of metric acquisition and presentation tool developed in the course of this project.

References

- [1] V. R. Basili and H. D. Rombach. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.
- [2] B. W. Boehm. *Software Engineering Economics*. Advances in Computing Science and Technology. Prentice Hall, 1981.
- [3] F. Breitenecker and I. Husinsky, editors. *Proceedings of the 1995 EUROSIM Conference, EUROSIM '95, Vienna, Austria, Sep. 11-15, 1995*, The Netherlands, 1995. Elsevier.
- [4] CACI Products Company. *MODSIM II. The Language for Object-Oriented Programming. Reference Manual*. CACI Products Company, 3333 North Torrey Pines Court, La Jolla, CA 92037, 1995.
- [5] G. Chroust, editor. *EUROMICRO Conference 1999, Workshop "Software Process and Product Improvement"*, Milan, Italy, Sep. 8–10, 1999, Los Alamitos, CA, 1999. IEEE Computer Society.
- [6] H. Fastabend. Anwendung der Simulation zur Mitarbeiterausbildung in der Fertigungssteuerung. In [24], pages 255–258, 1993.
- [7] G. Fellrieser. Improving Software Quality using CASE Methods ; ESSI Project No. 21358 (ISUC). In 8. Austrian ENCRESS Workshop, Dec. 1996.
- [8] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. Trakhtenbrot. Statemate: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403–414, Apr. 1990.
- [9] D. Harel and A. Naamad. The STATEMATE Semantics of Statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.
- [10] D. Harel and M. Politi. *Modeling Reactive Systems with Statecharts: The Statemate Approach*. I-Logix, 1996.
- [11] H. Honka. A simulation-based approach to testing embedded software. Technical Report VTT Publications 124, Technical Research Centre of Finland, Espoo, Finland, 1992.
- [12] International Organization for Standardization, Geneva, Switzerland. *ISO 9000-3: Quality management and quality assurance standards; Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software*, 1991.
- [13] P. Jocham and C. Kreiner. A Lean Metric Acquisition and Presentation Environment for the Assessment of a Test Process Improvement Experiment. In [5], pages 274–278, 1999.
- [14] P. Jocham, C. Kreiner, and H. Schrittwieser. Improving Test of Automatic Logistic Systems using Environment Simulation. In *CONQUEST'99: Conference on Quality Engineering in Software Technology, Nürnberg, Sep. 27–29, 1999*, pages 234–242. Arbeitskreis Software-Qualität Franken e.V., 1999.
- [15] H. Klumpp. Informations- und Datenverarbeitung in einem Logistikzentrum für Elektrogeräte. In *Kommissionieren in Industrie und Handel*, VDI Berichte ; 1131, pages 63–71, 1994.
- [16] C. Kreiner and C. Steger. Modellkopplung STATEMATE – MODSIM. In *5. Deutsches Anwenderforum für STATEMATE, Aschheim*, Apr. 21–22, 1997, D-85521 Ottobrunn, 1997. Berner & Mattner.
- [17] C. Kreiner, C. Steger, and R. Weiss. Improvement of Control Software for Automatic Logistic Systems using Executable Environment Models. In G. Chroust, editor, *EUROMICRO Conference 1998, Workshop "Software Process and Product Improvement"*, Västerås, Sweden, Aug. 25–27, 1998, pages 919–923, Los Alamitos, CA, 1998. IEEE Computer Society.

- [18] C. Kreiner, C. Steger, and R. Weiss. Model-Based Verification of Real-Time Software for Automatic Logistic Systems. In K. Juslin, editor, *EUROSIM '98. European Simulation Congress, Helsinki, Finland, Apr. 14–15, 1998*, pages 560–564, Espoo, Finland, 1998. VTT.
- [19] K. Mertins and M. Rabe, editors. *Erfahrungen aus der Zukunft: 8. ASIM-Fachtagung Simulation in Produktion und Logistik, Berlin, Feb. 16-17, 1998*, D-10587 Berlin, Pascalstr. 8-9, 1998. Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK).
- [20] G. J. Myers. *The Art of Software Testing*. John Wiley & Sons, 1979.
- [21] J. A. Ottjes and F. P. A. Hogedoorn. Design and Control of Multi-AGV Systems. Reuse of Simulation Software. In *Simulation in Industry, 8th European Simulation Symposium ESS'96*, pages 461–465, Ghent, Belgium, 1996. SCS.
- [22] J. Prinz and P. Liebe. Integration von Simulationssoftware mit Leitsystemen. In [19], pages 63–71, 1998.
- [23] B. Selic, G. Gullekson, and P. T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley Sons, 1994.
- [24] A. Sydow, editor. *Simulationstechnik, 8. Symposium in Berlin, Fortschritte in der Simulationstechnik Bd. 6*. F. Vieweg & Sohn, 1993.
- [25] G. Unthank and E. J. Fletcher. Using a simulation model to test the functionality of a Decision Support System which designs the interface between a logistic centre and other company systems. In [3], pages 1083–8, 1995.