Requirements Engineering
Ein persönlicher Rückblick und Ausblick
A Personal Reflection

Martin Glinz
www.ifi.uzh.ch/erg

Abschiedsvorlesung
Farewell Lecture
2017-05-09
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
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

The title, "Define graphics," is identical to the name inside the first box of Fig. 3, which is here being broken into five component worthy pieces, called the nested factors in SA terminology. Again the words written inside the boxes are legible, but are they understandable? How can "Bound context; relate/connect; show transformation; show circumstance; show means," be considered parts of "Define graphics?"

It is not very clear, is it?

It would seem that something must be wrong with SA for the touted understandability turns out to be, in fact, quite obscure!

Look at Fig. 4 again and see if you don't agree that we have a problem—and see if you can supply the answer to the problem.

The problem is not with SA at all, but with our too-glib approach to it. SA is a rigorous language and thereby is unforgiving in many ways. In order for the communication of understanding to take place we ourselves must understand and conform to the rules of that rigor. The apparent obscurity should disappear in a twinkling once the following factor is pointed out: namely, always be sure to do your understanding in the proper context.

In this case, the proper context was established by the title of Fig. 3, "Rationalize structured analysis features," and the purpose, to define graphical concepts and notations for the purpose of representing and conveying understanding of subject matters. Now, if we have all of that firmly implanted in our mind, then surely the name in Box 1 of Fig. 4 should be amply clear. Read, now, Box 4. 1 for yourself, and see if that clarity and communication of intended understanding does not take place.

3 To shorten references to figures, "Box 4. 1" will mean "Box 1 of Fig. 4," etc. in the following discussion.
Some examples: Semi-formal text

Structured textual requirements specification

Based on an entity-relationship model

PSL [Teichroew & Hershey 1977]
ESPRESSO [Ludewig 1983]
SPADES [Ludewig et al. 1985]
Some examples: Structured Analysis

Based on a hierarchy of dataflow diagrams

Later extended with capabilities to model state-dependent behavior

[DeMarco 1979]
Some examples: STDs, statecharts

Specification of event-driven, state-dependent system behavior

State-transition diagrams, statecharts

In inactive mode, the system switches to normal mode when count = 0.

In normal mode, when a card is sensed, validation begins.

- **Validating state**
  - Flash red light
  - If the card is invalid, the system remains in the validating state.
  - If the card is valid, the count is incremented, and the system progresses to the open state.

In the validating state, the system waits for the card to be validated.

In the open state, the system allows one turn:
- **Count** = **Count** +1
- Flash green light

In the open state, the system one turn done.

The state diagram includes transitions for:
- Start
- Stop
- Stop
- Close flush value
- Flush passed
- Open flush value

**IDLE**
- Start
- Stop
- Stop
- Close flush value
- Flush passed
- Open flush value

**HEATING**
- Reaction time reached
- Disable "heating"
- Enable "maintaining"
- Reaction time elapsed
- Disable "maintaining"
- Wait on flush

**MAINTAINING**
- Reacting time reached
- Disable "heating"
- Enable "maintaining"
- Reaction time elapsed
- Disable "maintaining"
- Wait on flush

**WAITING**
- Flush passed
- Open flush value

[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

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

requirements engineering – a personal reflection
© 2017 Martin Glinz
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

- **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 ...

Requirements Engineering – A Personal Reflection © 2017 Martin Glinz
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

[Glinz 1991]
1991-2011: Object-oriented modeling of requirements

- Hierarchically decomposed object models [GI-RE’93]
- based on scenarios and statecharts [ESEC 1995]
- Our focus activity for many years
ADORABasic 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

[A] [B] [C] [D] 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
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

Mobile

Collaborative

Freeform sketching

Assign meanings through annotations

Automatic inference

Multi-Platform

Identify similar symbols

beautification
discovery. One such solution is the Collaborative Creativity Canvas (see Figures 2b and 4). Facilitators can use it to foster creativity and replace the often frustrating requirements negotiation process with a lively cocreation process. It aims to turn stakeholder conflicts into opportunities for innovation.

Although Martin’s research included traditional literature reviews, expert opinion, and practitioner surveys, it was driven largely by industrial collaboration. Ideas conceived in industry were iteratively and incrementally improved as they moved back and forth between the lab and practice. As such, Martin’s creativity solutions resulted from industrial partnerships and not through the more traditional model in which an idea emerges from research and then incubates in a lab for five years before the finished product is offered to industry.

Martin explained that this project revealed the benefits of industrial co-design, especially through the ongoing guidance and feedback he obtained. He pointed out that working with industry didn’t produce short-sighted research because he had the time and freedom to consider, and explore, innovative ideas throughout the process.

Archie
Architectural knowledge and related quality concerns are often undocumented and tacit. So, developers often lose track of early design decisions. For example, system-level qualities representing “nonfunctional” requirements tend to become eroded during refactoring, bug fixing, and other maintenance activities.

To address this problem, my research group at DePaul University developed Archie (see Figures 2c and 5), an Eclipse plug-in. Archie focuses on requirements’ role in a project’s downstream design and maintenance phases. It parses source code and then automatically detects and visualizes a range of architectural tactics such as heartbeats, resource pooling, and role-based access control.

Archie was funded by grants from the US National Science Foundation and Department of Homeland Security and developed by Ahmed Fahkry (see Figure 6) and other students under Mehdi Mirakhorli’s supervision. To place Archie into practitioners’ hands, we released it on GitHub under Archie-Smart-IDE and on the Department of Homeland Security’s SWAMP (Software Assurance Marketplace).

Mirakhorli explained that one of the greatest challenges for technology transfer was in understanding the real users’ actual usage patterns. We addressed this through frequent iterations of prototyping, coding, and testing. However, the real test will come as industrial users adopt Archie in their development environments. As such, Archie is less advanced along the technology-transfer scale than FlexiSketch or Martin’s creative collaboration activities.

FIGURE 4. Participants in one of Martin Mahaux’s workshops use the Collaborative Creativity Canvas to explore innovative requirements ideas. For more details, see http://bit.ly/martinmahaux.

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.
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

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.

Topics of interest include, but are not restricted to: requirements elicitation, analysis, documentation, analysis, specification, validation, and verification.
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 traditional notion of *customer* or *user* in requirements engineering to all parties involved in a system’s requirements (see the sidebar “What Is a Stakeholder?”). The growing attention being paid to stakeholders’ needs and desires reflects the growing importance of...
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

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

Martin Glinz, University of Zurich

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 Re
guidance for a definition of quality requirements.) However,

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?
On shared understanding in software engineering: an essay

Martin Glinz · Samuel A. Fricker

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-
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

References – 2


