Lecture 02:
Requirements Engineering – Part 1

Dietmar Pfahl
email: dietmar.pfahl@ut.ee

Fall 2017
Project Teams & Exam Dates

THANK YOU for being very co-operative (and fast) with forming project teams!

Exam Dates:

• Exam 1: Tuesday, 09-Jan-2018 at 14:15-16:45, room 405 (limit: 49 students)
• Exam 2: Friday, 12-Jan-2018 at 14:15-16:45, room 405 (limit: 49 students)
• Exam 3: Friday, 19-Jan-2018 at 14:15-16:45, room 405 (limit: 49 students)
• Re-take Exam: Friday, 26-Jan-2018 at 14:15-16:45, room J. Liivi 2-611
Lab 1 Assignment

Joostes Marss AS

Customer: Home Improvement International (HII)

POS System

To do:
- Interviews to identify roles and requirements (user stories)
- List of 20 functional user stories
- List of 10 non-functional user stories (performance & usability)
Recap from last week
‘Normal’ User Story

<Actor/Role> As a user

... I want to narrow down people search results by location

... so I can find the right person more quickly

A good User Story is:

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

INVEST
Quality User Story Framework (13 Criteria – extending INVEST)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
<th>Individual/set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntactic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Well-formed</td>
<td>A user story includes at least a role and a means</td>
<td>Individual</td>
</tr>
<tr>
<td>Atomic</td>
<td>A user story expresses a requirement for exactly one feature</td>
<td>Individual</td>
</tr>
<tr>
<td>Minimal</td>
<td>A user story contains nothing more than role, means, and ends</td>
<td>Individual</td>
</tr>
<tr>
<td><strong>Semantic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conceptually sound</td>
<td>The means expresses a feature and the ends expresses a rationale</td>
<td>Individual</td>
</tr>
<tr>
<td>Problem-oriented</td>
<td>A user story only specifies the problem, not the solution to it</td>
<td>Individual</td>
</tr>
<tr>
<td>Unambiguous</td>
<td>A user story avoids terms or abstractions that lead to multiple interpretations</td>
<td>Individual</td>
</tr>
<tr>
<td>Conflict-free</td>
<td>A user story should not be inconsistent with any other user story</td>
<td>Set</td>
</tr>
<tr>
<td><strong>Pragmatic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full sentence</td>
<td>A user story is a well-formed full sentence</td>
<td>Individual</td>
</tr>
<tr>
<td>Estimatable</td>
<td>A story does not denote a coarse-grained requirement that is difficult to plan and prioritize</td>
<td>Individual</td>
</tr>
<tr>
<td>Unique</td>
<td>Every user story is unique, duplicates are avoided</td>
<td>Set</td>
</tr>
<tr>
<td>Uniform</td>
<td>All user stories in a specification employ the same template</td>
<td>Set</td>
</tr>
<tr>
<td>Independent</td>
<td>The user story is self-contained and has no inherent dependencies on other stories</td>
<td>Set</td>
</tr>
<tr>
<td>Complete</td>
<td>Implementing a set of user stories creates a feature-complete application, no steps are missing</td>
<td>Set</td>
</tr>
</tbody>
</table>
User Story Example – Wrong or Right?

As a care professional I want to save a reimbursement—add save button on top right (never grayed out).
User Story Example – **Wrong** or Right?

As a care professional I want to save a reimbursement—add save button on top right (never grayed out).

Not **problem-oriented** -> Hints at the solution
User Story Example – Wrong or Right?

As a User, I am able to edit the content that I added to a person’s profile page.
User Story Example – **Wrong** or Right?

As a User, I am able to edit the content that I added to a person's profile page.

Not **unambiguous** -> what is content?
User Story Example – Wrong or Right?

As a care professional I want to see my route list for next/future days, so that I can prepare myself (for example I can see at what time I should start traveling).
User Story Example – **Wrong** or **Right**?

As a care professional I want to see my route list for next/future days, so that I can prepare myself (for example I can see at what time I should start traveling).

Not estimatable -> it is unclear what ‘see my route list’ implies
User Story Example – Wrong or Right?

As an Administrator, I am able to add a new person to the database.

As a Visitor, I am able to view a person’s profile
User Story Example – **Wrong** or Right?

As an Administrator, I am able to add a new person to the database.

As a Visitor, I am able to view a person’s profile

Not independent -> viewing relies on first adding a person to the database
Schedule of Lectures (Tentative)

Week 01: Introduction to SE
Week 02: Requirements Engineering I
Week 03: Requirements Engineering II
Week 04: Analysis
Week 05: Development Infrastructure I
Week 06: Development Infrastructure II
Week 07: Architecture and Design
Week 08: Refactoring
Week 09: Verification and Validation I
Week 10: Crowdsourced Testing
Week 11: Continuous Development and Integration
Week 12: Agile/Lean Methods
Week 13: Software Craftsmanship
Week 14: Course wrap-up, review and exam preparation
Week 15: no lecture
Goal of this Lecture:
To give answers to the following questions …

What is ‘Requirements Engineering’?

Why is RE important?

Why is RE difficult?

Who is involved in RE?

What are ‘Requirements’?

What types of requirements exist?

What levels of requirements exist?

What process steps are involved in RE?

How to get started with RE?

How to elicit requirements?

How to represent/document requirements?

How to use requirements for project planning?
Goal of this Lecture:
To give answers to the following questions …

What is ‘Requirements Engineering’?

Why is RE important?

Why is RE difficult?

Who is involved in RE?

What are ‘Requirements’?

What types of requirements exist?

What levels of requirements exist?

What process steps are involved in RE?

How to get started with RE?

How to elicit requirements?

How to represent/document requirements?

How to use requirements for project planning?

to be continued next week
Structure of Lecture 02

- The Nature of Requirements Engineering (RE)
- The Nature of Requirements
- The Process of RE
- Project Planning based on Requirements
Definition: Requirements Engineering

RE is the process of establishing

• the services that the customer requires from a system

and

• the constraints under which it operates and is developed.

RE means to ...

... dig up, understand, write down, check, prioritize, select, follow up on ...

... the functions and properties of (software) products
The Goal of RE

What the Customer wants
What the Customer needs
What the Software (to-be) does

Application Domain
(User Requirements)

System Domain
(System Requirements)
Why is RE important?

Requirements (Specification) are central for development and verification / validation.

- Stakeholders
- Demands
- Elicitation
- Analysis
- Requirements Specification (Reqspec)
- Contract
- Design
- Program
- Test
- Operation & Maintenance

Tracing:
- Forwards...
- Backwards...

Req. management:
- Changing reqs

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Economic Consequences of RE Problems

[Davis, 1992]
RE is difficult, because …

- It typically involves many stakeholders.
- Stakeholders (often) don’t know what they really want.
- Stakeholders express requirements in their own terms (might be imprecise, ambiguous).
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- New stakeholders may emerge and the business environment change.
- The requirements change during the analysis process.
Stakeholders in SE

External:
- Customers
  - Those who pay for the software
- Users
  - Those who use the software

Internal:
- Software Developers
- Development (Project) Managers
- Product Managers
  - ...

© Dietmar Pfahl 2017
Example of a Real World Situation (Mobile Phones)

External stakeholders
- Customers
  - Direct customers
  - Operators
  - Global customers
  - Regional customers
  - Other key customers
- Retailers
- Indirect customers
- Consumers
  - Market segments
  - Service providers
  - Content providers

Product providers
- Direct Competitors
- Mobile phone developers
- Indirect Competitors
- Cameras
- Mobile music players
  - ... consumer wallet competition

Platform providers
- Operating Systems
- Technical Platforms
- Network system providers
- Standardization bodies
- Legislation and authorities
- National
- International
- Manufacturing sub-contractors
- Component providers

... find the right person to talk to ...
... get the deep domain knowledge ...
Example: Ambiguous Requirement

The tiny word ’only’

Version 1:
• *The spam filter* *only delivers* the e-mail that the user wants.

Version 2:
• *The spam filter* *delivers only* the e-mail that the user wants.

Same meaning? If not, what’s the difference?
Example: Ambiguous Requirement

The tiny word ‘only’

Version 1:
• The spam filter *only delivers* the e-mail that the user wants.

Version 2:
• The spam filter *delivers only* the e-mail that the user wants.

(Unfortunately, Version 2 is not considered correct English in the UK. The word ‘only’ must always go before the main verb.)
Example: Ambiguous Requirement

- Requirement: ‘A user of the Library Information System (LIS) shall be able to search the recent publications lists for all libraries.’

?
Example: Ambiguous Requirement

- Requirement: ‘A user of the Library Information System (LIS) shall be able to search the recent publications lists for all libraries.’

- Consider the term ‘search … for all’:

  - User intention: search for a publication across all recent publications lists in all libraries;
  - Developer interpretation: search for a publication in an individual recent publications list. User first chooses library then searches list.

Imprecise (ambiguous) requirements may be interpreted in different ways by developers and users.
Example: Ambiguous Requirement

• Requirement: ‘A user of the Library Information System (LIS) shall be able to search the recent publications lists for all libraries.’

• Consider the term ‘search … for all’:

  ![Diagram showing the confusion between user intention and developer interpretation.](image)
Example: Ambiguous Requirement

- Requirement: ‘A user of the Library Information System (LIS) shall be able to search the recent publications lists for all libraries.’

- Consider the term ‘search … for all’:
Example: Ambiguous Requirement

- Requirement: ‘A user of the Library Information System (LIS) shall be able to search the recent publications lists for all libraries.’
- Consider the term ‘search … for all’:
  - User intention: search for a publication across all recent publications lists in all libraries;
  - Developer interpretation: search for a publication in an individual recent publications list. User first chooses library then searches list.
- Imprecise (ambiguous) requirements may be interpreted in different ways by developers and users.
Example: Conflicting Requirements

• A performance requirement may indicate that
  • a core system must be **updated in real time**
  but
  • the **size and scope** of the system (as defined by other requirements) may preclude this.
  Updating such a large system may not be possible in real time.

• Need to apply conflict resolution procedures (→ negotiation with stakeholders)
SMART Requirements

• Specific
• Measurable
• Attainable (Achievable, Actionable, Appropriate)
• Realistic
• Time-bound (Timely, Traceable)

Note: The SMART analysis may also be applied to Tasks!

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
SMART Requirements

Counter-example (i.e., not SMART):

’The user interface of system xyz should look nice to all users and the response time to inquiries should be as fast as the speed of light’

S  Specific
M  Measurable
A  Attainable (Achievable, Actionable, Appropriate)
R  Realistic
T  Time-bound (Timely, Traceable)
SMART Requirements

Counter-example:

'The user interface of system xyz should look nice to all users and the response time to inquiries should be as fast as the speed of light'

- Not specific: Compund (2 requs)
- Not measurable: 'nice'?
- Not attainable: 'all' users will find that the UI looks nice
- Not realistic: Speed of light cannot be achieved
- Time-bound/Timely: To check this we would need to know the project plan and the planned date when the required function shall be available
- Traceable: Would need other documents to be able to check whether the req is traceable (BUT: since it is a compound, there is already some potential complication of traceability)
SMART Requirements

- **Specific**
  
  A good requirement is specific and not generic. It should not be open to misinterpretation when read by others.  
  - Avoid using conjunctions (and, or, but)  
  - Avoid indeterminate amounts of time (soon, fast, later, immediately)  
  - Etc.

- **Measurable**
- **Attainable** (Achievable, Actionable, Appropriate)
- **Realistic**
- **Time-bound** (Timely, Traceable)

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
SMART Requirements

- **Specific**
  This answers whether you will be able to verify the completion of the project. You should avoid signing up for any requirement that cannot be verified as complete.
  - These are especially risky when you use non-quantitative terms (best, optimal, fastest) for acceptance criteria.

- **Measurable**

- **Attainable** (Achievable, Actionable, Appropriate)

- **Realistic**

- **Time-bound** (Timely, Traceable)

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
SMART Requirements

• Specific
  This intends to ensure that the requirement is physically and logically possible to be achieved given existing circumstances. There is arguably overlap between attainable and realistic.
  • Reserve attainable to check the likelihood that it will be possible to achieve the requirement

• Measurable

• Attainable (Achievable, Actionable, Appropriate)

• Realistic

• Time-bound (Timely, Traceable)

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
SMART Requirements

• Specific
• Measurable
• Attainable (Achievable, Actionable, Appropriate)

This intends to ensure that the requirement is realistic to deliver when considering other constraints of the project and requirements.

• Realistic

• Time-bound (Timely, Traceable)

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
SMART Requirements

- Specific
- Measurable
- Attainable (Achievable, Actionable, Appropriate)
  - Where appropriate each requirement should be time-bound or specify by *when* or *how fast* a required function needs to be completed or executed.
- Realistic
- **Time-bound** (Timely, Traceable)

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
Structure of Lecture 02

- The Nature of Requirements Engineering (RE)
- The Nature of Requirements
- The Process of RE
- Project Planning based on Requirements
Functional vs. Non-Functional Requirements

**Functional reqs:**
- What the system shall do
- Often intended to be implemented as a whole or else not implemented at all
- Often regards input/output data and functions that process the input data to produce the output

**Non-functional reqs (NFR), Quality Requirements, (extra-functional reqs):**
- How good the system shall do it
- Often measured on a scale
- Often put constraints on the system (or the development process)
- Often cross-cutting; may impact many functions

Performance
Reliability
Usability
Safety, Security
Interoperability
Maintainability
...

But the division is not black and white...
What non-functional (=quality) requirements do you expect from a Student Information System (SIS)?
SW Product Quality -> ISO 9126 (now ISO 25000)

Efficiency = Performance

Software Quality Model
(ISO 9126 – Part 1)
### Example – Efficiency Requirements

**Performance requirements:**

- **R1:** Product shall be able to process 100 payment transactions per second in peak load.
- **R2:** Product shall be able to process one alarm in 1 second, 1000 alarms in 5 seconds.
- **R3:** In standard work load, CPU usage shall be less than 50% leaving 50% for background jobs.
- **R4:** Scrolling one page up or down in a 200 page document shall take at most 1 s. Searching for a specific keyword shall take at most 5 s.
- **R5:** When moving to the next field, typing must be possible within 0.2 s. When switching to the next screen, typing must be possible within 1.3 s. Showing simple report screens, less than 20 s. (Valid for 95% of the cases in standard load)
- **R6:** A simple report shall take less than 20 s for 95% of the cases. None shall take above 80s. (UNREALISTIC)

---

Cover all product functions?

---

From: Soren Lauesen: Software Requirements  
© Pearson / Addison-Wesley 2002
Example – Usability Requirements

<table>
<thead>
<tr>
<th>Problem counts</th>
<th>Design-level reqs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: At most 1 of 5 novices shall encounter critical problems during tasks Q and R. At most 5 medium problems on list.</td>
<td>R6: System shall use screen pictures in app. xx, buttons work as app. yy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task time</th>
<th>Product-level reqs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2: Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes.</td>
<td>R7: For all code fields, user shall be able to select value from drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keystroke counts</th>
<th>Guideline adherence</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3: Recording breakfast shall be possible guest. No mouse.</td>
<td>R8: System shall follow style guide zz. Menus shall have at most three levels.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opinion poll</th>
<th>Development process reqs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R4: 80% of users shall find system easy to recommend system to others.</td>
<td>R9: Three prototype versions shall be made and usability tested during design.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Score for understanding</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R5: Show 5 users 10 common error messages large. Ask for the cause. 80% of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From Goal to Design

Requirements can be formulated at various levels:

- **Goal-level requirement**
  - R1. Our pre-calculations shall hit within 5%
  - Underlying purpose, business, goals, expected/intended improvements

- **Domain-level requirement**
  - R2. Product shall support cost recording and quotation with experience data
  - Context, how user and system-to-be-developed collaborate in order to achieve the goals

- **Product-level requirement**
  - R3. Product shall have recording and retrieval functions for experience data
  - Externally visible functions and properties of the system

- **Design-level requirement**
  - R4. System shall have screen pictures as shown in app. xx
  - Precise description of data, functions, user-interfaces, etc.

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Structure of Lecture 02

• The Nature of Requirements Engineering (RE)
• The Nature of Requirements
• The Process of RE
• Project Planning based on Requirements
RE Activities

Requirements gathering (= Requirements elicitation)

- Interacting with stakeholders to discover their requirements:
  - What is to be accomplished?
  - How the system will fit into the needs of the business?
  - How the system will be used on a day-to-day basis?

Requirements analysis

- Refining, classifying/clustering, structuring, prioritizing, and modifying the gathered requirements

Requirements specification

- Documenting the (system) requirements in a semiformal or formal manner to ensure clarity, consistency, and completeness

Requirements validation

- Checking the requirements
RE Activities: Iteration & Concurrency

- Initial information
- Scope
- Constraints

Model

- Elicit
- Analyze
-Specific
- Validate

Requirements traced back to their source

+ Requirements Management
Where/How do we start?
Where/How do we start?

**Identify the problem**
- what is the objective of the project?
- the “vision” of those who are pushing for it?
  - e.g., “Meeting scheduling is too costly right now”

**Identify solution scenarios**
- given the problem, what is the appropriate business process for solving it?
  - e.g. “Anyone who wants to schedule a meeting goes to the secretary, gives details and the secretary handles the rest”, …or…

**Scope the problem**
- given the vision, how much do we tackle?
  - e.g. “Build a system that schedules meetings”, …or…
  - e.g. “Build a system that maintains people’s calendars” …or…

**Scope the solution**
- Given a business process, what parts should be automated, and how?
  - e.g. “Computer takes in scheduling request details, outputs a solution” …or…
  - e.g. “Solution arrived at interactively by secretary and computer” …or…
Difficulties of Elicitation

• Implicit (tacit) knowledge / Limited observability
• Conflicting information / Thin spread (distributed) domain knowledge
• Say-do problem
• Probe (Hawthorne) effect
• Bias
Example: Elicit the rules and procedures for approving a loan

Why this might be difficult?

- **Implicit knowledge:**
  - There is no document in which the rules for approving loans are written down

- **Conflicting information:**
  - Different bank staff have different ideas about what the rules are

- **Say-do problem:**
  - The loan approval process described to you by the loan approval officers is quite different from your observations of what they actually do

- **Probe effect:**
  - The loan approval process used by the officers while you are observing is different from the one they normally use

- **Bias:**
  - The loan approval officers fear that your job is to computerize their jobs out of existence, so they are deliberately emphasizing the need for case-by-case discretion (to convince you it has to be done by a human!)
Elicitation Techniques

• Analyzing existing documents & data
• “Brainstorming” possible requirements
  • within the dev. Team
• Interviews (one-on-one)
  • This is what you need to do for Lab 1…
• Focus groups or workshops (one-on-many)
  • useful for larger projects
• Prototyping/mockups
• Meetings with the customer/users
  • E.g. for checkpoints, or showing prototypes
Analyzing documents & data

Sources of information:

• company reports, organization charts, policy manuals, job descriptions, reports, documentation of existing systems, etc.

Advantages:

• Helps the analyst to get an understanding of the organization before meeting the people who work there
• Helps to prepare for other types of fact finding
  e.g. by being aware of the business objectives of the organization.
• may provide detailed requirements for the current system

Disadvantages:

• written documents often do not match up to reality
• Can be long-winded with much irrelevant detail
Interviews

Types:

Structured - agenda of fairly open questions
Open-ended - no pre-set agenda

Advantages

Rich collection of information
Good for uncovering opinions, feelings, goals, as well as hard facts
Can probe in depth, & adapt follow-up questions to what the person tells you

Disadvantages

Large amount of qualitative data can be hard to analyze
Interviewing is a difficult skill to master

Source: Adapted from Goguen and Linde, 1993, p154.
Investigate the “problem”/”opportunity”

- What (Which) problem needs to be solved?
  - identify problem Boundaries
- What might prevent us solving it?
  - identify Feasibility and Risk
- Where is the problem?
  - understand the Context/Problem Domain
- Whose problem is it? Who is affected?
  - identify Stakeholders
- Why does it need solving?
  - identify the stakeholders’ Goals
- When does it need solving?
  - identify Development Constraints
- How does the problem manifest itself?
  - collect some Scenarios

W6H
The journalist’s technique:
What?
(Which?)
Where?
Who?
Why?
When?
How?
Meetings

• Used for summarization and feedback
  • E.g. meet with stakeholders towards the end of each stage:

• Every meeting should have a clear objective:
  • E.g. presentation, problem solving, conflict resolution, progress analysis, gathering and merging of facts, training, planning,...

• Plan the meeting carefully:
  • Schedule the meeting and arrange for facilities
  • Prepare an agenda and distribute it well in advance
  • Keep track of time and agenda during the meeting
  • Follow up with a summary to be distributed to meeting participants
Prototyping (a.k.a. mockups)

- Paper prototyping
- Wireframes
- Interactive wireframes
- Rich interactive prototypes
  - e.g. Concept.ly

*People often don’t know what they want until they see what they can get.*

A. Wildavsky – Speaking Truth to Power
Combine Different Techniques

- Background reading (e.g., Internet?)
- (Initial) Meeting
- Hard Data analysis
- Brainstorming
- Interviews
- Prototyping
- Meeting
- ...

- ...
Representation Styles

- Natural language (plus supporting tables and graphs)
- Structured natural language / Scenarios
  - e.g., user, stories, use case descriptions, CRC cards, ...
- Semi-formal notations
  - e.g., UML diagrams (use case diagrams, class diagrams, state charts, sequence charts, etc.)
- Formal notations (with formal semantics)
  - e.g., abstract model-based (VDM, Larch, B, ...) or algebraic (OBJ, ACT-ONE, ...)
Example: Home Access Control

Objective:
Design an electronic system for:

• Home access control
  • Locks and lighting operation
• Intrusion detection and warning

Diagram:
- Lock
- Photosensor
- Switch
- Alarm bell
- Light bulb

System

1 2 3 4 5
X Y
Example – More Details

Backyard doors: External & Internal lock

Front doors: External & Internal lock

Please read:
Ch 1.2 / 1.3.1 / 2.2
Structure of Lecture 02

• The Nature of Requirements Engineering (RE)
• The Nature of Requirements
• The Process of RE
• Project Planning based on Requirements
  • Will be continued next week
Next Lecture

- Date/Time:
  - Friday, 22-Sep, 10:15-12:00

- Topic:
  - Requirements Engineering II

- For you to do:
  - Get started with homework assignment 1!
  - Go to labs next week: assessment of intermediate results for homework assignment 1
Acknowledgements

Textbooks:


Sören Lauesen: Software Requirements - Styles and Techniques, Pearson Education, 2002

Lectures on RE:

Prof. Björn Regnell, Lund University, Sweden

Prof. Steve Easterbrook, University of Toronto, Canada