Requirements Engineering

Ein persönlicher Rückblick und Ausblick

A Personal Reflection

Martin Glinz

www.ifi.uzh.ch/rerg

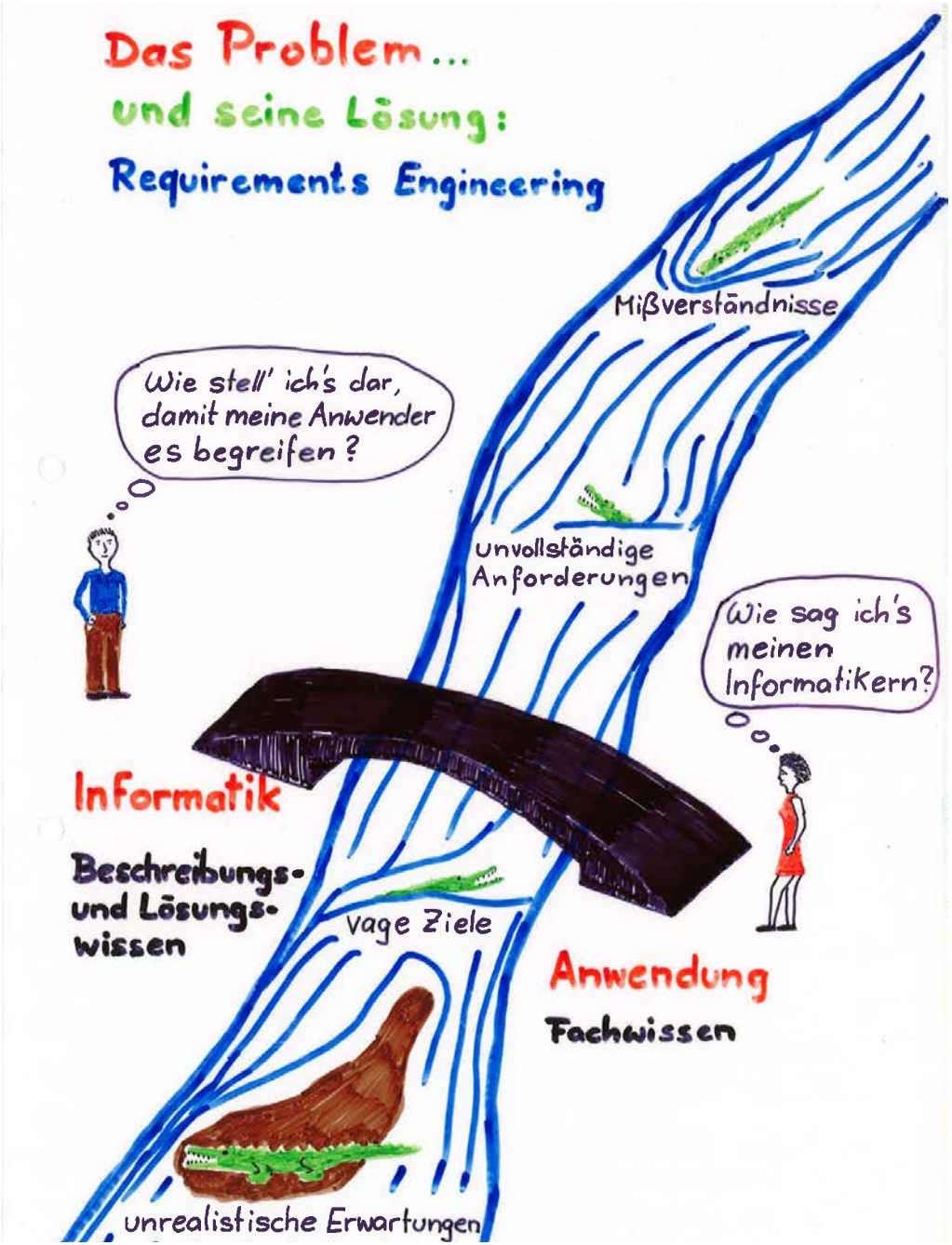
Abschiedsvorlesung
Farewell Lecture
2017-05-09

University of Zurich[UZH]

Department of Informatics

Requirements Engineering Research Group

First slide from my inaugural lecture 1994-06-27



Two fundamental invariants of Requirements Engineering (RE)

- Bridging the gap
- Creating and ensuring shared understanding

... between stakeholders and developers

My life with RE – in five stories

1 The early days

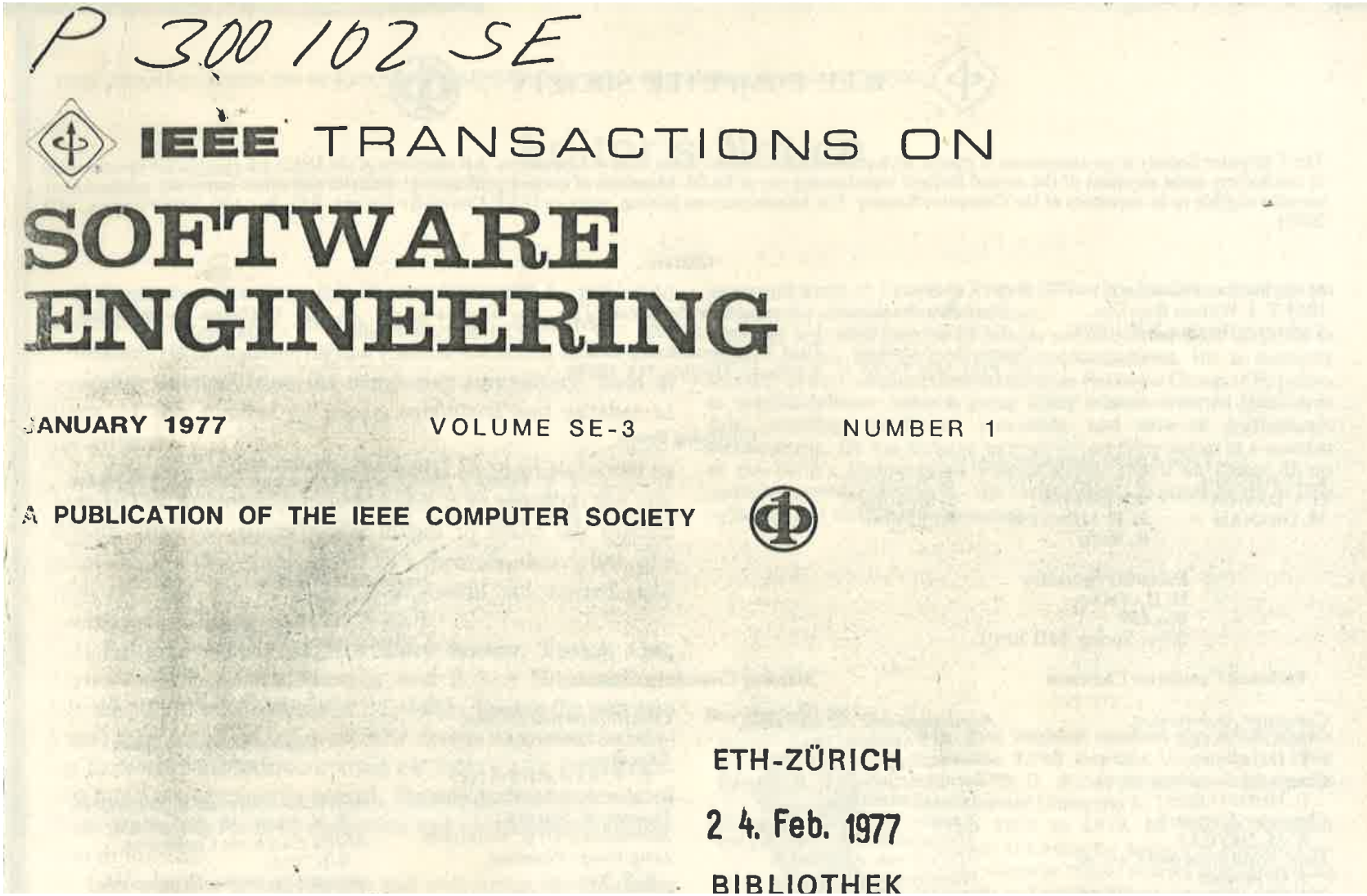
2 Requirements modeling

3 Stakeholders

4 Value-orientation

5 Shared understanding

The birth of a new discipline



P 300 102 SE

IEEE TRANSACTIONS ON

SOFTWARE ENGINEERING

JANUARY 1977 VOLUME SE-3 NUMBER 1

A PUBLICATION OF THE IEEE COMPUTER SOCIETY

ETH-ZÜRICH

2 4. Feb. 1977

BIBLIOTHEK

TSE SE-3(1) Special Collection on Requirement Analysis

Requirements Engineering in 1977

- Languages and tools (plus some methods)
- Requirements came (by some magic) from the "customer"
- Core parts of today's body of RE such as

  Elicitation, validation, and management of requirements, including negotiation, change management or traceability)

  not yet recognized
  - neither as practical problems
  - nor as research topics

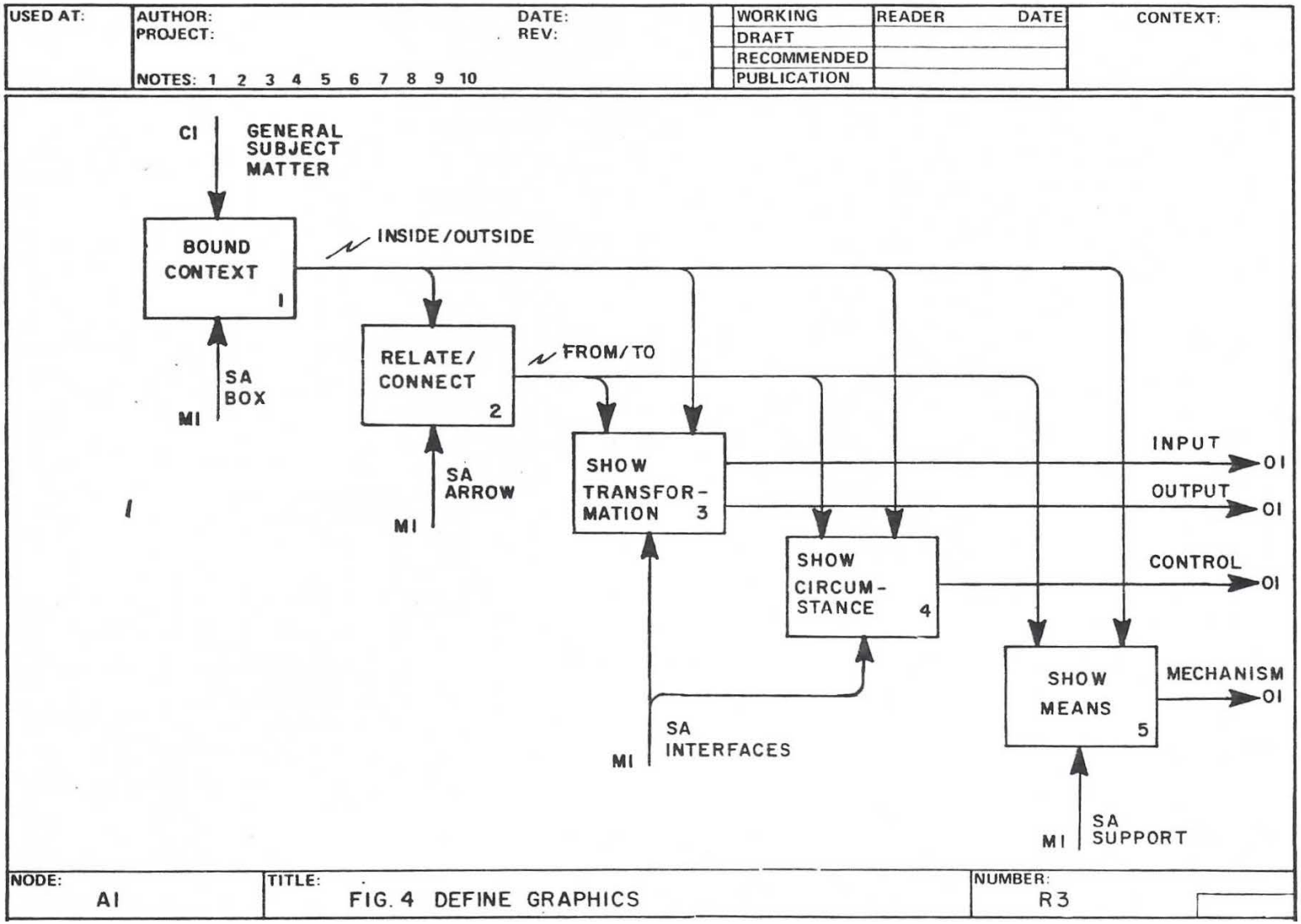
In January 1977 I was finishing my diploma thesis in Mathematics at RWTH Aachen University and had no idea that this work would shape my life some day

The vision of RE 1970' & 1980'

- Create complete and unambiguous specifications of the problem
- using formal or semi-formal models
- supported by powerful tools
- Textual (entity-relationship-based)
- Graphic (flow-based or state-event based)

Some examples: SADT

[Ross 1977]



Some examples: Semi-formal text

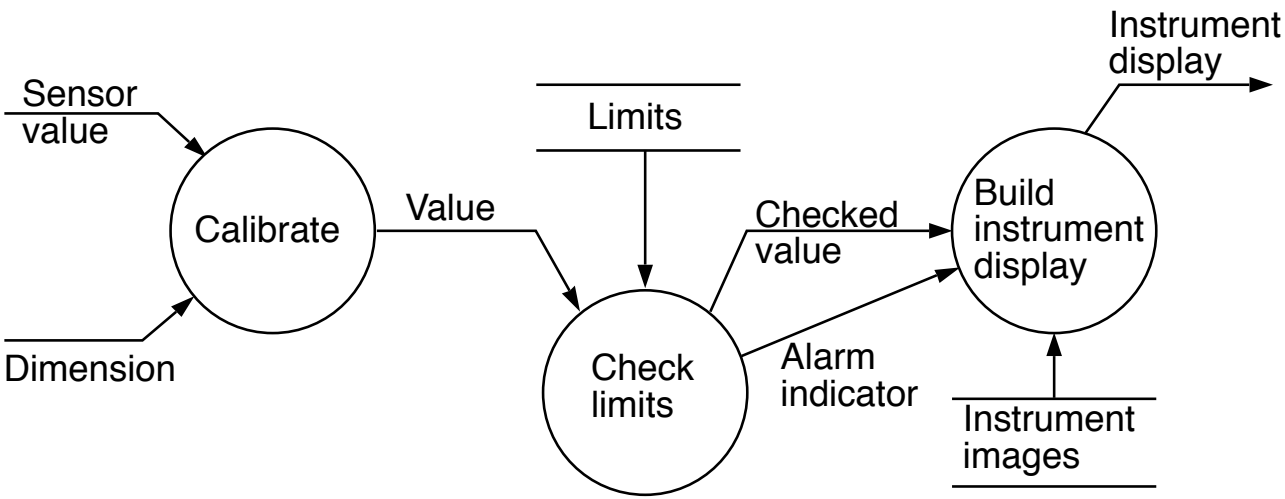
Structured textual requirements specification

Based on an entity-relationship model

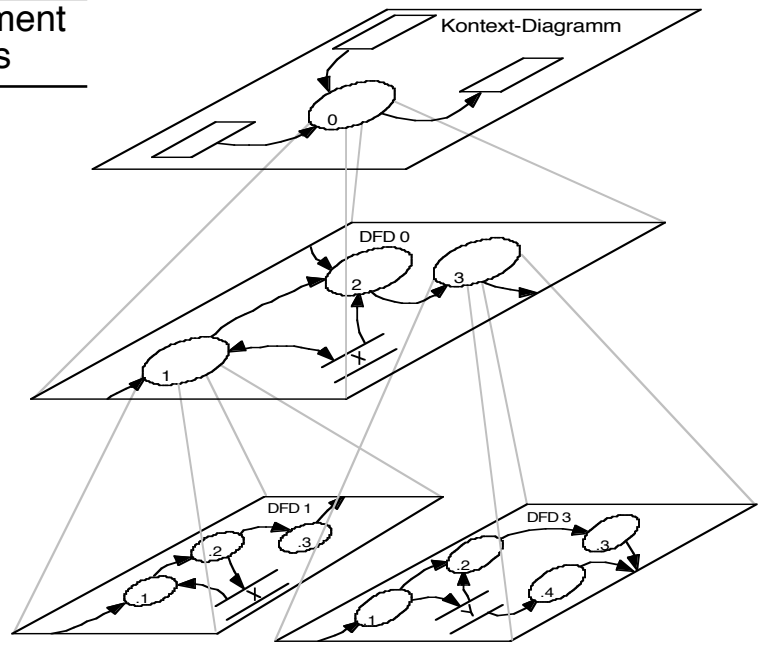
```
Parameters:   DB=-EXBDB  NAME=hourly-employee-processing  NOINDEX  NOPUNCHED-NAMES  PRINT  EMPTY
  NOPUNCH  SMARG=5  NMARG=20  AMARG=10  BMARG=25  RNMARG=70  CMARG=1  HMARG=60  NODESIGNATE
  SEVERAL-PER-LINE  DEFINE  COMMENT  NONEW-PAGE  NONEW-LINE  NOALL-STATEMENTS
  COMPLEMENTARY-STATEMENTS  LINE-NUMBERS  PRINTEOF  DLC-COMMENT

  1 PROCESS                                                hourly-employee-processing;
  2      /*   DATE OF LAST CHANGE - JUN 26, 1976, 13:56:44 */
  3     DESCRIPTION;
  4          this process performs those actions needed to interpret
  5          time cards to produce a pay statement for each hourly
  6          employee.;
  7     KEYWORDS:      independent;
  8     ATTRIBUTES ARE:
  9          complexity-level
 10                        high;
 11     GENERATES:     pay-statement, error-listing,
 12                    hourly-employee-report;
 13     RECEIVES:      time-card;
 14     SUBPARTS ARE:  hourly-paycheck-validation, hourly-emp-update,
 15                    h-report-entry-generation,
 16                    hourly-paycheck-production;
 17     PART OF:       payroll-processing;
 18     DERIVES:       pay-statement
 19       USING:       time-card, hourly-employee-record;
 20     DERIVES:       hourly-employee-report
 21       USING:       time-card, hourly-employee-record;
 22     DERIVES:       error-listing
 23       USING:       time-card, hourly-employee-record;
 24     PROCEDURE;
 25          1. compute gross pay from time card data.
```

PSL [Teichroew&Hershey 1977]
Espreso [Ludewig 1983]
SPADES [Ludewig et al. 1985]

Some examples: Structured Analysis



[DeMarco 1979]



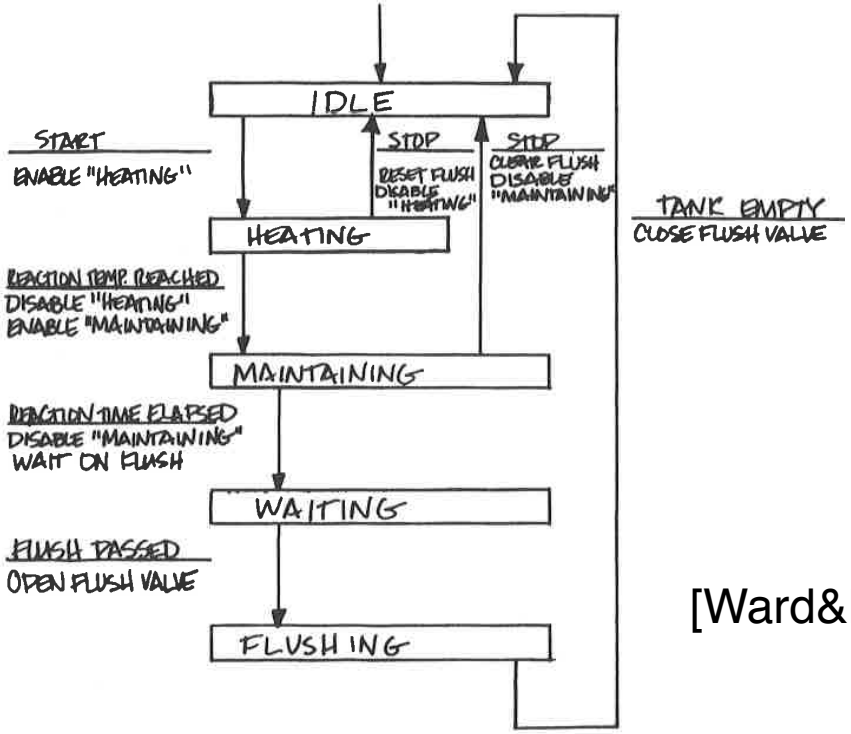
Based on a hierarchy of
dataflow diagrams

Later extended with capabilities
to model state-dependent behavior

Some examples: STDs, statecharts

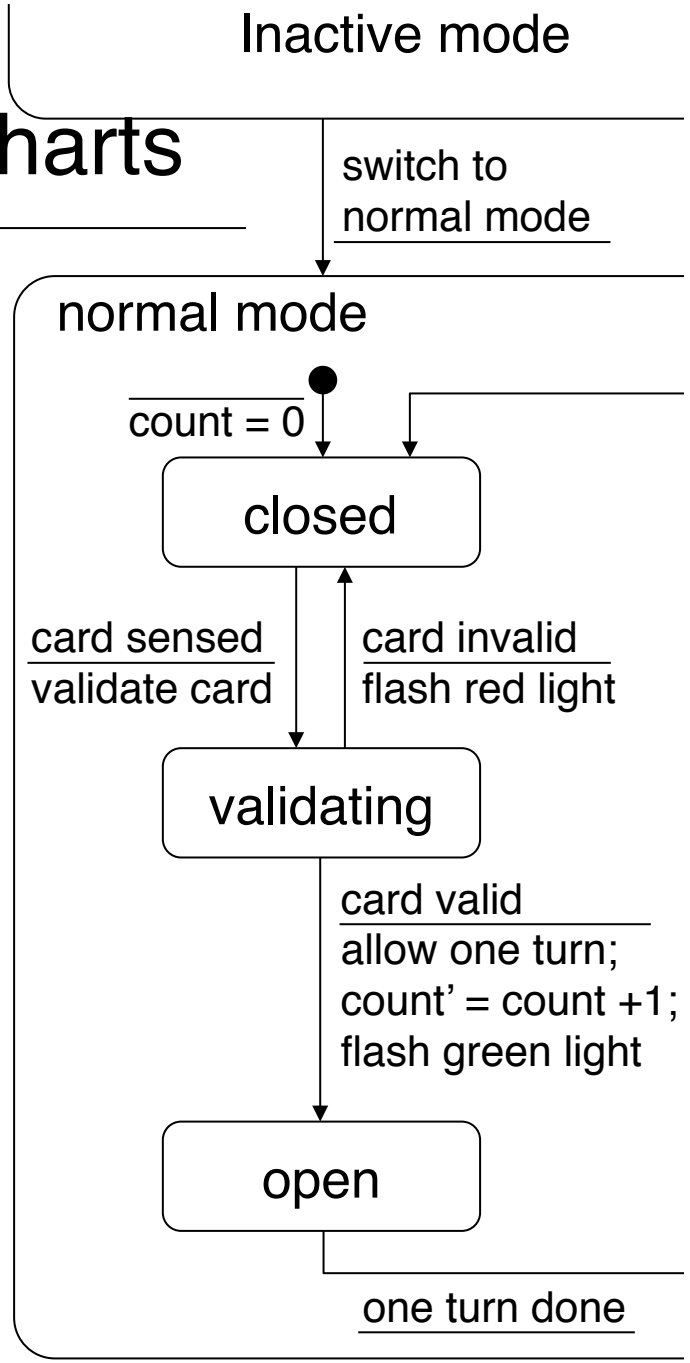
Specification of event-driven, state-dependent system behavior

State-transition diagrams, statecharts



[Ward&Mellor 1985]

[Harel 1987]



1983 – The beginning of a personal liaison

In March 1983, I joined Brown Boveri (BBC) Research as a postdoc

By then I considered myself as a database person

Our goal at BBC research was to build an integrated software engineering environment – everything from requirements to source code

Requirements were to be stated in a text-based, semi-formal language: ESPRESO / SPADES

My task was to design the database

This was the plan

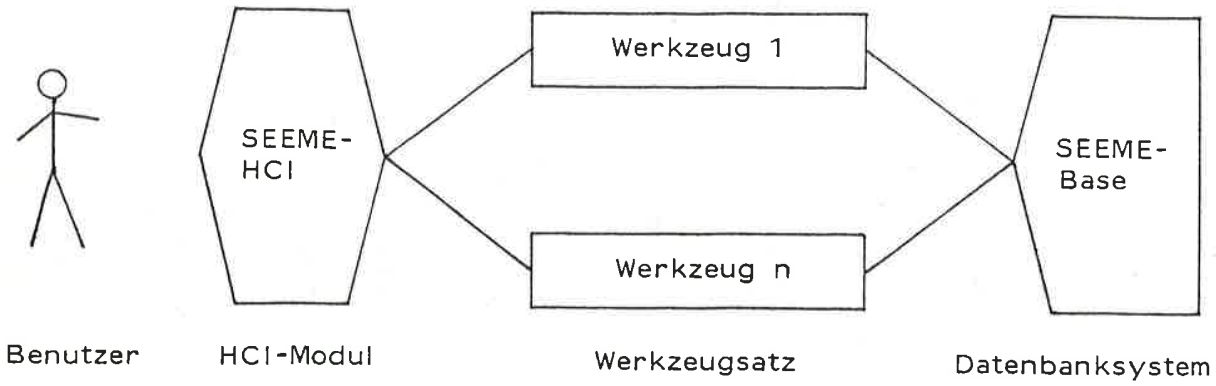


Bild 1.1: Grundstruktur der Software-Entwicklungsumgebung SEEME

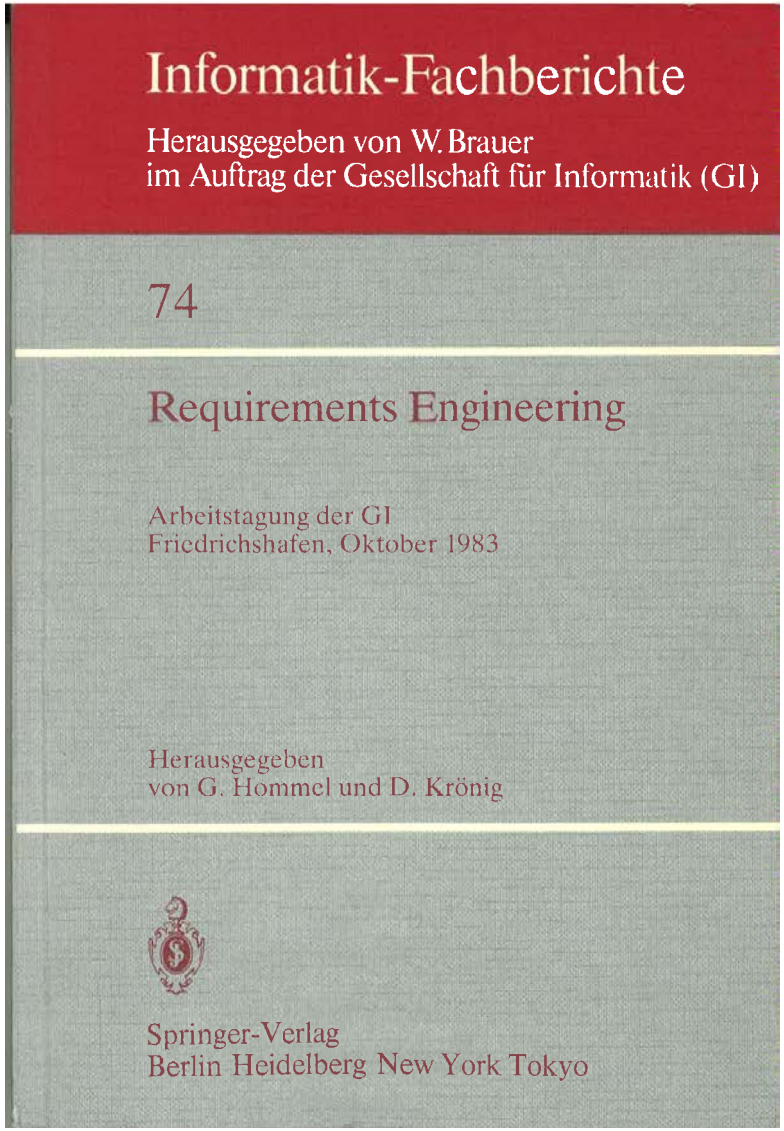
[Glinz&Ludewig 1984]

And we believed in it...

Today we know that we were doomed to fail.

However, there was an effect: RE took possession of my life

The very first RE conference – in fall 1983



So I came across RE

and found it exciting

which I still do today.

[Hommel&Krönig 1983]

My life with RE – in five stories

1 The early days

2 Requirements modeling

3 Stakeholders

4 Value-orientation

5 Shared understanding

My journey with requirements modeling

From 1984 until today

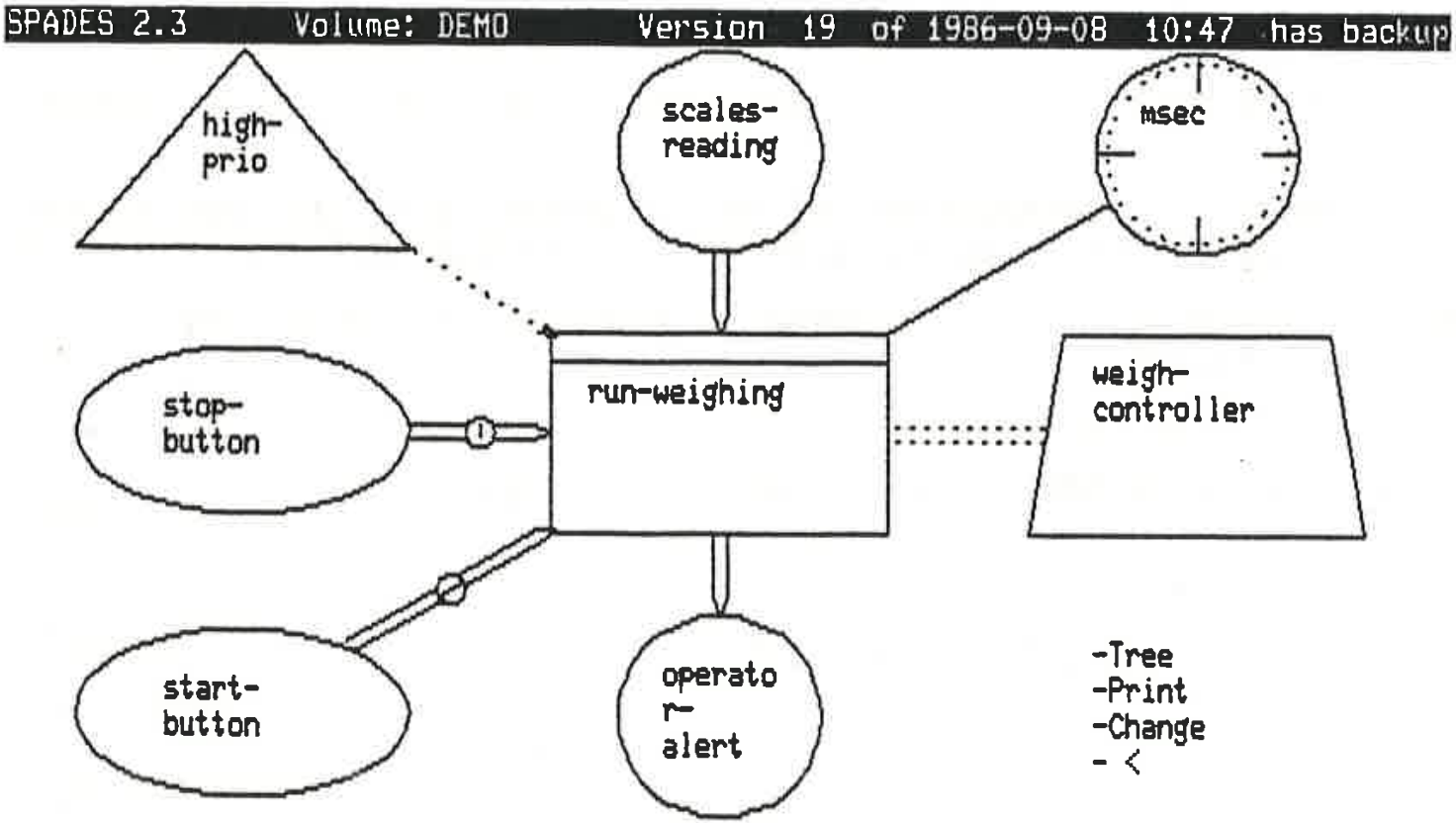
1984-1987: SPADES

[Ludewig et al. 1985]
[Glinz&Ludewig 1986]
[BBC 1986a, 1986b]

```
*****************************************************************************
*  SPADES 2.3   SPADES-L Form - REPORT       Generated: 1986-10-21  10:51     *
*                                                                             *
*  Volume:  DEMO                  Based on version:   2  1986-10-21  06:21   *
*****************************************************************************


(* ************************    'MODULE'-DIVISION   ********************** *)

 'MODULE' weigh-automation:
 "Automates a weighing process for liquids ";
 'COMPRISES'
     'MODULE' weighing-system:
     "Control and monitoring of weighing process for liquids ";
     'TEXT' Function "
     This is a demo specification of a system that monitors weighing of
     liquids. Its main components are two subsystems: !-
 - the initialization subsystem !weigh-init
 - the run time system !weigh-rts. ";
     'COMPRISES'
        'BUFFER' scales-reading
     'AND'
        'MODULE' weigh-init
     'AND'
        'MODULE' weigh-rts:
        "Weighing run time system ";
        'COMPRISES'
           'PROCEDURE' run-weighing:
           "Runs the weighing process ";
```

- Textual language
- Based on a set of fixed types for entities and relationships

1984-1987: SPADES



- Tool supported incremental specification and rich textual reporting
- Very limited capabilities for graphic output

1986-1991: SA/RT

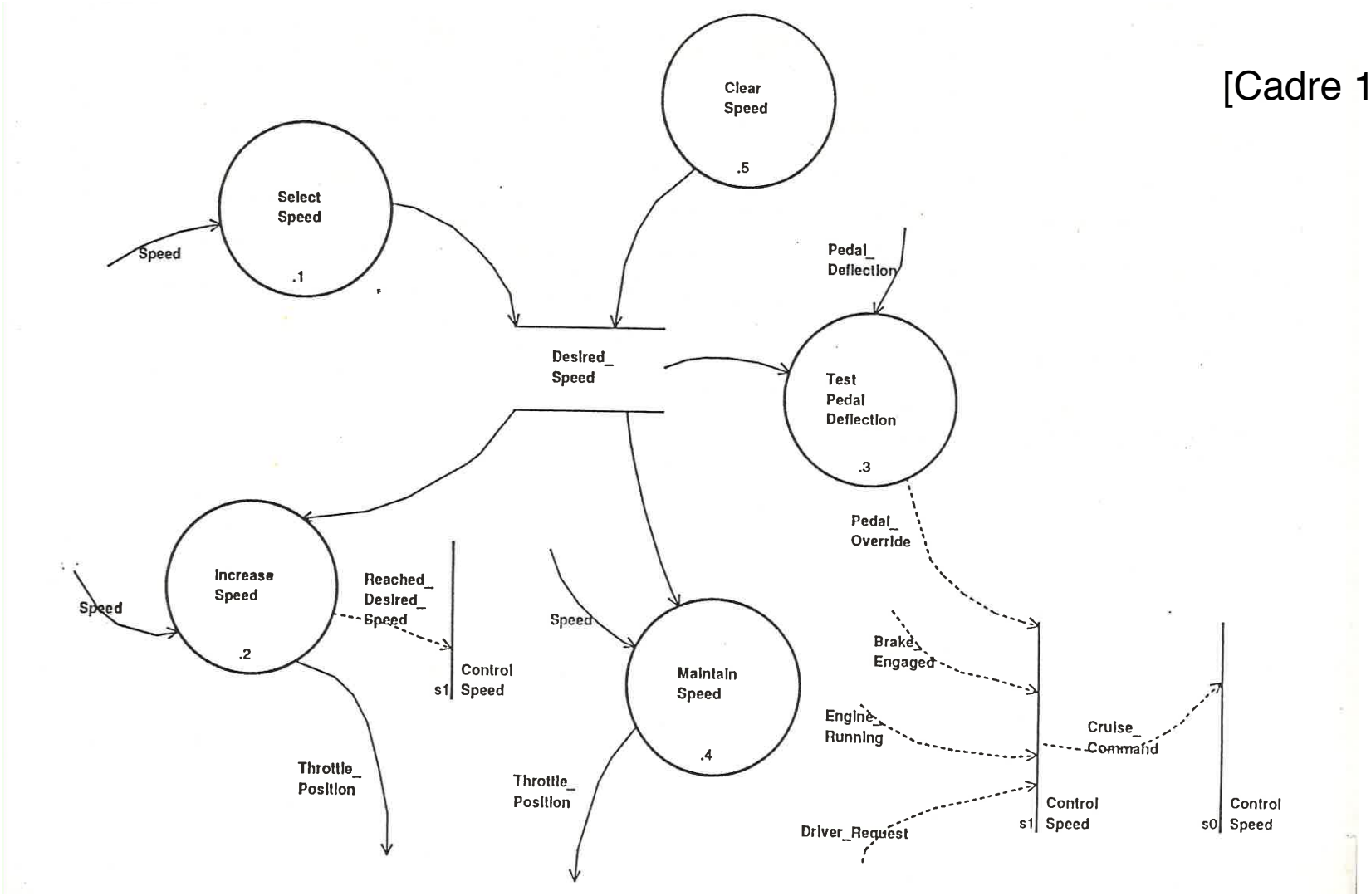
- In 1986, the first practically usable tools for graphic modeling of dataflow diagrams appeared on the market
- Tools for graphic modeling eventually killed SPADES
- Learned and taught SA/RT: Structured Analysis with real time extensions [Ward&Mellor 1985] [Hatley&Pirbhai 1987]
- Teaching it taught me how to make it practically applicable

A crystallization point:
Tool exhibition at ICSE ’87
in Monterey

- Software through Pictures by IDE
- Teamwork by Cadre
- Statemate by AD CAD / David Harel
  and many more ...

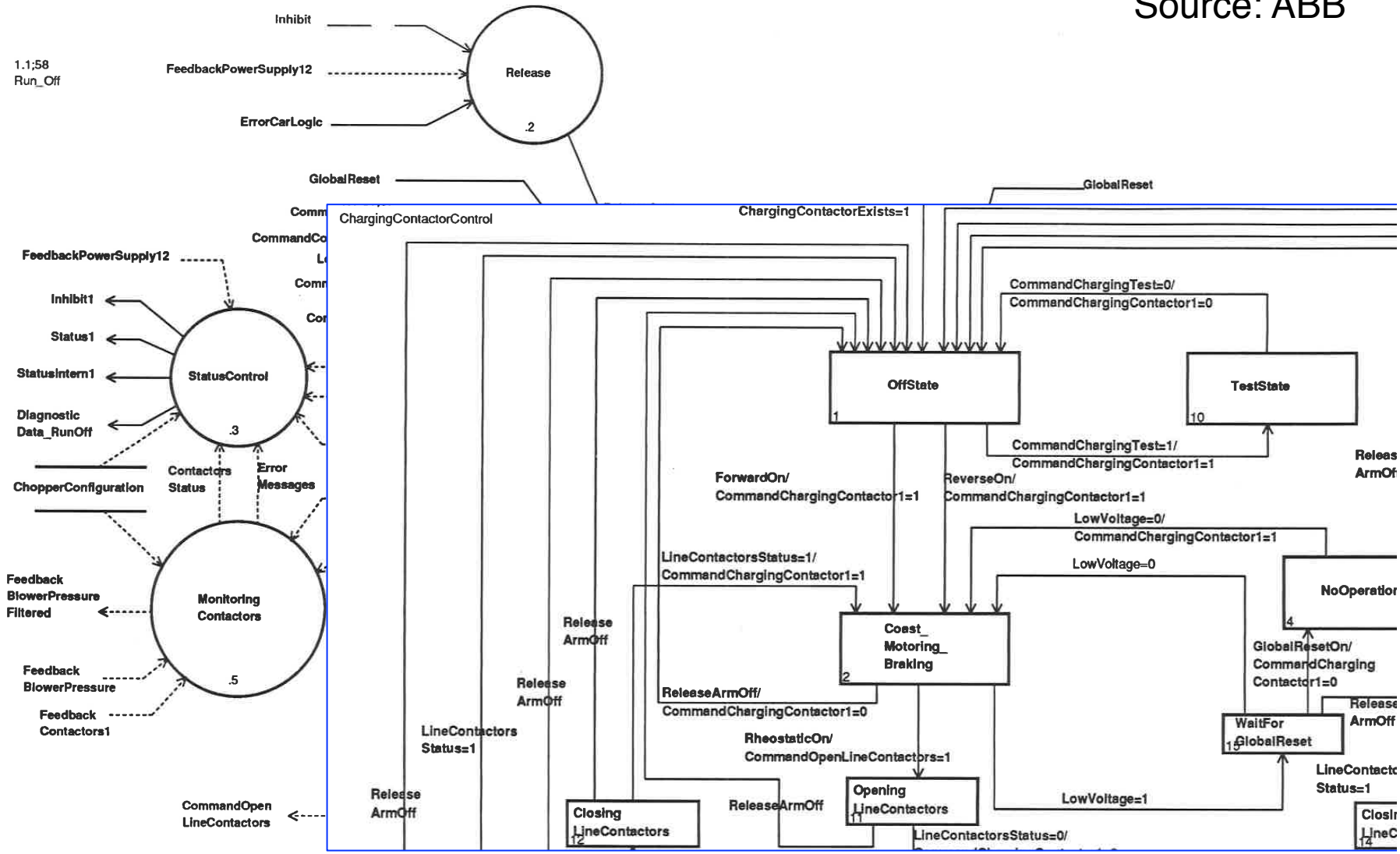
SA/RT: A textbook example

[Cadre 1986]



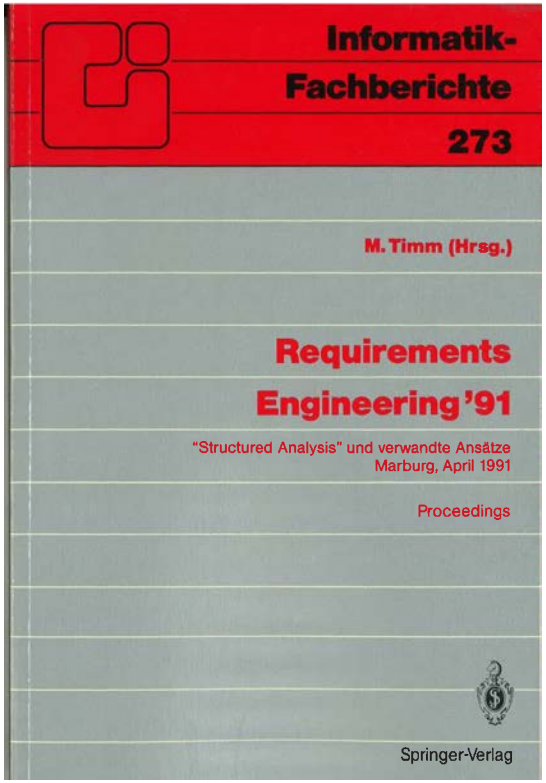
SA/RT: ... and a real-world example

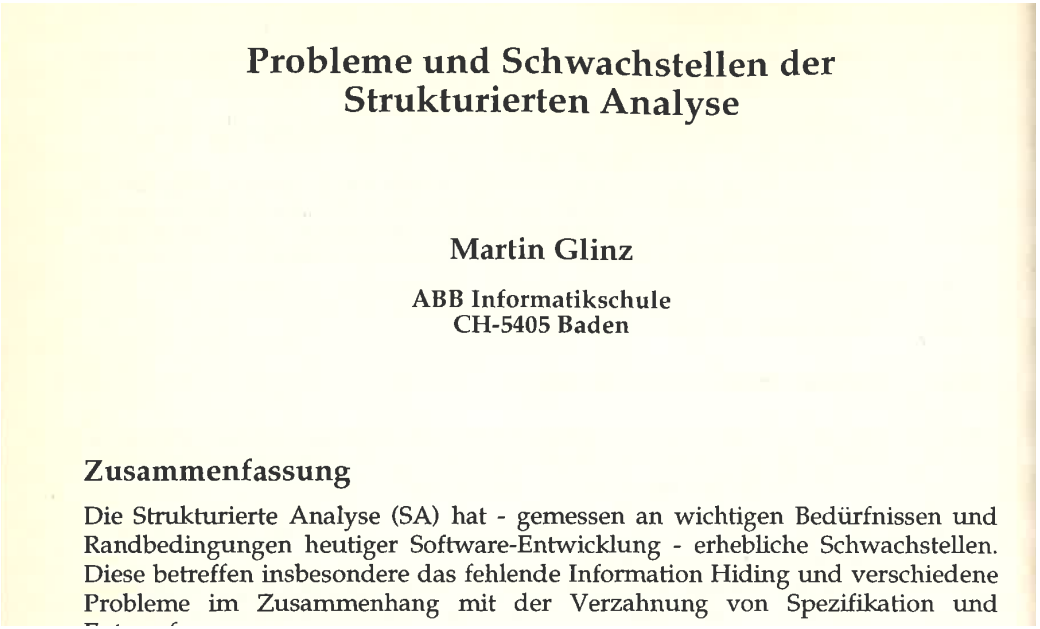
Source: ABB



1991: The rise of object-oriented modeling

- Inherent problems and weaknesses of SA/RT
- Started investigating object-oriented modeling





Probleme und Schwachstellen der Strukturierten Analyse

Martin Glinz

ABB Informatikschule
CH-5405 Baden

Zusammenfassung

Die Strukturierte Analyse (SA) hat - gemessen an wichtigen Bedürfnissen und Randbedingungen heutiger Software-Entwicklung - erhebliche Schwachstellen. Diese betreffen insbesondere das fehlende Information Hiding und verschiedene Probleme im Zusammenhang mit der Verzahnung von Spezifikation und

[Glinz 1991]

1991-2011: Object-oriented modeling of requirements

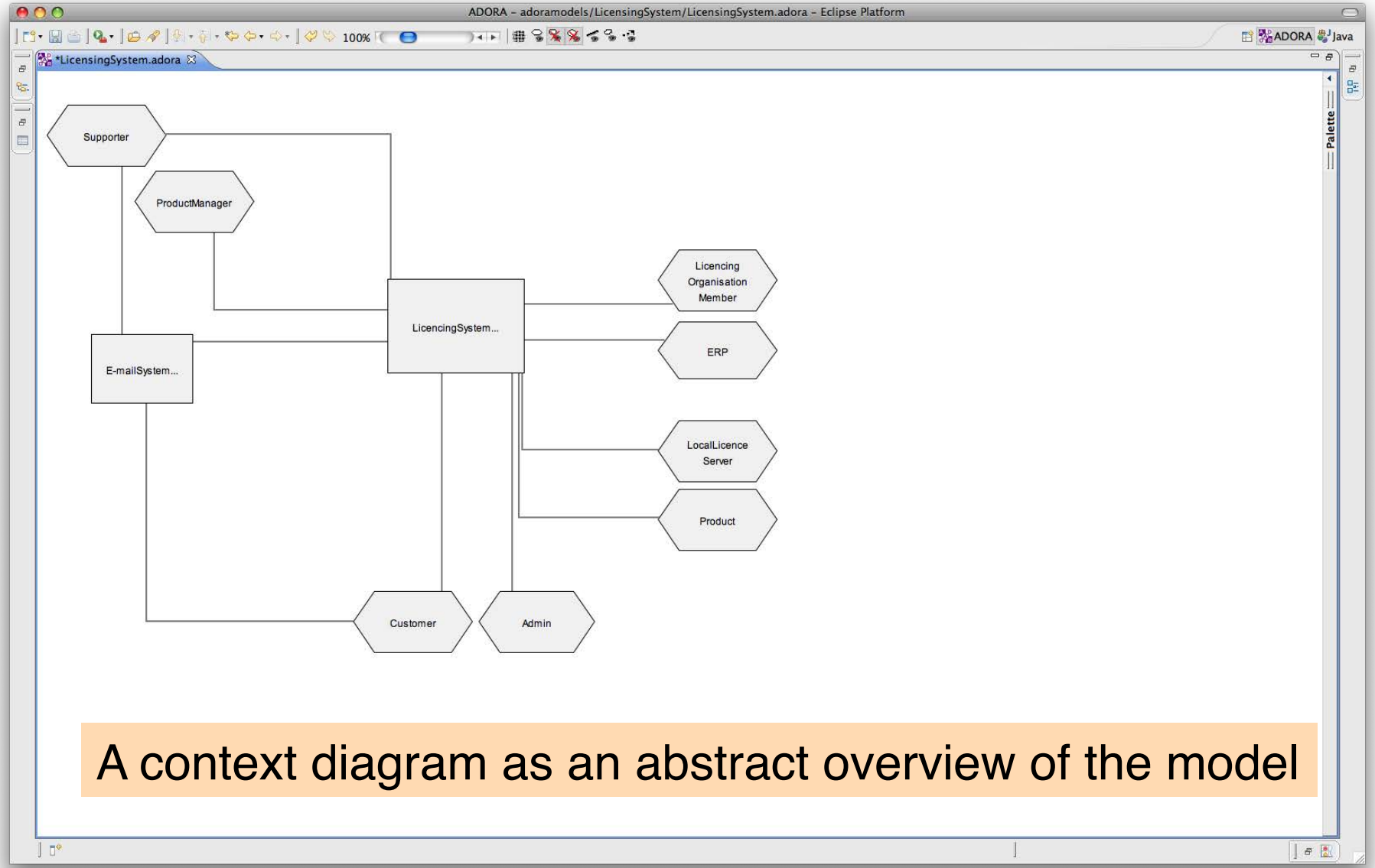
- Hierarchically decomposed object models [GI-RE'93]
- based on scenarios and statecharts [ESEC 1995]
- ADORA [ASE 1998, Inf. Syst. 2002, ASE 2006, ICSE 2008, ...]
- Our focus activity for many years

ADORA

- A language and tool for object-oriented specification of software-intensive systems
- Basic concepts
  - Modeling with abstract objects
  - Hierarchic decomposition of models
  - Integration of object, behavior and interaction modeling
  - Model visualization in context with generated views
  - Adaptable degree of formality

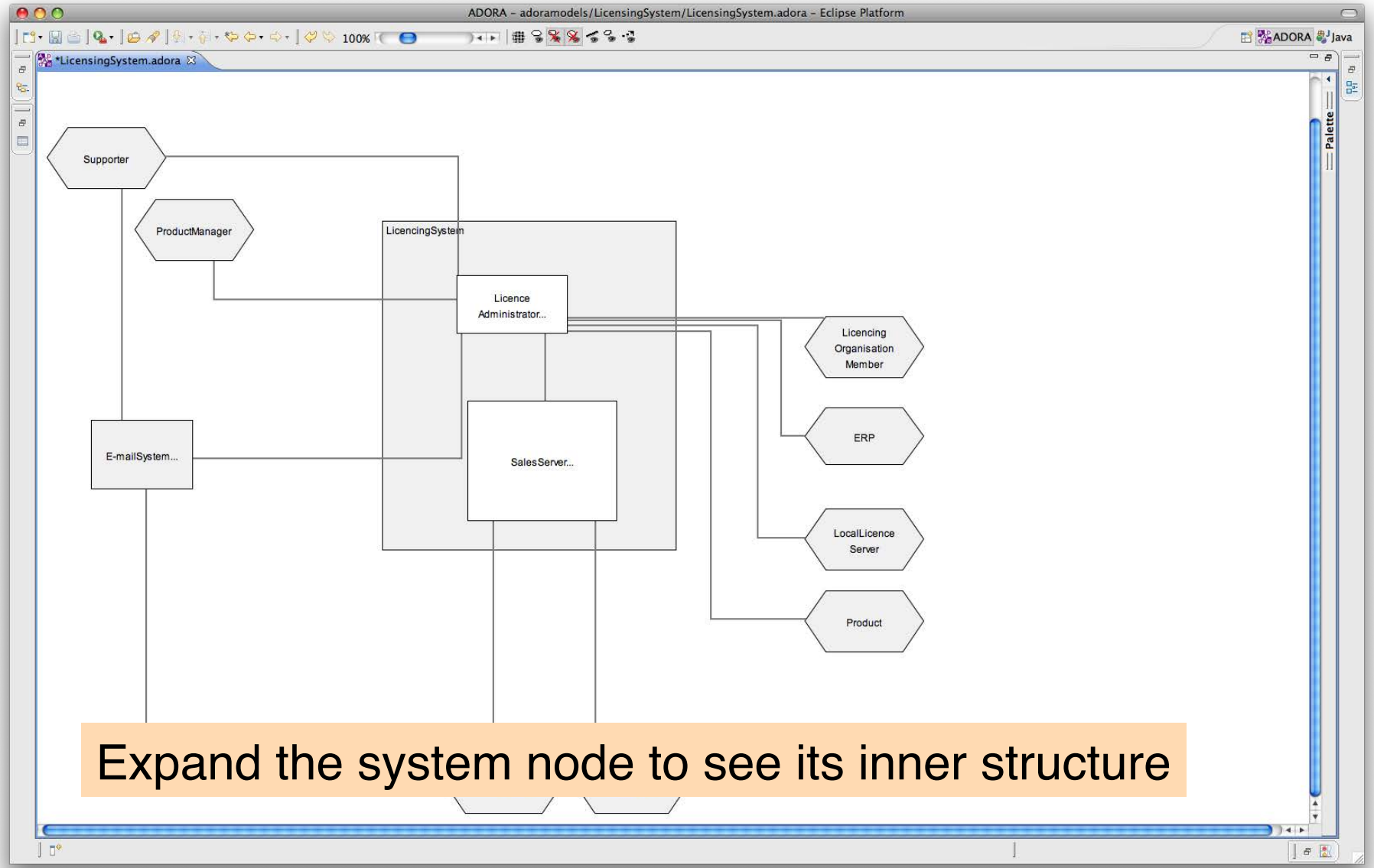
[Berner et al. 1998]
[Glinz et al. 2002]
[Reinhard et al. 2006]
[Reinhard et al. 2008]
and many more

The ADORA Tool – A Short Demo



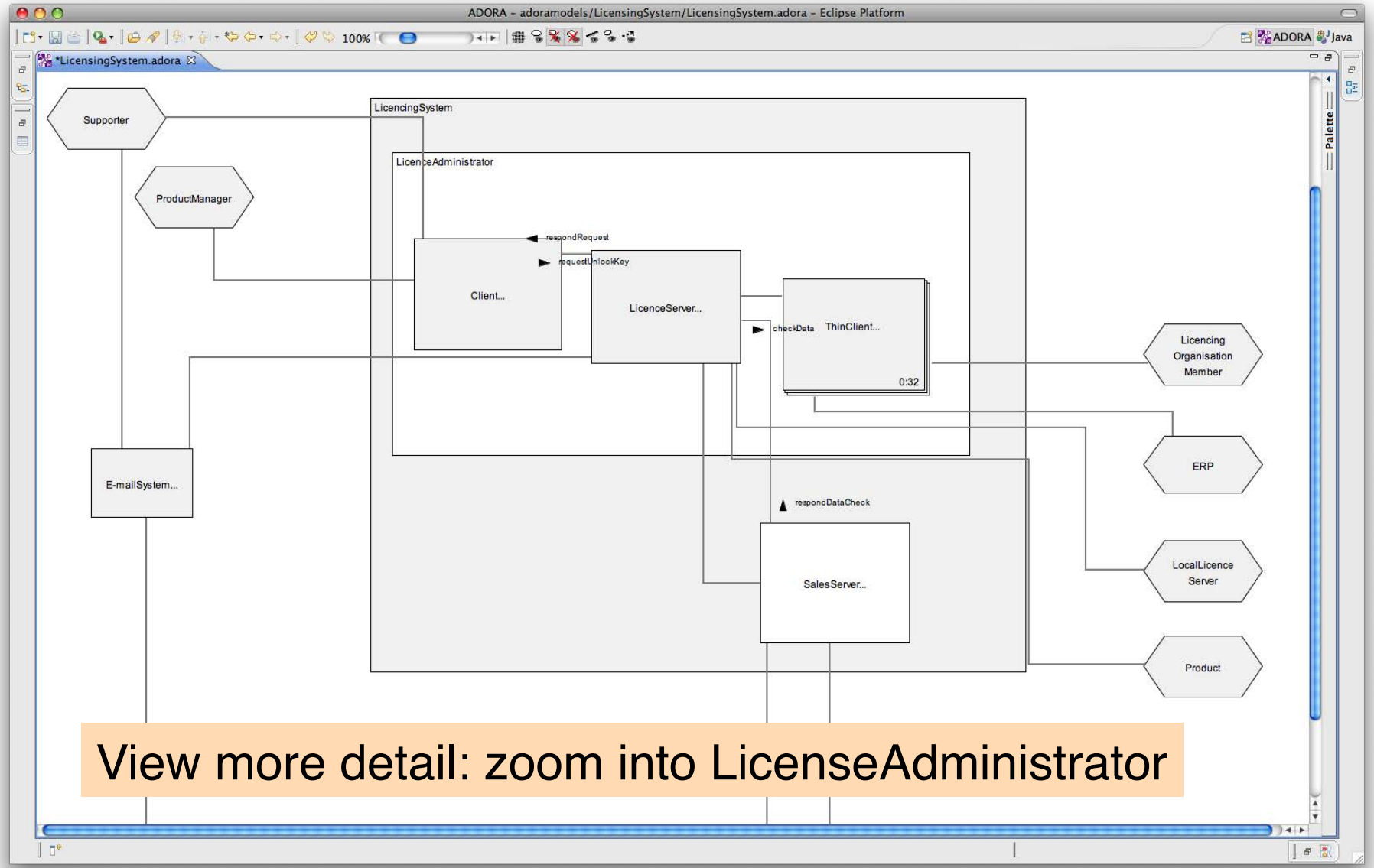
A context diagram as an abstract overview of the model

The ADORA Tool – A Short Demo



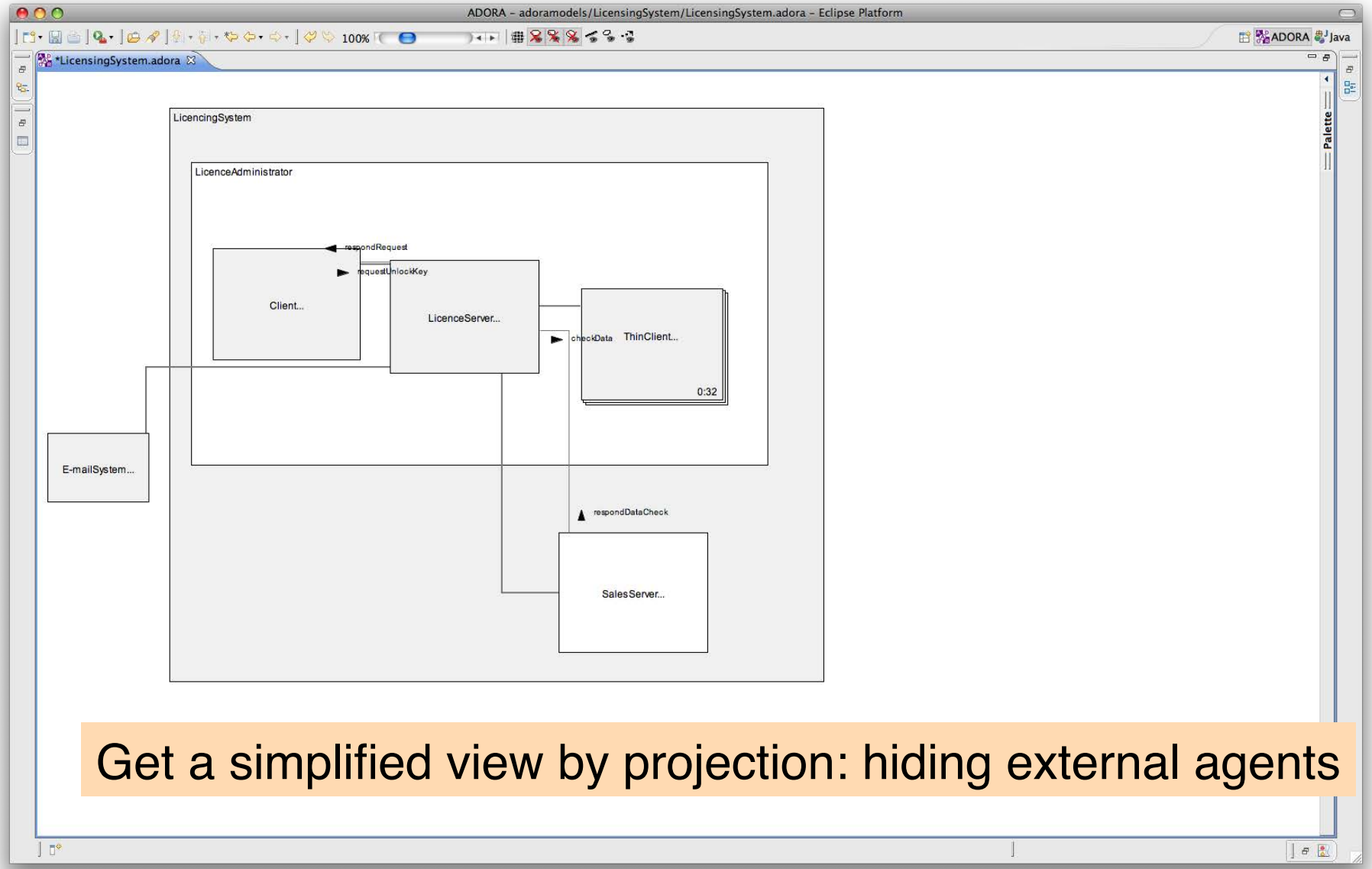
Expand the system node to see its inner structure

The ADORA Tool – A Short Demo



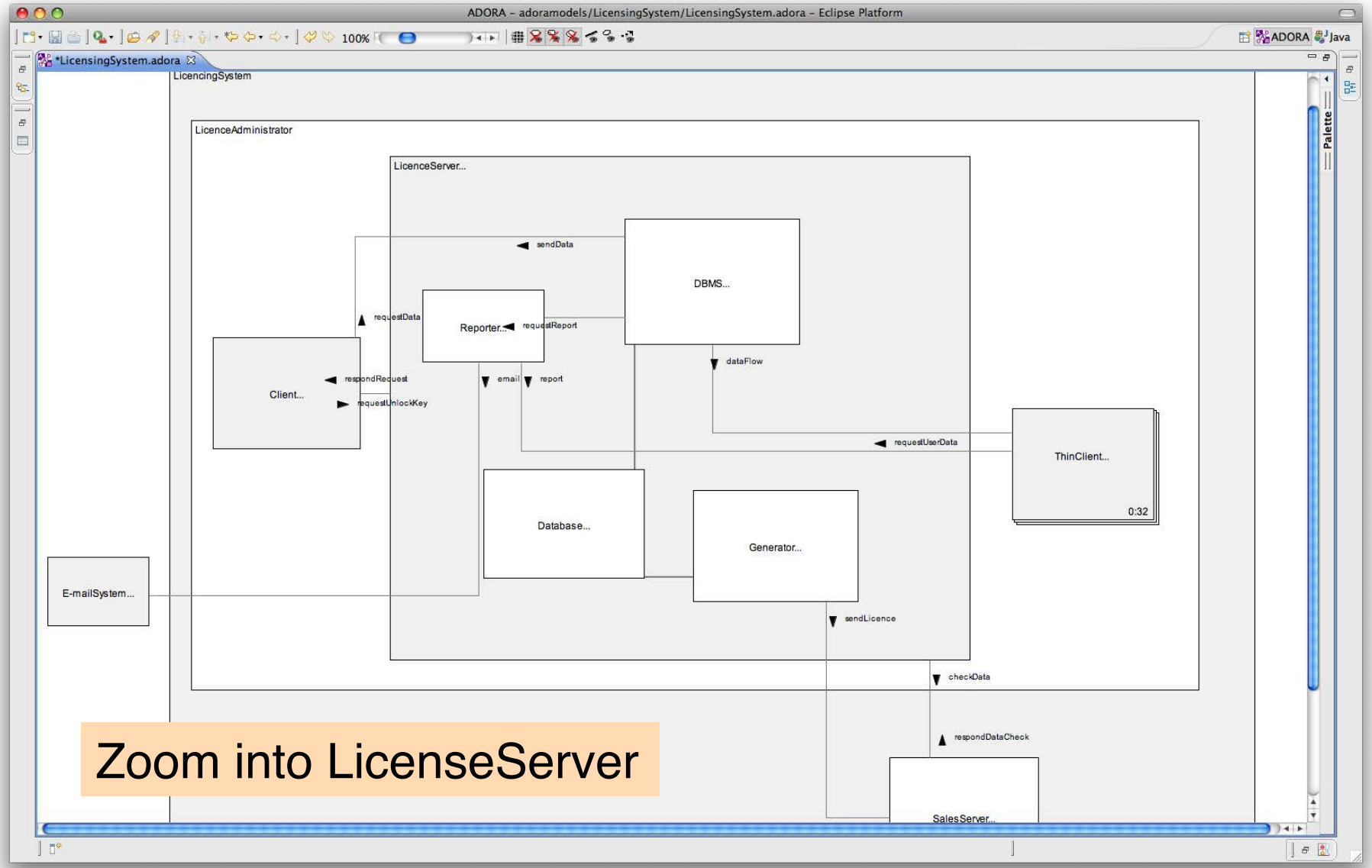
View more detail: zoom into LicenseAdministrator

The ADORA Tool – A Short Demo



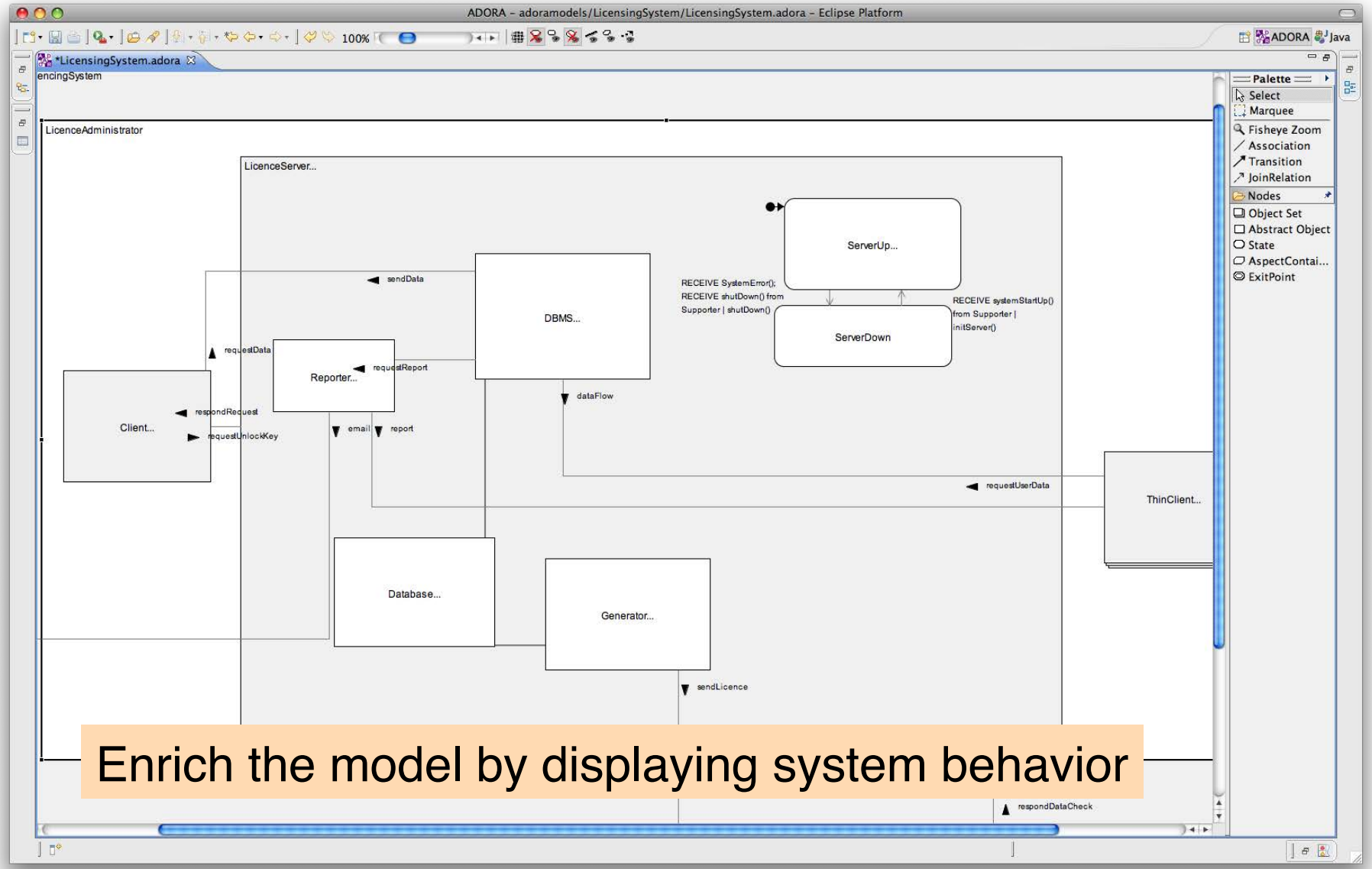
Get a simplified view by projection: hiding external agents

The ADORA Tool – A Short Demo



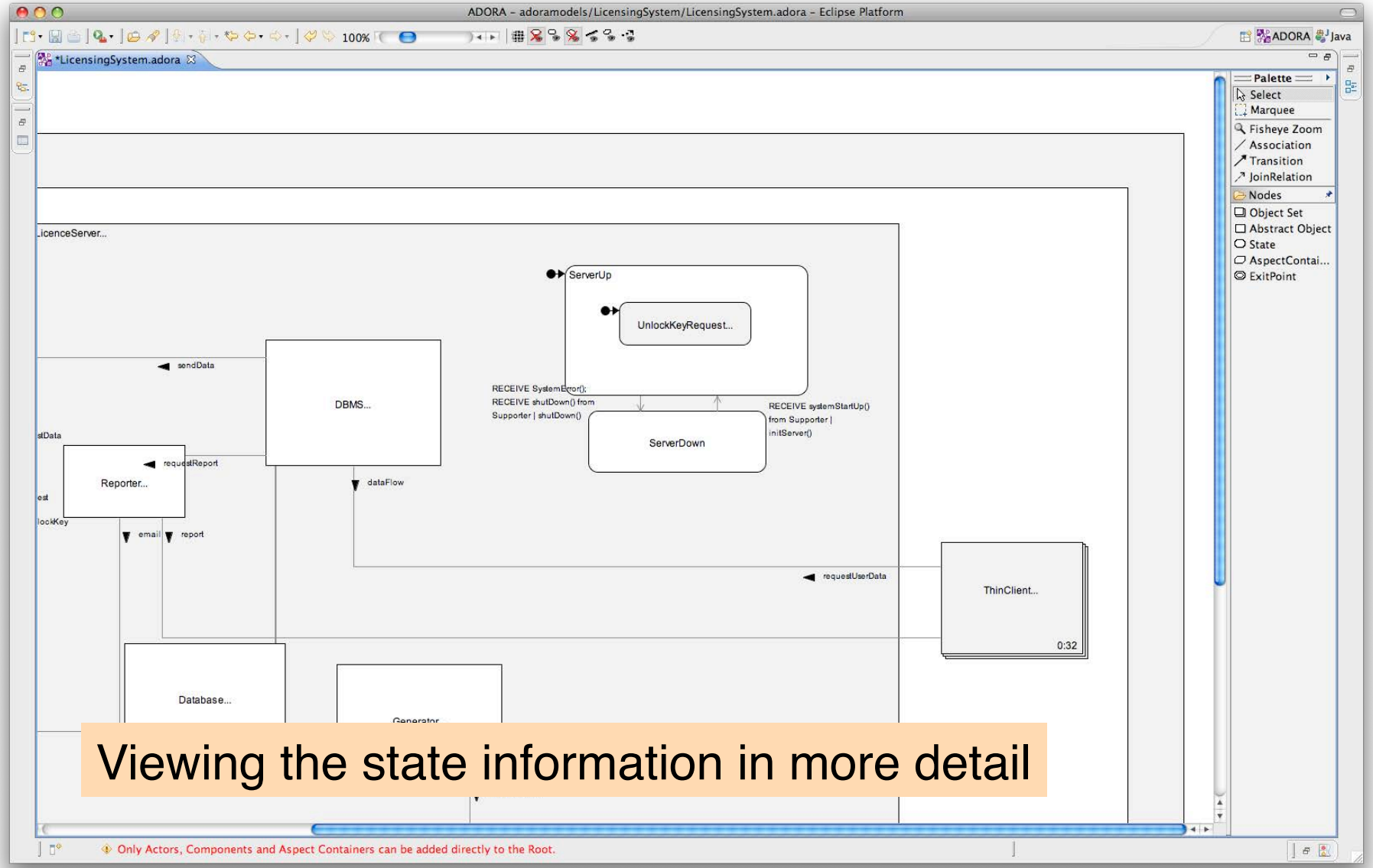
Zoom into LicenseServer

The ADORA Tool – A Short Demo



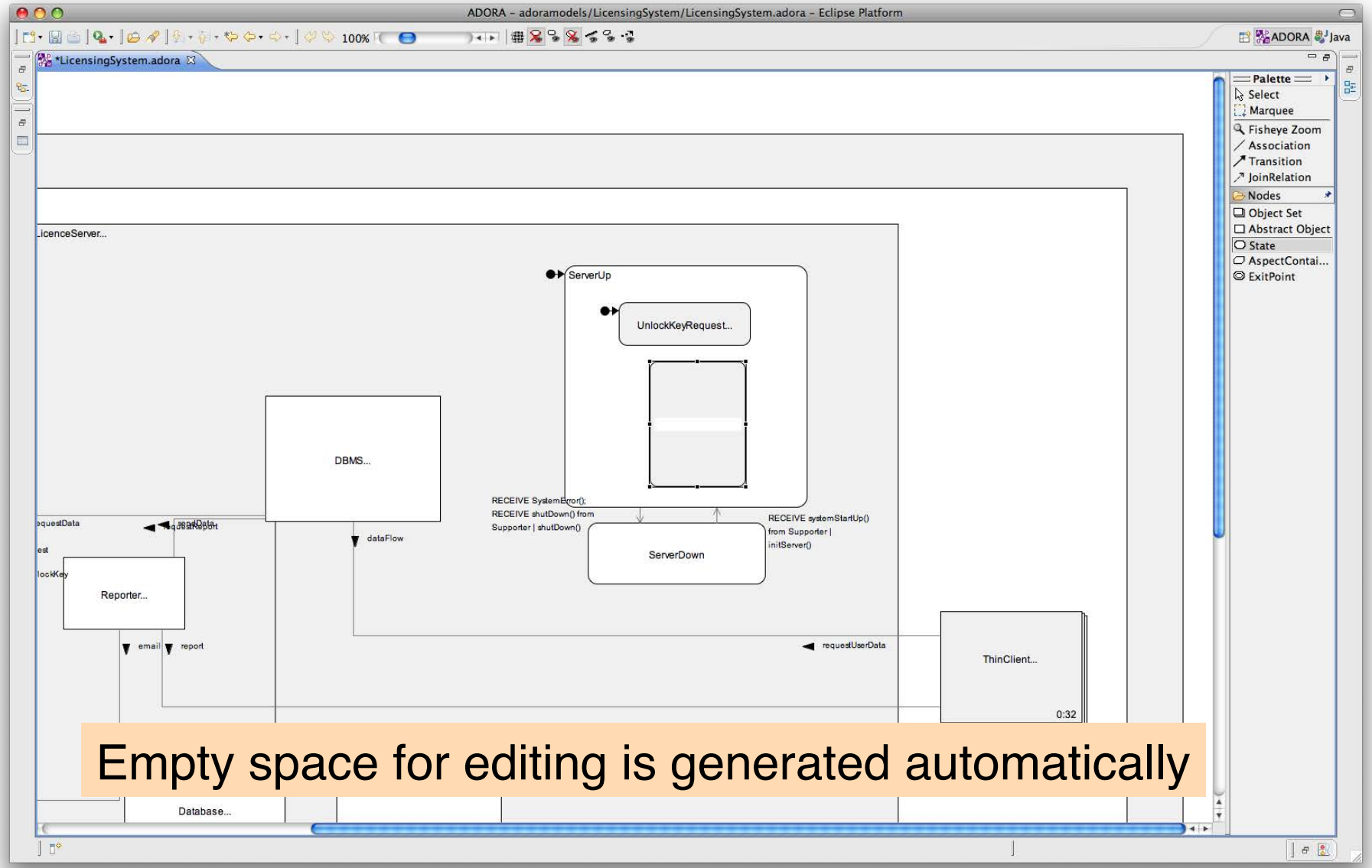
Enrich the model by displaying system behavior

The ADORA Tool – A Short Demo



Viewing the state information in more detail

The ADORA Tool – A Short Demo



Empty space for editing is generated automatically

Eventually, ADORA did not make it

due to

- Our inability to develop the tool prototype into a product
- The dominance of UML

Despite the weaknesses of UML as a requirements language

[Glinz 2000]

Problems and Deficiencies of UML as a Requirements Specification Language

Martin Glinz
Institut für Informatik, Universität Zürich
Winterthurerstrasse 190
CH-8057 Zurich, Switzerland
glinz@ifi.unizh.ch

Abstract

In recent years, UML has become a standard language for modeling software requirements and design

requirements specification problems. We try to model these issues with UML and discuss the difficulties encountered.

2009 – today: Flexible modeling

A somehow disturbing observation:

Plain old text + Rich Pictures won the Next Top Model Contest at RE'09

Hmmm.

Spawned our work on flexible, lightweight modeling

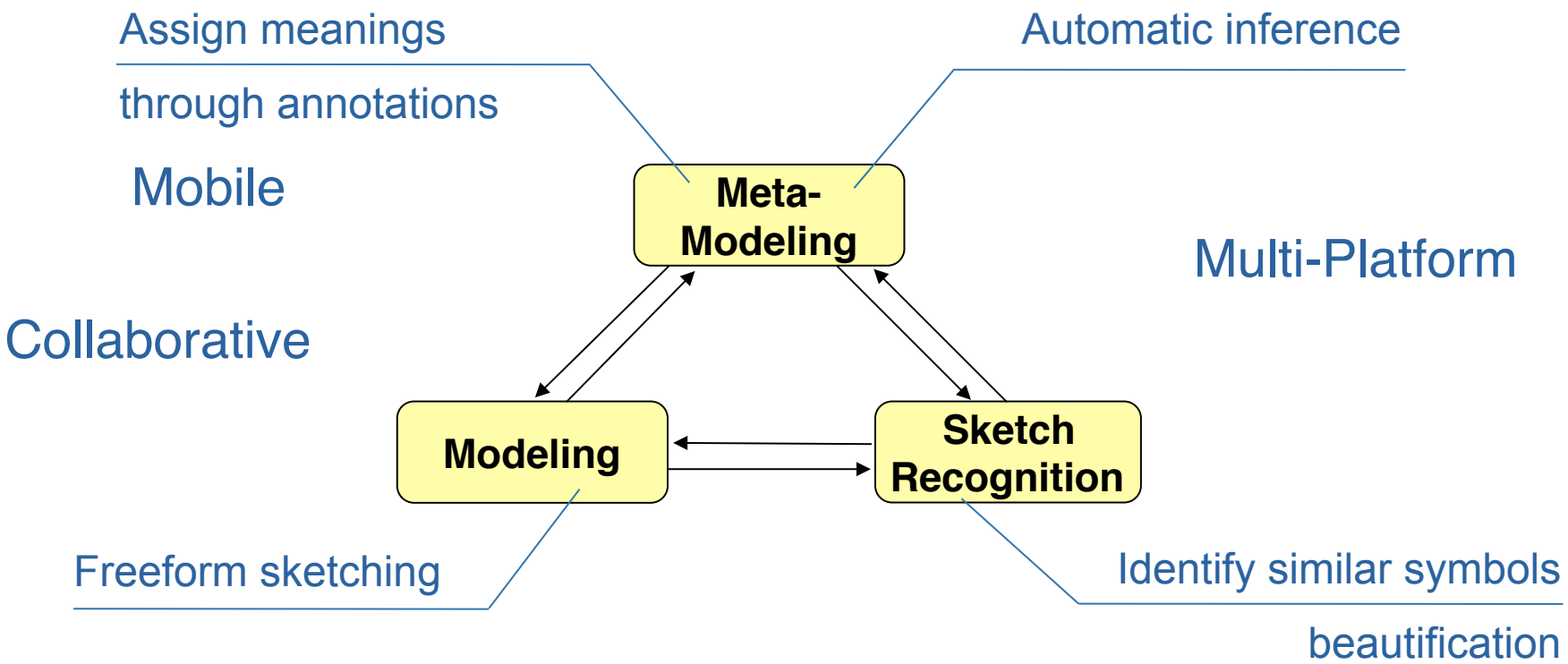
- A Very Lightweight Modeling Language [RE'10]
- FlexiSketch [REFSQ 2011, ASE 2013, RE'15, ICSE 2015]

[Glinz 2010], [Wüest&Glinz 2011]
[Wüest et al. 2013, 2015a, 2015b]

FlexiSketch

Liberating Modelers from the Tyranny of a Strict Modeling Language

- Support a flexible sketching / modeling process
- Allow users to define their own notations / languages on the fly
- Co-evolve models and metamodels

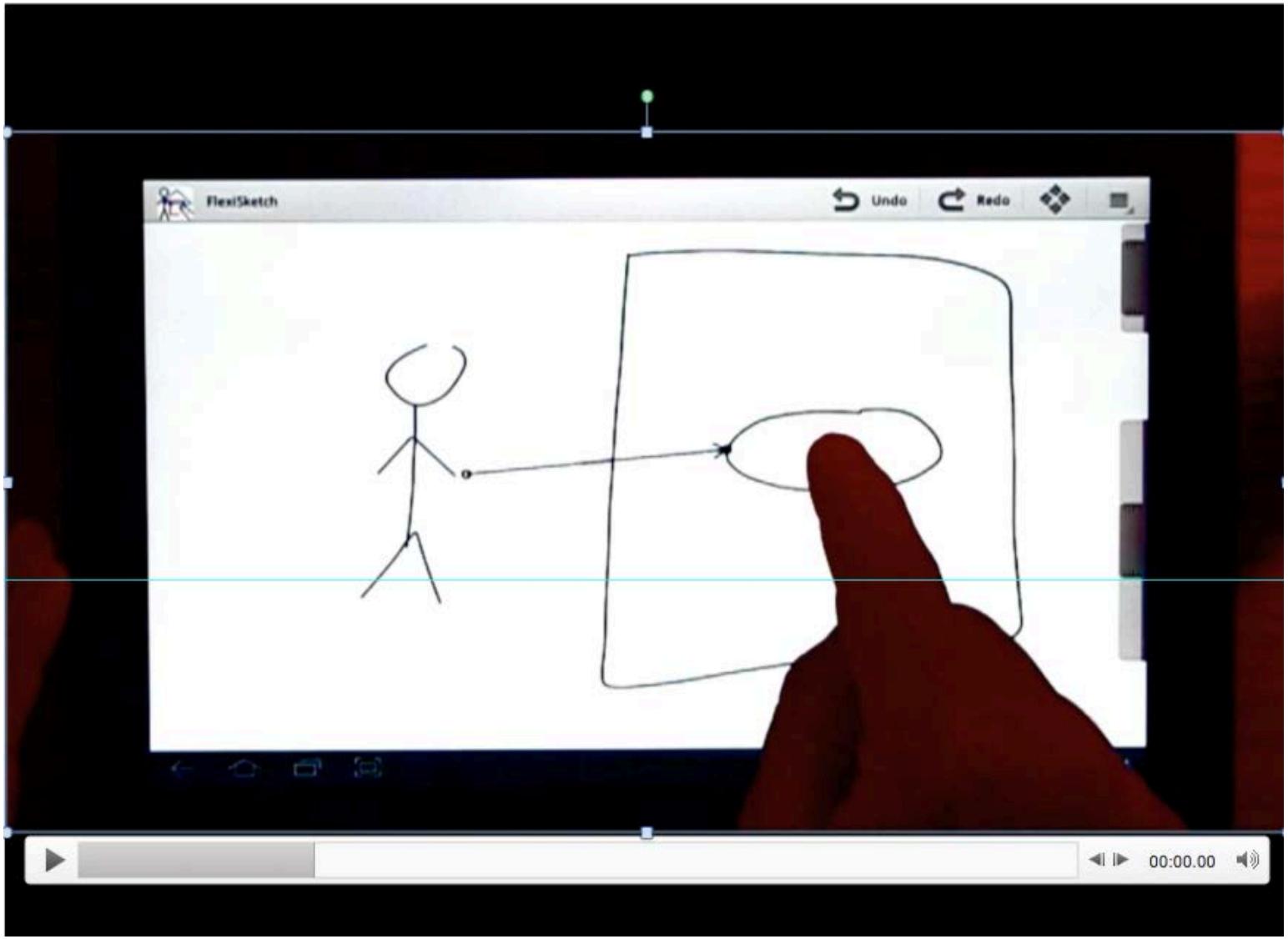


FlexiSketch in Action

FIGURE 3. Martin Glinz, Norbert Seyff, Dustin Wüest, and Parisa Ghazi work with FlexiSketch. For additional information, visit www.ifi.uzh.ch/rerg/research/flexiblemodeling.html.

IEEE Software
32(1), p. 20
Jan/Feb 2015

A FlexiSketch Teaser



My life with RE – in five stories

1 The early days

2 Requirements modeling

3 Stakeholders

4 Value-orientation

5 Shared understanding

A personal paradigm shift

DEFINITION [1993]. Requirements Engineering – The application of a systematic, disciplined, quantifiable approach to the specification and management of requirements; that is the application of engineering to requirements.

Adapted from the definition of software engineering in IEEE Std. 610.12 [IEEE 1990]

DEFINITION [1998]. Requirements Engineering – Understanding and documenting the customers' desires and needs.

I came up with this definition in 1998 when writing a column on Requirements Engineering for the newsletter of an IT company I had collaborated with.

With the notion of stakeholders becoming pervasive, this became "the stakeholders' desires and needs"

A personal milestone

14th IEEE International
Requirements Engineering
Conference

Minneapolis/St. Paul, Minnesota, USA
September 11-15, 2006

http://www.re06.org

Conference Organization

General Chair
Robyn Lutz, Iowa State University and Jet Propulsion Lab, USA

Program Chair
Martin Glinz, Universität Zürich, Switzerland

Local Arrangements
Mats Heimdahl, University of Minnesota, USA

Financial Chair
Jane Cleland-Huang, DePaul University, USA

Practitioner Tracks
Ian Alexander, Independent Consultant, UK
Frank Houdek, DaimlerChrysler Research, Germany

Call for Papers and Proposals

Understanding the stakeholders' desires and needs

Requirements engineering has increasingly become a dominant activity in systems development—the more we can generate or outsource design and construction, the more we need requirements that adequately reflect the stakeholders' desires and needs.

The IEEE International Requirements Engineering conference is the premier requirements engineering conference, providing a forum for researchers, practitioners, educators, and students to present and discuss the most recent innovations, trends, experiences, and concerns in the field of requirements engineering.

focus

guest editors' introduction

A guest editors' introduction with > 100 citations

Stakeholders in Requirements Engineering

Martin Glinz, *University of Zurich*

Roel J. Wieringa, *University of Twente*

To build a useful system, we need to know its requirements; to know its requirements, we need to know the stakeholders' desires and needs. The term *stakeholder* generalizes the

My life with RE – in five stories

1 The early days

2 Requirements modeling

3 Stakeholders

4 Value-orientation

5 Shared understanding

2003: Two disturbing questions

Why not just code what the stakeholders desire and need?

Why can agile projects succeed with little or even no RE?

Hmmm.

Another personal paradigm shift

DEFINITION [2003]. Requirements Engineering – Specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs.

I created this definition when preparing an invited talk on Requirements Engineering at the Swiss Association for Quality in September 2003. It is now part of the IREB Glossary of Requirements Engineering Terminology [Glinz 2014].

A paradigm shift with consequences

If RE is considered a means for controlling risk,

... then how much RE do we need?

What's the value of requirements?

How can we achieve specifications that create optimal value?

The value of requirements

Value means

- The benefit of an explicit specification

  Bringing down the probability for developing a system that doesn't satisfy its stakeholders' expectations and needs to an acceptable level

minus

- The cost of writing, reading and maintaining this specification

The eleventh commandment ... of traditional RE

Thou shalt quantify

... all your quality requirements

... to make them unambiguous and verifiable.

If it's not measurable, make it measurable.

Regardless of the effort involved?

Does this create optimal value?

A new approach to quality requirements

focus

quality requirements

A Risk-Based, Value-Oriented Approach to Quality Requirements

Martin Glinz, *University of Zurich*

This value-oriented approach to specifying quality

When quality requirements are elicited from stakeholders, stated qualitatively, such as "the response time must be fast" a highly available system." (See the "Defining Quality R sidebar for a definition of quality requirements.) However

IEEE Software
Vol. 25, No. 2
March/April 2008
pp. 34-41

This is part of my work on non-functional requirements. This work constitutes another story in my professional life which I skipped in this talk to keep it at a reasonable length.
[Glinz 2005, 2007, 2008, 2016].

My life with RE – in five stories

1 The early days

2 Requirements modeling

3 Stakeholders

4 Value-orientation

5 Shared understanding

2012: Back to a fundamental problem

How much to invest in RE

... to secure successful communication of desires and needs

... between stakeholders and developers?

→ Study the problem of shared understanding

Shared Understanding

"The ticketing system shall provide discounted tickets which are for sale only to skiers staying in one of the resort's hotels and are valid from the first to the last day of the guest's stay."

Do all involved parties have the same understanding of this requirement?

If so, do we need to specify it explicitly or can we rely on implicit shared understanding?

Comput Sci Res Dev (2015) 30:363–376
DOI 10.1007/s00450-014-0256-x

SPECIAL ISSUE PAPER

On shared understanding in software engineering: an essay

Martin Glinz · Samuel A. Fricker

Published online: 1 July 2014

Abstract Shared understanding is essential for efficient software engineering when the risk of unsatisfactory outcome and rework of project results shall be low. Today, however, shared understanding is used mostly in an unreflected, ad-hoc way. This affects the quality of the engineered soft-

users, customers, operators, and managers are typical examples of stakeholders. A *software engineer* is a person involved in the specification, design, construction, deployment, evolution, and maintenance of software systems. Requirements engineers, architects, developers, coders, and testers are

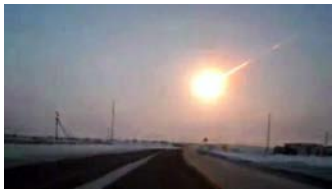
Can we rely on implicit shared understanding?

This requires:

- Low probability for misunderstandings
- Reduction of impact of misunderstandings

- Impact =
  - Cost for detection and correction

  +
  - Cost created by
    - Non-satisfied stakeholders
    - Rework



→ We need techniques for building and assessing shared understanding

Towards a bright future of RE

New emerging topics

- Requirements evolution
- User feedback, runtime monitoring, requirements mining
- Complementarity of requirements and product design: a new paradigm shift for RE on the horizon

A comeback of requirements modeling?

Requirements Engineering is in a healthy and thriving state

... due to many engaged people with whom I had/have the pleasure to work with

...and many others who worked with me or were advised by me

Let me say thank you

To all of my former and current PhD students and Postdocs,

To my academic teachers and peers, as well as my colleagues from practice,

To my students and course participants,

To my wife and my family,

And to everybody in the audience for honoring me with their presence today.

References

BBC Brown Boveri (1986a). *SPADES Language Manual*, version 2.3. Document Ex CTT 86/30.

BBC Brown Boveri (1986b). *SPADES Tool User's Manual*, version 2.3. Document Ex CTT 86/33.

S. Berner, S. Joos, M. Glinz, M. Arnold (1998). A Visualization Concept for Hierarchical Object Models. *13th IEEE Conference on Automated Software Engineering (ASE 1998)*, Honolulu, Hawaii. 225–228.

Cadre Technologies Inc. (1986). *Cruise Control Requirements.* White paper (Note 54, 11 Jul 86).

T. DeMarco (1979). *Structured Analysis and System Specification*. New York: Yourdon Press.

M. Glinz, J. Ludewig (1984). *SEED – Das Datenbanksystem für die Software- Entwicklungsumgebung SEEME*. Brown Boveri Research Center, Research Report KLR 84-143 C.

M. Glinz, J. Ludewig (1986). SEED – A DBMS for Software Engineering Applications Based on the Entity-Relationship Approach. *2nd International Conference on Data Engineering*, Los Angeles. 654–660.

M. Glinz (1991). Probleme und Schwachstellen der Strukturierten Analyse. In M. Timm, (ed.): *Requirements Engineering '91*. Informatik-Fachberichte vol. 273, Berlin: Springer. 14–39.

M. Glinz (1993). Hierarchische Verhaltensbeschreibung in objekt-orientierten Systemmodellen – eine Grundlage für modellbasiertes Prototyping. *GI-Symposium Requirements Engineering '93*. Report no. 41 of the German Chapter of the ACM, Stuttgart: Teubner. 175–192.

M. Glinz (1995). An Integrated Formal Model of Scenarios Based on Statecharts. In W. Schäfer and P. Botella (eds.): *Software Engineering – ESEC'95. Proceedings of the 5th European Software Engineering Conference*, Sitges, Spain. Lecture Notes in Computer Science 989, Berlin: Springer. 254–271.

References – 2

M. Glinz (2000). Problems and Deficiencies of UML as a Requirements Specification Language. *Tenth International Workshop on Software Specification and Design*. San Diego, Ca. 11–22.

M. Glinz, S. Berner, S. Joos (2002). Object-Oriented Modeling with ADORA. *Information Systems* 27(6):425–444.

Glinz, M. (2005). Rethinking the Notion of Non-Functional Requirements. *Third World Congress for Software Quality (3WCSQ 2005)*, Munich, Germany, Vol. II, 55-64.

M. Glinz (2007). On Non-Functional Requirements. *15th IEEE International Requirements Engineering Conference (RE'07)*, Delhi, India. 21–26.

M. Glinz, R.J. Wieringa (2007) Stakeholders in Requirements Engineering. *IEEE Software* 24(2):18–20.

M. Glinz (2008). A Risk-Based, Value-Oriented Approach to Quality Requirements. *IEEE Software* 25(2):34–41.

M. Glinz (2010). Very Lightweight Requirements Modeling. Invited contribution, *18th IEEE International Requirements Engineering Conference (RE'10)*, Sydney, Australia. 385–386.

M. Glinz (2014). *A Glossary of Requirements Engineering Terminology*, Version 1.6. International Requirements Engineering Board (IREB). https://www.ireb.org/de/downloads/#cpre-glossary, visited 2017-05-06

M. Glinz, S.A. Fricker (2015). On Shared Understanding in Software Engineering: An Essay. *Computer Science – Research&Development* 30(3-4): 363–376.

M. Glinz (2016). How Much Requirements Engineering Do We Need? *Softwaretechnik-Trends* 36(3):19–21.

References – 3

D. Harel (1987). Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming* 8(3):231–274.

D.J. Hatley, I.A. Pirbhai (1987). *Strategies for Real-Time System Specification*. New York: Dorset House.

G. Hommel, D. Krönig (eds.)(1983). *Requirements Engineering*. GI working conference, Friedrichshafen, Germany. Informatik-Fachberichte vol. 74, Berlin: Springer.

IEEE (1990). *Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990. IEEE Computer Society Press.

J. Ludewig (1983). ESPRESO - A System for Process Control Software Specification. *IEEE Transactions on Software Engineering* 9(4): 427–436.

J. Ludewig, M. Glinz, H. Huser, G. Matheis, H. Matheis, M. F. Schmidt (1985). SPADES - A Specification and Design System and Its Graphical Interface. *8th International Conference on Software Engineering (ICSE'85)*, London, UK. 83–91.

T. Reinhard, C. Seybold, S. Meier, M. Glinz, N. Merlo-Schett (2006). Human-Friendly Line Routing for Hierarchical Diagrams. *21st International Conference on Automated Software Engineering (ASE 2006)*, Tokyo, Japan. 273–276.

T. Reinhard, S. Meier, R. Stoiber, C. Cramer, M. Glinz (2008) Tool Support for the Navigation in Graphical Models. *30th International Conference on Software Engineering (ICSE 2008)*, Leipzig, Germany. 823–826.

D. T. Ross (Ed.) (1977). Special Collection on Requirement Analysis. *IEEE Transactions on Software Engineering* SE-3(1).

References – 4

D. T. Ross (1977). Structured Analysis (SA): A Language for Communicating Ideas. *IEEE Transactions on Software Engineering* SE-3(1):16–34.

D. Teichroew and E. A. Hershey III (1977). PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems. *IEEE Transactions on Software Engineering* SE-3(1):41–48, 1977.

P.T. Ward, S.J. Mellor (1985). *Structured Development for Real-Time Systems, Vol. I-III*. Englewood Cliffs, N.J.: Prentice-Hall.

D. Wüest, M. Glinz (2011). Flexible Sketch-Based Requirements Modeling. *17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2011)*, Essen, Germany. 100–105.

D. Wüest, N. Seyff, M. Glinz (2013). Semi-automatic Generation of Metamodels from Model Sketches. *28th International Conference on Automated Software Engineering (ASE 2013)*, Palo Alto, Ca. 664–669.

D. Wüest, N. Seyff, M. Glinz (2015a). FLEXISKETCH TEAM: Collaborative Sketching and Notation Creation on the Fly. *37th International Conference on Software Engineering (ICSE 2015)*, Volume 2, Florence, Italy. 685–688.

D. Wüest, N. Seyff, M. Glinz (2015b). Sketching and Notation Creation with FlexiSketch Team: Evaluating a New Means for Collaborative Requirements Elicitation. *23rd IEEE International Requirements Engineering Conference (RE'15)*, Ottawa, Canada. 186–195.

A comprehensive list of my publications is available at http://www.ifi.uzh.ch/en/rerg/publications.html.