Requirements Engineering for Embedded Systems

MOOSE Seminar Tutorial 11.06.2004
Päivi Parviainen, Maarit Tihinen,
Jukka Kääriäinen, Juha Koskela
VTT Electronics

Tutorial agenda

- 13:30 Welcome and introduction to tutorial
- 13:45 Requirements engineering introduction
  - Päivi Parviainen
- Break
- 14.30 Requirements management tools and cases
  - Jukka Kääriäinen, Juha Koskela
- Break
- 15:45 Industrial experiences of RE technologies
  - Maarit Tihinen
- 17:00 End
Background

Requirements engineering (RE) is generally accepted to be the most critical and complex process within the development of embedded systems [1, 2 & 3].
- RE has the most dominant impact on what the resulting product is capable of and is not capable of.
- RE is the process in which the most diverse set of product demands from the most diverse set of stakeholders is being considered.

According to two surveys in Europe, the most important problem of systems engineering is the need to improve the definition of system requirements [2, 4]

**Terminology**

> Requirement:
> - (1) A condition or capability needed by a user to solve a problem or achieve an objective.
> - (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
> - (3) A documented representation of a condition or capability as in (1) or (2). (IEEE Std 610.12 – 1990)

Requirements are commonly classified as (IEEE Std 830 – 1998):
- Functional: A requirement that specifies an action that a system must be able to perform, without considering physical constraints; a requirement that specifies input/output behaviour of a system.
- Non-functional: A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability.

**Requirements Development (RD)**
- Activities related to discovering, analysing, documenting and validating requirements.

**Requirements Management (RM)**
- Activities related to requirements identification, traceability and change management.
- Parallel and continuous support process for other RE processes.
- Strong connection with traditional configuration management (CM).

**Requirements Engineering (RE)**
- Activities that cover discovering, analysing, documenting, validating and maintaining a set of requirements for a system.
- Includes Requirements Development and Requirements Management.
Traditionally, RE is performed in the beginning of the system development lifecycle. However, in large and complex systems development, developing an accurate set of requirements that would remain stable throughout the development has been realised to be impossible in practice.
- RE is an incremental and iterative process, performed in parallel with other system development activities such as design, and going into more detail in each iteration.
- RE activities cover the entire system and software development lifecycle.

The process introduced in the following slides describes requirements engineering for embedded systems, where software requirements engineering is a part of the process.
Methods for Requirements Engineering

Methods used in RE are commonly not isolated, but used within a certain systems or software engineering methodology, which encompasses some other life-cycle phases as well, or combined with another approach.

Activity specific methods (described in the following slides).

General methods.
- Not phase specific methods, but can be used during several RE phases.
- Often used as part of other methods.
- For example, Brainstorming, Facilitated meeting, Interviews, Prototyping, Survey, Workshops,...

Agile methods
- Not specifically focused on RE, but they have an integral point of view, where RE is one of the activities of the whole cycle.
- For example: Adaptive Software Development (ASD), Crystal Methodologies, Extreme Programming, Scrum, .. (see Abrahamsson et al for more details)


Main RE Activities: System Requirements Development

Purpose is to examine and gather desired objectives of the system from different viewpoints (e.g., customer, users, marketing). These objectives are identified as a set of functional requirements of the system.

Activities include
- Requirements gathering/elicitation from various sources.
- Requirements analysis.
- Negotiation.
- Priorisation and agreement of raw requirements.
- System requirements documentation and validation.
### Example Methods: System RD (1)

<table>
<thead>
<tr>
<th>Activity</th>
<th>Example methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gathering requirements</td>
<td>Ethnography (Suchman, 1983)</td>
<td>Observing methods use techniques that may help to understand the thoughts and needs of the users, even when they cannot describe these needs or they do not exactly know what they want.</td>
</tr>
<tr>
<td></td>
<td>Protocol Analysis (Ericsson &amp; Simon, 1993)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Focus groups (Maguire, 1998)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JAD (Joint Application Development) (Ambler, 1998)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Volere (Robertson &amp; Robertson, 1999)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Provides a generic process for gathering requirements, ways to elicit them from users, as well as a process for verifying them.</td>
<td></td>
</tr>
<tr>
<td>Requirements analysis</td>
<td>QFD (Quality Function Deployment) (Marvelle et al., 1998.)</td>
<td>Identifying customer needs, expectations and requirements, and linking them into the company's products.</td>
</tr>
<tr>
<td></td>
<td>SCRAM (Scenario-based Requirements Engineering) (Subsite, 1998)</td>
<td>Develop requirements (whole RE) using scenarios. The scenarios are created to represent paths of possible behaviour through use cases, and these are then investigated to develop requirements.</td>
</tr>
<tr>
<td></td>
<td>SSADM (Structured System Analysis and Design Methodology) (Ashworth, 1990)</td>
<td>Can be used in the analysis and design stages of systems development. It specifies in advance the modules, stages and tasks which have to be carried out, the deliverables to be produced and the techniques used to produce those deliverables.</td>
</tr>
</tbody>
</table>

### Example Methods: System RD (2)

| Negotiation and prioritisation    | CORE (Controlled Requirements Expression) (Mullery, 1979)                      | The purpose of viewpoint oriented methods is to produce or analyse requirements from multiple viewpoints. They can be used while resolving conflicts or documenting system and software requirements. |
|                                   | WinWin approach (Bose, 1995)                                                  |                                                                            |
| System requirements documentation | IEEE Std 1233-1998                                                             | Standards define the contents of a SyRS.                                    |
|                                   | VDM (Vienna Development Model) (Böhm & Jones, 1978)                           | In formal methods requirements are written in a statement language or in a formal / mathematical way. |
|                                   | Specification language Z (Sheppard, 1995)                                     |                                                                            |
|                                   | HPM (Hatley-Pirbhai Methodology) (Hatley & Pirbhai, 1987)                     | Helps to identify and prioritise requirements and also can be utilised when documenting system and software requirements. |
|                                   | VORD (Viewpoint-Oriented Requirements Definition) (Kotonya & Sommerville, 1996)|                                                                            |
Main RE Activities: Requirements Allocation & Flow-Down

Purpose is to make sure that all system requirements are fulfilled by a subsystem somewhere or by a set of subsystems collaborating together.

Activities include:
- Allocation: architectural work carried out in order to design the structure of the system and to issue the top-level system requirements to subsystems.
- Flow-down: writing requirements for the lower level elements in response to the allocation.
- Requirements identification and traceability have to be taken into account.

Allocation and flow-down may be done for several hierarchy levels.

Requirements and architecture work are closely linked, both disciplines are needed to achieve the desired result, a system that meets its users' needs.

Example Methods: Allocation and Flow-Down

<table>
<thead>
<tr>
<th>Activity</th>
<th>Example methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation</td>
<td>SRA (System Requirements Allocation Methodology) (Hadel &amp; Lakey, 1995)</td>
<td>A customer-oriented systems engineering approach for allocating top-level quantitative system requirements. It aims at creating optimised design alternatives, which correspond to the customer requirements using measurable parameters.</td>
</tr>
<tr>
<td></td>
<td>HPM (Hatley-Pirbhai Methodology) (Hatley &amp; Pirbhai, 1987)</td>
<td>Verifies requirements allocation and manages changes during allocation phase.</td>
</tr>
<tr>
<td></td>
<td>QADA (Matinlaski &amp; Niemelä, 2002)</td>
<td>Methods for architecture design and analysis. See more from (Dobrica &amp; Niemelä, 2002).</td>
</tr>
<tr>
<td></td>
<td>FAST (Weiss et al., 1999)</td>
<td>Methods for architecture design and analysis.</td>
</tr>
</tbody>
</table>
Main RE Activities: SW Requirements Development

- Activity of determining what functions the software will perform and documenting those functions and other requirements in a software requirements specification.
- Enables the specification of software functions and performance, indication of software’s interface with other system elements, and establishment of design constraints that the software must meet.
- Refines the software allocation and builds models of the process, data, and behavioral domains that will be treated by software.
- Activities include:
  - Analysing, modelling and validating both the functional and quality aspects of a software system.
  - Defining, documenting and validating the contents of SW subsystems.

Example methods: SW RD (1/2)

<table>
<thead>
<tr>
<th>Activity</th>
<th>Example methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software requirements analysis</td>
<td>OMT (Object Modeling Technique) (Bourbeau &amp; Cheng, 1995)</td>
<td>Object oriented methods model systems in an object oriented way or support object-oriented development in the analysis and design phases.</td>
</tr>
<tr>
<td></td>
<td>UML (Unified Modeling Language) (Booch et al., 1999)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SADT (Structured Analysis and Design Technique) (Schuman &amp; Ross, 1977) SASS (Structured Analysis and System Specification) (Davis, 1990).</td>
<td>Structured methods consist of sequential steps or structured work practices for analysing software requirements.</td>
</tr>
<tr>
<td></td>
<td>B-methods (Schneider, 2001)</td>
<td>Formal methods are often used for safety-critical systems.</td>
</tr>
<tr>
<td></td>
<td>XP (Extreme Programming) (Beck, 1999).</td>
<td>Agile methods are not specially focused on RE, but they have an integral point of view, where RE is one of the activities of the whole cycle. See more from (Abrahamsson et al., 2000).</td>
</tr>
<tr>
<td></td>
<td>CARE (COTS-Aware Requirements Engineering) (Chung et al., 2001) OTSO (Off-the-Shelf Option) (Kontio, 1999).</td>
<td>Specific methods for RE when using COTS (Commercial off-the-shelf). The COTS is a ready-made software product, which is supplied by a vendor and has specific functionality as part of a system.</td>
</tr>
<tr>
<td></td>
<td>PLanguage (Gilb, 2003)</td>
<td>Consists of a systems engineering language for communicating systems engineering and management specifications, and a set of methods providing advice on best practices.</td>
</tr>
</tbody>
</table>
Example Methods: SW RD (2/2)

<table>
<thead>
<tr>
<th>Requirements documentation</th>
<th>IEEE Std 830-1998</th>
<th>IEEE defines contents of an SRS. (The standard doesn't describe sequential steps to be followed, but defines the characteristics of a good SRS and provides a structure template for the SRS. This template can be used in documenting the requirements, and also as a checklist in other phases of the requirements engineering process.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements validation</td>
<td>Volere (Robinson &amp; Robertson, 1999)</td>
<td>Provides process for gathering, eliciting and validating both system and software requirements. Sequences of computer-generated displays, called storyboards, are used to simulate the functions of the formally implemented system beforehand. This supports the communication of system functions to the user, and makes the trade-offs non-functional and functional requirements visible, traceable and analysable. Also several other methods such as object oriented methods provide some support for validation and verification.</td>
</tr>
</tbody>
</table>

RE Process Main Activities

- Continuous Activities
  - Requirements Documentation
    - Standards define contents of requirements documents.
  - Requirements Validation & Verification
    - Validation: actions that try to confirm that the behaviour of a developed system meets user needs.
    - Verification: actions that try to confirm that the product of a system development process meets its technical specifications.
    - Examples of available techniques:
      - Requirements reviews with the stakeholders,
      - Prototyping.
  - Requirements Management
    - Described in detail in the following slides.
RM Activities

Requirements management

- Requirements management planning

- Identification
- Traceability
- Change management

- Configuration management (CM) -based information management can be utilised during RM (item IDs, attributes, versions, baselines, change objects, linking objects, prevents simultaneous updates, ...)

Requirements identification

- Requirements identification focuses on the assignment of unique identifier for each requirement according to project's identification schemes.
- Includes also identification and management of requirement versions.
- Requirement attributes can be used to record additional information about requirements. By using attributes better management of complexity of information is achieved.
- Example techniques:
  - Identification schemes: Unique identification of each requirement and versions, e.g., numbering schemes, structure based identification, symbolic identification.
  - Dynamic renumbering.
  - Database record identification.
  - Baselining: Technique to identify clear set of items, which serve as basis for further development.
Requirements Traceability

- Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement in both a forwards and backwards direction.
- Traceability between requirements and other information (design, implementation, rationale,...).
- Requirements should be traceable between specification levels to ensure that all requirements are flowed from the top, through all requirements levels.
- Requirements traceability is a critical part of requirements change management.

Requirements Change Management

- Requirements change management (RChM) refers to the ability to manage changes to system requirements throughout the development lifecycle.
- Change management, in general, includes identifying, analysing, deciding on whether change will be implemented, implementing and validating change request.
- Requirements baselines form basis for RChM.
- RChM ensures that similar information is collected for each proposed change and that overall judgements are made about the costs and benefits of proposed change.
Example Methods: Requirements Management

<table>
<thead>
<tr>
<th>Activity</th>
<th>Example methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements traceability</td>
<td>Cross reference, Traceability matrices, Automated traceability links. (Sommerville &amp; Sawyer, 1997)</td>
<td>Techniques can be used for presenting and managing requirements as separate entities, and describing and maintaining links between them e.g. during allocation, implementation or verification.</td>
</tr>
<tr>
<td>Change management</td>
<td>Olsen’s ChM model (Olsen, 1993) V-like model (Harjani &amp; Quelle, 1992) Ince’s ChM model (Ince, 1994)</td>
<td>Traceability can be supported by using languages or notations.</td>
</tr>
</tbody>
</table>

RE/RM Tool Support

> RE support tools can be divided into following two types:
  - Modelling and validation tools support the development of system models. The completeness and consistency of the models can be checked.
  - Management tools help to manage the requirements database and support the requirements change management.

> Tool lists available in the Internet
  - Volere pages: [http://www.volere.co.uk/tools.htm](http://www.volere.co.uk/tools.htm)
Method and Tool Selection

- **Characteristics of a product development project:**
  - The size of the system being developed (total number of requirements).
  - Number of developers involved in the project.

- **Organisational issues:**
  - Number of stakeholders involved in the development process.
  - Policies and maturity of the organisation's existing requirements engineering process.
  - The tools that are already in use in the organisation, and new tools that are compatible with the existing systems.
  - Tool and training budget.
  - Working environment type - distributed or centralised.
Method and Tool Selection

> Characteristics of the product:
  - Safety-criticality of the product.
  - Need for real-time operation (real-time requirements and constraints present).
  - Software-hardware ratio.

> Other
  - Maturity of the method/tool and availability of training and support.
  - Understandability and usability of the method/tool.
  - Stability of the tool providing companies (that also means the availability of the tools).

Conclusions

> Requirements Engineering is a complex process that considers product demands from many viewpoints, roles, responsibilities and objectives.
> A large set of supporting methods and tools is available for Requirements Engineering, each with their specific strengths and weaknesses.
> The methods' feasibility and applicability varies between phases or activities.
> Method descriptions often lack the information of the methods' suitability to different environments and problem situations,
  - The selection of an applicable method or combination of methods to be used in a particular real-life situation is complicated.
  - MOOSE project aims to provide experiences and guidelines for usage of the methods in practice.
> This presentation has only listed a few examples of methods. For a more comprehensive listing of methods see (Parviainen et al., 2003).

Requirements Management / Engineering Tools and Cases

June 11, 2004
Jukka.Kaariainen@vtt.fi
Juha.Koskela@vtt.fi

Contents

> RE/RM tools
> RM adaptation CASES/Examples
  - RM adapted into XP context in VTT
  - RM tool / Eclipse demonstration
  - RM tool support enhancement in Solid

Copyright © 2004
MOOSE Consortium
Finland
RE/RM tools

Classification of RM/RE tools

  - Modelling & validation tools
  - Management tools
  - General-purpose tools
  - Special-purpose tools
  - Workbenches, environments
RM tools: basic functionality

- Maintain unique identifiable description of all requirements.
- Classify requirements into logical user-defined groups.
- Specify requirements with textual, graphical, and model based description.
- Define traceable associations between requirements.
- Maintain an audit trail of changes, archive baseline versions, and engage a mechanism to authenticate and approve change requests.
- Support secure, concurrent co-operative work between members of a multidisciplinary development team.
- Support standard systems modelling techniques and notations.
- Generate predefined and ad hoc reports.
- Generate documents that comply with standard industrial templates.
- Connect seamlessly with other tools and systems.

RM tools: lists of tools in web

- Rabi Archrafi’s descriptions of available requirements engineering tools. URL: http://www.volere.co.uk/tools.htm
- INCOSE’s pages, which contain taxonomy of requirements engineering tools divided into hierarchical categories according to the purpose of use. URL: http://www.incose.org/tools/tooltax/reqengr_tools.html
- Ian Alexander’s pages, which present a list of RE tool vendors and freeware suppliers. URL: http://easyweb.easynet.co.uk/~iany/other/vendors.htm
RM tools: shortcomings based on literature and MOOSE industrial partners (1/2)

Tool requirements which current requirement management tools cover only partly:
- Adaptation to the situation
- Lack of support for collaborative work
- Usability for non-technical users
- Configuration management is difficult
- Interoperability of tools is insufficient

RM tools: shortcomings based on literature and MOOSE industrial partners (2/2)

Shortcomings analysis of Rational’s Requisite Pro tool (e.g.)
- Has a closed database, and thus limited reporting facilities
- Lacks connection with configuration management (double registration of requirements). Baselining is not really supported
- Process in larger teams becomes procedural
- Soda doesn’t make it possible to use all requisite queries
Some aspects for tool selection
(collected from MOOSE documentation (industrial partners and literature))

Tool selection in an organisation (e.g.)
- Tools already in use
- Tool and training budget
- Size of the system being specified
- Specific guidelines that are implemented
- Stability of the companies supplying the tool
- Technical aspects
  - RM functionality
  - Interfaces with legacy systems
  - Configuration management support

RM cases
RM cases

- Requirements management for extreme programming (XP) in VTT
  - Survey of RM/RE in XP context
  - Adaptation of RM into XP
  - Building RM tool support for XP

- Pre-requirements management tool support for Solid
  - Enhancing tool support for pre-requirements management

Case: Requirements management in XP
Extreme Programming (XP)

- «Agile» software development methodology
- Has emerged in the late 90’s
- Also other agile methods such as DSDM (Dynamic System Development Method ), RUP,...
- XP tries to simplify management tasks and documentation while the traditional software engineering places more emphasis on strict control and extensive documentation
- Four basic values (communication, simplicity, feedback, courage)
- 12 practices (e.g. planning game, refactoring, pair programming, open workspace, ...)
- Suits for small projects

User story = user’s conception about the system
Task = technical description about the system

Releases in short cycle
A release is divided into iterations

The basic process starts with user story -planning containing the definition, analysis and the selection of stories for next release
Then tasks are described and planned for next iteration.
Then stories and tasks are implemented and tested
RM in traditional XP

- A user story can be thought as a high level requirement or a user requirement.
- A task is a more fine-grained presentation about a story (story is split into few tasks).
- **Requirements ID**: Stories and tasks are presented in paper cards ("requirements items") including relevant attributes (ID, date, status,...)
- **Requirements traceability**: Cards can be placed onto the wall (stories and their respective tasks) => simple and powerful way of visualising traceability between stories and tasks + baselining
- **Requirements ChM**: ChM usually formal and bureaucratic => problematic in XP => "lightweight" practices
- Existing tools for working with user stories (e.g.): MILOS, http://userstories.com/links.php

---

**Story card**

- ID attribute
- Other attributes

---

Copyright © 2004 MOOSE Consortium

Paivi.Parviainen@vtt.fi

11.06.2004
Stories and their respective tasks (RT)

Needs for electronic user story and task management

- Findings related to user story management:
  - Basic abilities for defining project-specific user story and task attributes as well as abilities for storing, relating and retrieving user stories and tasks
  - Second, the tool should allow the XP process to remain agile. This means that the tool should not jeopardize XP's intentions for open communication and lightweight management and documentation
  - Third, requirements management tool support should be integrated into the project's overall development environment. This allows the project team to operate via one channel from a user story definition, through implementation, up to testing
RM for XP

Planning

- Identification
  - Items: Story and Task cards
  - Pre-defined attributes for items + user-defined attributes
  - Baselining facility (through release / iteration attribute)
- Traceability:
  - Relationships:
    - Story <--> Task
- Change Management: History information
- Integration with existing systems

The proposed solution has been integrated into Eclipse – environment (http://www.eclipse.org)

- Used Eclipse configuration contains:
  - StoryManager
  - JDT (Java Development Tooling)
  - CVS (Concurrent Versions System)
  - JUnit (Regression testing framework)

This framework complemented with StoryManager extension enables integrated tool environment for XP-based development from stories through implementation up to testing

- All project data is managed under relevant Eclipse project (CVS project database and StoryManager project database are connected together)
StoryManager in Eclipse

- MySQL
- CVS = Concurrent Versions System
- JDT = Java Development Tooling
- JUnit = Regression testing framework

StoryManager functionalities

- Project set-up
  - Define attributes for story/task cards => adaptation into project context
- Planning game / release planning
  - Define stories (ID, attributes)
  - Baseline stories using release/iteration attributes (BL)
- Planning game / iteration planning
  - Define tasks (ID, attributes)
  - Relate with stories (RT)
  - Baseline tasks using iteration attribute (BL)
- Process phase independent facilities
  - Browsing
  - Reporting
  - Check in/out
Definition of requirements presented in:
Kolehmainen, K; Agile requirements engineering: building tool support for XP,

First version presented in:
Supporting requirements engineering in extreme programming: managing user stories;
In Proceedings of the ICSSEA 2003,

First experiment ready and results analysed and will be presented in:
Results of analysis will be published in Euromicro conference: Special session on "Advancements in Agile Software Development" Rennes, France, 31 August – 3 September, 2004

Second tool development round is planned to be presented in ?????:

Case: Enhancing pre-requirements engineering/management tool support in Solid
Background

- VTT as sparring partner
  - VTT's main contribution related to tool definition and process connection
  - Practical implementation by Solid's system administrators
- RM concepts
  - ID
  - RT
  - RChM
  - Documentation
- Lotus Notes was selected after tool consideration for RE tool (existing system)

Case description

- Common understanding about pre-requirements definition (common understanding about attributes that will be collected)
- Requirements ID was carried out using AutoID functionality of the system
- Status attribute definitions (describes maturity of req. during SW development)
- Requirements traceability (e.g. rationale, views (releases))
- Definition of views to the collection of requirements in order to support grouping of requirements during SW development (release content, implemented, ...)

Copyright © 2004
MOOSE Consortium
How the work was done

> Common understanding about tool support and used attributes (tool specification document)

> Strong co-operation with process definition (tool workflow as part of process description) => tool should support and be consistent with process (terminology, etc...)

> Iterative development
Content

> “How widely RE technologies are used in industry?”
  - overview based on literature studies
>
> Overview of the used RE methods and tools based on
  - the HIRVI survey (19 Finnish companies)
  - the MOOSE web repository (randon sample)
>
> Industrial experiences of MOOSE cases
>
> “Risks and problems related to the RE process and technologies”

“How widely RE technologies are used in industry?”
The survey included a list of known techniques; participants could select all that applied:

- Several general methods were used for requirements elicitation
- The most popular requirements elicitation approaches were group-consensus-type techniques such as User-centered design, Joint Application Design, Interviews and Focus groups.
- Over 50% surveyed used scenarios or Use Cases (while only 30% used object-oriented analysis for modeling, see figure 2)

- 33% indicated that they used no methodology at all for requirements analysis and modeling
- Over 50% surveyed has used scenarios or use cases (see figure 1), while only 30% had used object-oriented analysis for modeling
- Structured approaches were selected by 7-14% of respondents
Twelve SW companies were interviewed. The survey describes that a lot of literature is available on solid RE practices, while the survey indicates poor customs, for example:

- No company had requirements management (RM) tools in use, figure 3:
- only two companies used requirements templates in a standard manner, including, e.g., identification, description, rationale behind decision.

The survey points out that companies do not know how to start practical RE process improvement efforts.

---

The article contains interview results of seven companies and one research institute (MOOSE consortium's industrial partners).

- methods, tools, and techniques used were rather general SW engineering technologies:
  - e.g., memory, power or real-time requirements weren't so a vital role because general technologies didn't have special features for dealing these requirements.
- requirements specification: companies usually specified requirements in natural language and processed them with an ordinary word processor:
  - templates and guidelines were used, but not in all projects...
- UML was not common practice yet, but most companies were at least considering its possibilities for applying UML to RE:
  - use cases were the most used UML constructs.
- Formal specifications were rarely used (only one used notation Z).
"Overview of the Used Methods and Tools in Requirements Definition and Configuration Management"

Results of HIRVI survey:

- Methods and tools used in Requirements Definition

<table>
<thead>
<tr>
<th>PROJECT SIZE</th>
<th>METHODS USED in Reqs Definition</th>
<th>TOOLS USED in Reqs Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Man-Years</td>
<td># of People # of Answers</td>
<td>Name</td>
</tr>
<tr>
<td>&lt; 1</td>
<td>1-5     1 1</td>
<td>1</td>
</tr>
<tr>
<td>1-5</td>
<td>6-10    3 3</td>
<td>1</td>
</tr>
<tr>
<td>6-10</td>
<td>11-20   1</td>
<td>1</td>
</tr>
<tr>
<td>10-50</td>
<td>1-5     1 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>11-20   1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 20    1</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>1-5     2 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>19</td>
</tr>
</tbody>
</table>
## Random sample from WEB repository

- Methods and tools used in Requirements Definition

<table>
<thead>
<tr>
<th>PROJECT SIZE</th>
<th>METHODS USED in Reqs Definition</th>
<th>TOOLS USED in Reqs Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Proprietary</td>
</tr>
<tr>
<td>&lt; 1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1-5</td>
<td>1-5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6-10</td>
<td>5</td>
</tr>
<tr>
<td>6-10</td>
<td>6-10</td>
<td>2</td>
</tr>
<tr>
<td>11-20</td>
<td>11-20</td>
<td>3</td>
</tr>
<tr>
<td>10-50</td>
<td>1-5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6-10</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>11-20</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>&gt; 20</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>30</td>
</tr>
</tbody>
</table>

## Results of HIRVI survey

- Tools used in CM and PR/CR handling

<table>
<thead>
<tr>
<th>PROJECT SIZE</th>
<th>TOOLS USED in configuration management</th>
<th>TOOLS USED in PR/CR handling (Problem reports /change requests)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Tools used (named)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; 1</td>
<td></td>
<td>[CVS], [PVCS], [ClearCase], [DocLib], [Proprietary]</td>
</tr>
<tr>
<td>1-5</td>
<td>1-5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>6-10</td>
<td>3</td>
</tr>
<tr>
<td>6-10</td>
<td>11-20</td>
<td>1</td>
</tr>
<tr>
<td>10-50</td>
<td>1-5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>11-20</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 20</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>1-5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>19</td>
</tr>
</tbody>
</table>
### Randon sample from WEB repository

- Tools used in CM and PR/CR handling

<table>
<thead>
<tr>
<th>PROJECT SIZE</th>
<th>TOOLS USED in configuration management</th>
<th>TOOLS USED in PR/CR handling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>1-5</td>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
</tr>
<tr>
<td>6-10</td>
<td>5 (ClearCase), 1 (VisualSourceSafe), 1 (CM Synergy + PVCS)</td>
<td>3 (ClearCase), 1 (ChangeSynergy)</td>
</tr>
<tr>
<td>6-10</td>
<td>9-20</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
</tr>
<tr>
<td>6-10</td>
<td>9-20</td>
<td>3 (ChangeSynergy) + 2 (VisualSourceSafe)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (ClearCase)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
<tr>
<td>1-5</td>
<td>4 (ClearCase), 1 (VisualSourceSafe)</td>
<td>3 (ChangeSynergy) + 2 (PVCS)</td>
</tr>
</tbody>
</table>
### MOOSE Case11 and Experiences:

**Early phases of RE**

- The objective of the case was to examine and improve RE process. The following steps were taken:
  - current state analysis of RE process
  - RE process re-definition
  - selection of RE technologies
  - piloting new RE process
  - final deployment of the new RE process (including training)

**Experiences of the technologies:**

- an SME should be careful in tool selection
- there were surprisingly few tools that integrate well to existing tool set(s)
- if something is working do not scrap it but enhance it!

**Technologies tried:**

- current state analysis and RE process re-definition was made using Primer method
- inventory results were reviewed
- existing tools were examined

- The result was that a new tool was not employed but existing TPR system was decided to enhance.

**Case experiences:**

- It is difficult to combine Moose case work with regular work activities ... and organizational changes do not help either.
  - Problems: organizational changes and limited resources (despite commitment)
  - Success: rethinking the RE process, reasoned choice of technology, and the sparring partner system during the case was good and helpful

- "In the end we got useful results"

### MOOSE Case7 and Experiences:

**RE in multi-disciplinary environment**

- The objective of the case was to improve and experiment with the requirements engineering process in a multi-disciplinary environment (inc. a consistent set of system, architecture, sub-system requirements). The following steps were taken:
  - Analysis of existing practices, definition and trying new practices
  - Evaluate /update (TBDL)

**Experiences of the technologies:**

- Spiral lifecycle seems to work
- Requirements template works!
- DOORS works very well
- Strengths: Traceability, customisation;
- Weaknesses: User interface

**Technologies tried:**

- Spiral model, Volere requirements template and DOORS method, e.g.
  - Tailoring the Spiral model
  - Tailoring the Volere requirements template
  - Document structure in DOORS
  - Requirements template in DOORS
  - including trainings and workshops

**Case experiences:**

- A real project as carrier has severe disadvantages:
  - competing priorities: project priorities always win!
  - coupling in time and uncertainty in resources

- Advantage real project: possible real success story instead of succeeded experiment

- Getting commitment for executing a case-study may take some time (several months).
**MOOSE Case14 and Experiences (1/2):**

<table>
<thead>
<tr>
<th>Requirements description, design definition and metrics gathering:</th>
<th>Technologies tried:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The objectives of the case were:</td>
<td></td>
</tr>
<tr>
<td>1) establish requirements definition and documentation method (e.g., requirements definition template developed)</td>
<td></td>
</tr>
<tr>
<td>2) improve requirements reusability between similar projects and for future version of a system, requirements based validation with extraction of Test Cases (e.g., test case specification template generated)</td>
<td></td>
</tr>
<tr>
<td>3) traceability of requirements along the development and validation,</td>
<td></td>
</tr>
<tr>
<td>4) measure the effect of the systematic requirements engineering on the product development cycle (6 effort metrics and one economic metric were measured).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>First approach to a formal methodology</td>
</tr>
<tr>
<td></td>
<td>- Developers involved found both the requirements and the test case specification easier to translate to code or to perform the tests</td>
</tr>
<tr>
<td></td>
<td>- Overall time dedicated to the specification and first stages of development in the project greater than usually due to the deficiency of practice with the new methodology</td>
</tr>
<tr>
<td></td>
<td>- In the test cases definition stage, some lacks in the requirements definition were found</td>
</tr>
<tr>
<td></td>
<td>- Some requirements were not sufficiently clear enough to describe the test case and were redefined</td>
</tr>
<tr>
<td></td>
<td>- Starting point of a requirements repository to improve the reuse</td>
</tr>
<tr>
<td></td>
<td>- First stage: simple database</td>
</tr>
<tr>
<td></td>
<td>- Later: the possibility of developing a database tool will be studied</td>
</tr>
</tbody>
</table>

**MOOSE Case14 and Experiences (2/2):**

<table>
<thead>
<tr>
<th>Experiences of the technologies:</th>
<th>Case experiences:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems:</td>
<td>Real benefits will be noticed better in subsequent projects:</td>
</tr>
<tr>
<td>(-) Starting from scratch has been a difficult task because we didn’t have any model of reference</td>
<td></td>
</tr>
<tr>
<td>(-) More reliable measurements will be obtained when we have more projects to compare with</td>
<td></td>
</tr>
<tr>
<td>Success:</td>
<td>More reliable measurements with more projects to compare with</td>
</tr>
<tr>
<td>(+) Gain experience in embedded software methodologies</td>
<td></td>
</tr>
<tr>
<td>(+) Close collaboration with sparring partner</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2004 MOOSE Consortium 11.06.2004

Paivi.Parviainen@vtt.fi
### MOOSE Case15 and Experiences (1/2):

#### Case: Requirements capturing:
- **Objectives of the case:**
  1. Improvement of requirements engineering methods (detecting when insufficient knowledge of the application and needs could result in design errors).
  2. Improvement in formal specification methods for design.
  3. Apply metrics for evaluation, and introduction and evaluation of automatic testing methods.
- **Steps taken:**
  - Current state analysis of RE process (identification of strong and weak points in current practice).
  - Evaluation of different options for RE.
  - Definition and introduction of new RE methods.
  - Evaluation of new methods (metrics).

#### Technologies tried:
- Rose tools and MagicDraw UML5.0
  - MagicDraw is cheap compared to Rose, but lacks of direct integration, non-intuitive, complex programming language.
  - A decision was to continue using UML modelling of requirements and design (Rose).
- Incremental model and TestWorkflow.

### MOOSE Case15 and Experiences (2/2):

#### Experiences of the technologies:
- Complementary views of the requirements and more complete requirement lists have been recorded. Inconsistencies between different requirements have been detected and the need of a trade-off between opposite requirements have been recognized.
- Relevant customers have been incorporated to the requirement gathering process. The innovative aspect to the company have been to involve the users not only in the requirements definition but also in the requirements review process.
- One of the most interesting experiences with Moose have been to incorporate the methodology to our work in standardization bodies.
- Requirement description using formal/non-formal notation: this have been an evolution from the position of the company at the beginning of the project. Formal notation as UML has been finally defined as the base for every development in the company, from the requirement engineering phase to the design stage.
- Requirement validation and QC: this have been one of the most important results of the project. Different participants from inside but also external to the company have been invited to participate in the validation process. Using formal notation has help to communicate to external users.

#### Case experiences:
- **Success or problems:**
  - (+) Help from sparring partner has been very positive to the project.
  - (-) Very limited resources (SME).
  - (-) "Time to market" as a major constraint.
  - The full product cycle of the experiment is surpassing the time constraints and resources of the project.
  - RE result was useful: decreasing development cost, maintenance and time efficiency.
  - Formal notation increases communication effectiveness (internal and external).
  - Involvement of users in RE and in validation increases quality.
  - Collaborative work is very useful, cost is optimised by sharing effort.
  - Training is needed.

Copyright © 2004 MOOSE Consortium

Paivi.Parviainen@vtt.fi

11.06.2004
Summary of the Moose Cases:

> All cases were carried out in real projects
> Cases implemented by using "step by step" -approach: typically current state analysis was the first step
> Activities /technologies tried cover often rather simply and clear wholeness, e.g.,
>   - process definition concerning early phase activities
>   - creating and modifying templates for RE identification, specification
>   - reviewing and tailoring existing tools and methods
>   - knowledge sharing within the case
> Re-thinking own RE-process, methods or tools was useful experience!
> The case was realized to be as a start for further development activities!

“Risks and problems related to The RE process and Technologies or adopting them”
Top Risks of Requirements Engineering:
Lawrence Brian, Wiegers Karl, Ebert Christof, IEEE Software 2001

- Overlooking a crucial requirement:
  - missing a critical functional or attribute requirement: this situation leads to huge damages, for example, if it will be realized late in the development process (e.g., in the implement phase)
- Inadequate customer representation:
  - one of the central activities in RE is to achieve agreement on requirements: you must find out what your customer really needs!
- Modelling only functional requirements
  - functional requirements are the most obvious ones to the user, so most elicitation discussions focus on them. However, it’s also important to gain agreement on the quality attribute requirements like reliability, performance, security, ease of use etc.
- Not inspecting requirements
  - inspecting your requirements is the most effective way to identify ambiguities, unstated assumptions, conflicting requirements, and other defects at the earliest possible point.
  - choose inspection teams with a broad constituency, including testers.
- Attempt to get perfect requirements before beginning construction
  - it’s safer to assume that your requirements are going to change than they won’t
- Representing requirements in the form of designs
  - you can choose a particular solution that might not be the best one to implement

Problems related to RE

Typical problems are:
- Misunderstood requirements:
  - problems arise from insufficient communication, poor quality of requirements documents, and requirements engineering process itself (e.g., insufficient requirements elicitation or negotiation activities)
- Changing requirements:
  - RE is a multidisciplinary process and changes should be analyzed and tracked for all stakeholders
  - relationships between requirements may cause problems (requirements traceability, prioritization and rationale behind decisions)
- Missing requirements:
  - these problems can be prevented only by careful planning and improving of the requirements engineering process

→ Can some method give support for avoiding risks and problems?
  - Does requirements analysis depend ONLY on practitioner’s experience...?
Workshop issues:

"Why industry doesn’t use many of the available technologies?"

"Own experiences related to the RE process development work"

Experiences in adopting RE methods, practices or tools:

> Literature study shows [*] that there are several methods and tools available for RE. However, they have different focus areas or purposes of use (e.g., methods support different activities during the RE process)

- In addition, case reports and experiment based articles have been written from different viewpoints:
  - a project may not have been carried out in a real-life (~experimental environments)
  - participants may have a lot of experiences of use of the applied methods or tools  this may distort the truth!
  - they don’t inform why or how a method (/used tools) was selected (e.g., own background or previous research activities may have a selection criteria, even before industrial needs!)

> Each case report should be examined deeply, so that a real knowledge of the case can be achieved!
  - Short descriptions of few experience articles will be delivered during the Tutorial.

More information available:

- MOOSE project web pages:  
  http://www.mooseproject.org/

- MOOSE web repository:  
  http://moose.outdare.net/

Thanks for your attention!