Lecture 13: Summary

Last Week:
Integrated RE
method engineering
problem frames

This Week:
Summary
how is RE currently done in practice?
+ Course Summary and Evaluation

What is the state of the practice?

→ Lubars (1993) field study of 10 organizations

- Customer-specific projects:
  - usually given large monolithic statements of requirements
  - seldom in machine-readable form; are often sketchy, ill-defined;
  - concept of "superdesigner" used for interpretation, filling gaps.
- Market-specific projects
  - have smaller requirements, often produced in-house.

→ Findings

- Customer interaction always hard.
- Many projects use doc. standards, few adopt any particular method.
  - Projects using OO methods had trouble modularizing their requirements
  - some thought that structured analysis led to unintelligible specifications
- Many projects use some organizational approach to requirements validation.
  - About 1/3 of the projects did some sort of prototyping.
- Requirements evolution a major concern.

→ Conclusions

- Organizational solutions preferred over technology.
- General-purpose technology preferred over CASE.
- Requirements activities under-capitalized (only 1/3 used some tools)
- Market-driven projects increasingly important.

Managing Process Change

Source: Adapted from Humphrey, 1989, chapter 1.

→ Humphrey's principles:

- Major changes to software processes must start at the top
  - … with senior management leadership
- Ultimately everyone must be involved
- Effective change requires a goal and knowledge of the current process
  - you need a map
  - you need to know where you are on the map!
- Change is continuous
  - process improvement is not a one-shot effort
- Software process change will not be retained without conscious effort and periodic reinforcement
- Software process improvement requires investment

→ Software Engineering Process Groups (SEPGs)

- Team of people within a company responsible for process improvement
  - identifies key problems, establishes priorities, assigns resources, tracks progress, etc.
- Needs senior management support

Capability Maturity Model

Source: Adapted from Humphrey, 1989, chapter 1

Level	Characteristic	Key Challenges
5. Optimizing	Improvement fed back into process	Identify process indicators "Empower" individuals
4. Managed	(Quantitative) measured process	Automatic collection of process data Use process data to analyze and modify the process
3. Defined	(Qualitative) process defined and institutionalized	Process measurement Process analysis Quantitative Quality Plans
2. Repeatable	(Intuitive) process dependent on individuals	Establish a process group Identify a process architecture Introduce SE methods and tools
1. Initial	Ad hoc / Chaotic No cost estimation, planning, management.	Project Management Project Planning Configuration Mgmnt, Change Control Software Quality Assurance

Recent Findings

Source: "Requirements Problems in Twelve Software Companies: An Empirical Analysis" Tracy Hall, Sarah Beecham & Austen Rainer, 2001.

→ General issues

- Developers are more aware of requirements process problems than project managers and senior managers
- Most requirements problems (64%) were human/organisational in nature:
  - Poor staff retention has an impact on requirement process capability.
  - Better communication needed between developers and customers.
  - Requirements growth and change had less of an impact than expected.

→ Organisational issues

- Organisational problems are more important than process/technical issues
  - But are harder to address
  - Organisational problems amplify some process problems.
- Lack of skills exacerbates these problems

→ Maturity issues

- Higher maturity companies tend to exhibit fewer requirements problems.
  - Problems in higher maturity companies tend to be organisational
  - High maturity processes more resistant to 'damage' from organisational issues.
- Manager groups in high maturity companies understand requirement process problems better

Example Problems

Source: "Requirements Problems in Twelve Software Companies: An Empirical Analysis" Tracy Hall, Sarah Beecham & Austen Rainer, 2001.

→ Data from 12 companies

- Range of maturity levels
- Small, medium and large companies

→ Data collection

- 45 focus groups
- 200 staff
- Sept '99 to Mar '00

→ 3 types of group:

- Senior managers
- Project managers
- technical staff

Organisational requirements problems

	Frequency	Percentage
Developer communication	56	24
Inappropriate skills	47	20
Inadequate resources	33	14
Staff retention	29	13
User communication	28	12
Lack of training	19	8
Company culture	18	8
Total organisational problems	230	100

Process-based requirements problems

	Frequency	Percentage
Vague initial requirements	33	25
Undefined requirements process	32	24
Requirements growth	31	23
Complexity of application	27	20
Poor user understanding	5	4
Inadequate requirements traceability	4	3
Total process problems	132	100

Barriers to cultural change

Source: Adapted from Kauppinen et. al. 2002

Obstacles	Examples	But experience shows…
"It is not worth discovering needs directly from users"	"We've been developing such products for a long time and know users' needs"	Studies show developers are often surprised by user behaviour and expectations
	"We also use our own products and can act as users ourselves"	Developers tend to be biased by their technical expertise
	"It's a new product - therefore users cannot have any needs for it"	Still need to understand the current context and existing tasks
"It is difficult to discover needs directly from users"	"Users are unable to say what they need and want"	Combination of observation and other elicitation techniques works
	"There are so many users we cannot interview them all"	It is possible and useful to identify representative potential users
"It is risky to discover needs directly from users"	"Customers might think we don't know the basics of their business"	A well planned site visit improves the developer's image among customers
	"May spoil relations with the customer by asking stupid questions"	
"It is not worth documenting user requirements systematically"	"Customers want to see the technical specs, not user reqts"	investigating needs often reveals that a technical solution won't work
	"Documenting the requirements takes too much time"	Documented requirements save time later

RE and Extreme Programming

→ Example XP practices:
- strong customer representation
  - an on-site customer rep. always available
- relies on oral communication
  - The only documented artifacts are test-cases and code
- short planning cycle, rapid development
  - e.g. two month release cycle
  - a release split into iterations (eg 3 weeks)
- the planning game
  - customer comes up with some user stories
  - developer estimates effort for each
  - customer chooses which to include
- pair programming
  - very effective for producing quality code
- Other XP features:
  - no architectural design ('metaphor' instead)
  - test-first coding - write test cases first
  - continuous integration - integrate new code on daily basis
  - automated testing - each integration must pass all test cases
  - programmers only work a 40-hour week

→ Problems
- assumes a single customer perspective
  - single customer rep. may be biased
- no documented requirements
  - story cards capture key needs
  - but these contain very little detail…
  - …and are not usually maintained
- maintaining test cases is hard
  - test cases are a substitute for requirements…
  - …but are not traceable
- no validation (e.g. no inspections)
  - pair programming is only informal
- no documentation of changes
  - except as add/replace 'user stories'

→ Note:
- XP only suited to small projects
  - e.g. up to 12 programmers
- most XP projects only use a fraction of the XP practices

Course Summary

→ Course Goals
- Examine the state-of-the-art for research & practice in RE.
  - Role of RE in software & systems engineering
  - Techniques, notations, methods, processes & tools used in RE
- Gain practical experience in selected RE techniques
- Understand the essential nature of RE
  - Breadth of skills needed for RE, and disciplines on which it draws
  - Contextual factors & practicalities
- Gain a basic grounding for research in RE
  - Methodological issues for research
  - Current research issues & direction of the field
  - Awareness of the literature

→ Course Syllabus
- Introductory stuff
  - What is RE?
  - Why is it important?
- Foundations
  - inter-disciplinary aspects of RE
- Basic RE activities
  - Eliciting Requirements
  - Modelling & Analysing Requirements
  - Communicating Requirements
  - Agreeing Requirements
  - Evolving Requirements
- Integrated RE
  - Method Engineering
  - Patterns and Problem Frames

Feedback Questions

Did the course meet your expectations?

How useful do you think the course was to you?

What do you feel you have learned?

What did you *not* learn, that you had hoped to?

What was the best part of the course?

What was the worst part of the course?

How might the course be improved in the future?