Topics Today

• Software Product Management Overview
• Requirements Elicitation, Specification & Management
• Release Planning
• Tool Demo: ReleasePlanner 2.0
• Homework: Assignment 3
Software Product Management (SPM)

Overview
What is Software Product Management?
The Software Product Manager’s Reality

- Market
  - Market trends
- Development
  - Scope changes
  - Contracts, Partner requests
- Board
  - Strategy
- Sales
  - Feature requests
- R&D
  - Technologies
- Customers
  - Bugs, Feature requests

MTAT.03.244 / © Dietmar Pfahl 2016
The SPM Framework

<table>
<thead>
<tr>
<th>Strategic Management</th>
<th>Product Strategy</th>
<th>Product Planning</th>
<th>Development</th>
<th>Marketing</th>
<th>Sales and Distribution</th>
<th>Service and Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate Strategy</td>
<td>Positioning and Product Definition</td>
<td>Product Life-Cycle Management</td>
<td>Engineering Management</td>
<td>Marketing Planning</td>
<td>Sales Planning</td>
<td>Service Planning and Preparation</td>
</tr>
<tr>
<td>Portfolio Management</td>
<td>Delivery model and Service Strategy</td>
<td>Roadmapping</td>
<td>Project Management</td>
<td>Customer Analysis</td>
<td>Channel Preparation</td>
<td>Service Provisioning</td>
</tr>
<tr>
<td>Market Analysis</td>
<td>Pricing</td>
<td></td>
<td>Quality Management</td>
<td>Product Launches</td>
<td>Operational Distribution</td>
<td>Sales Support</td>
</tr>
<tr>
<td>Product Analysis</td>
<td>Ecosystem Management</td>
<td></td>
<td></td>
<td></td>
<td>Operational Marketing</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Participation</th>
<th>Core SPM</th>
<th>Orchestration</th>
</tr>
</thead>
</table>

Source: ISPMA – International Software Product Management Association
# The SPM Framework

<table>
<thead>
<tr>
<th>Strategic Management</th>
<th>Product Strategy</th>
<th>Product Planning</th>
<th>Development</th>
<th>Marketing</th>
<th>Sales and Distribution</th>
<th>Service and Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate Strategy</td>
<td>Positioning and Product Definition</td>
<td>Product Life Cycle Management</td>
<td>Engineering Management</td>
<td>Marketing Planning</td>
<td>Sales Planning</td>
<td>Service Planning and Preparation</td>
</tr>
<tr>
<td>Portfolio Management</td>
<td>Delivery model and Service Strategy</td>
<td>Roadmapping</td>
<td>Project Management</td>
<td>Customer Analysis</td>
<td>Channel Preparation</td>
<td>Service Provisioning</td>
</tr>
<tr>
<td>Market Analysis</td>
<td>Pricing</td>
<td>Quality Management</td>
<td>Product Launches</td>
<td>Operational Distribution</td>
<td>Sales Support</td>
<td></td>
</tr>
<tr>
<td>Product Analysis</td>
<td>Ecosystem Management</td>
<td></td>
<td></td>
<td></td>
<td>Operational Marketing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Legal and IPR Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Performance and Risk Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Participation</td>
<td>Core SPM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orchestration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The SPM Competence Model

Source: Willem Bekkers, University of Utrecht
Requirements
Elicitation, Specification & Management
What is SW Requirements Engineering?
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
Definition: Requirements Engineering

Requirements Engineering (RE) is a set of activities concerned with identifying and communicating
1. the purpose of a software-intensive system, and
2. the contexts in which it will be used.

Hence,
RE acts as the bridge between
1. the real world needs of users, customers, and other constituencies affected by a software system, and
2. the capabilities and opportunities afforded by software-intensive technologies
**Definition: Requirements Engineering**

Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies.

- **Not a monolithic phase or stage!**
- **Communication is as important as the analysis**
- **Quality means fitness-for-purpose. Cannot say anything about quality unless you understand the purpose**
- **Designers need to know how and where the system will be used**
- **Requirements are partly about what is needed...**
- **...and partly about what is possible**
- **Need to identify all the stakeholders - not just the customer and user**
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

Requirements traced back to their source

+ Requirements Management

Elicit
Analyze
classify, organize, prioritize, negotiate

Specific
Validate

Model

MTAT.03.244 / © Dietmar Pfahl 2016
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
Where/How to 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”

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…

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 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…

Application Domain ← System Domain
RE questions to answer …

• How to Elicit Requirements?
  • Different Project Settings
  • Complex Customer-Supplier Relationships / Value-Chains
  • Many (Types of) Stakeholders
• How to Describe/Specify Requirements?
  • Different Levels & Types of Requirements
  • Quality of Requirements?
  • Different Representation Styles
• How to Prioritize/Assign/Manage Requirements?
RE questions to answer …

- How to Elicit Requirements?
  - Different Project Settings
  - Complex Customer-Supplier Relationships / Value-Chains
  - Many (Types of) Stakeholders

- How to Describe/Specify Requirements?
  - Different Levels & Types of Requirements
  - Quality of Requirements?
  - Different Representation Styles

- How to Prioritize/Assign/Manage Requirements?
Requirements Elicitation
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

Traditional techniques
- Introspection
- Reading existing documents
- Analyzing hard data
- Interviews
  - Open-ended
  - Structured
- Surveys / Questionnaires
- Meetings

Collaborative techniques
- Focus Groups
  - Brainstorming
  - JAD/RAD workshops
- Prototyping
- Participatory Design

Contextual (social) approaches
- Ethnographic techniques
  - Participant Observation
  - Ethnomethodology
- Discourse Analysis
  - Conversation Analysis
  - Speech Act Analysis
- Sociotechnical Methods
  - Soft Systems Analysis

Cognitive techniques
- Task analysis
- Protocol analysis
- Knowledge Acquisition Techniques
  - Card Sorting
  - Laddering
  - Repertory Grids
  - Proximity Scaling Techniques
Elicitation Topic Map (ETM)

**Topic Set 1 (independent)**

**Items** (who/what?): the salient entities existing. Those entities can be human or not, living or not, physical or not.

Examples: employees of a company, furniture, servers, printers, but also abstract entities such as ideas or knowledge.

**Rules** (why/how?): the constraints that are applicable, and which somehow influence the behaviour of Items.

Examples: laws, norms, cultures or habits.

**Localization** (where/when?): the physical position of. Localization is divided into two subcategories: time and place.

**Topic set 2 (dependent)**

**Activities:** the goals and actions of Items.

Examples: business strategies, people’s personal motivations, intentions, and goals.

**Connections:** the relationships between Items and/or Rules.

Examples: collaboration, friendship, competition, and applicability of a rule.

**Granularity:** the nature, the quantity and the level of any additional piece of information that is provided.

Examples: age of a person, the temperature in a room or the sanction that is applicable when a rule is violated.
ETM
Example:

Goal is to develop a new social network system.

<table>
<thead>
<tr>
<th>Items</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1. Type of person you want to meet on the new network?</td>
<td>R1. Laws and regulations that you can think of, that influence the way you use social networks?</td>
</tr>
<tr>
<td>I2. Device(s) you want to use to access the new network?</td>
<td>R2. Personal rules when you share content on a social network?</td>
</tr>
<tr>
<td>I3. Other networks or online services that you already use?</td>
<td>R3. Your habits, which are not directly related to the social network?</td>
</tr>
<tr>
<td>I4. Type of content you want to share on the new network?</td>
<td>R4. Suggestions that your parents, friends, or others, give you, about using a social network?</td>
</tr>
<tr>
<td>I5. Way you categorize friends you have in different groups?</td>
<td>R5. Best practices when sharing content on a network?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1. Place where you want to access the new network?</td>
<td>A1. Main occupations you have during the day?</td>
</tr>
<tr>
<td>L2. Routines you have when you come back from school/work?</td>
<td>A2. Reason why you need the new network?</td>
</tr>
<tr>
<td>L3. How often you post status updates, pictures, etc.?</td>
<td>A3. Things you want to do with the new network?</td>
</tr>
<tr>
<td>L4. What recurring events would you like to be notified of?</td>
<td>A4. Number of contacts you want to have, so that your network is large enough and hence useful?</td>
</tr>
<tr>
<td>L5. What past events you want to share?</td>
<td>A5. Values or ethical ideas that you feel are important?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection</th>
<th>Granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1. Type of relationships you expect to find on the new network?</td>
<td>G1. Atmosphere between you and your friends</td>
</tr>
<tr>
<td>C2. Different privileges you want to provide to your contacts?</td>
<td>G2. Information about marital status, children, or your revenue</td>
</tr>
<tr>
<td>C3. Extent to which the new network would replace existing ones?</td>
<td>G3. Way you want to evaluate your new social network?</td>
</tr>
<tr>
<td>C4. Strength of relationships you expect to establish on the new network?</td>
<td>G4. Collaborations you expect to find between some groups of friends?</td>
</tr>
<tr>
<td>C5. Extent to which you trust the analyst that interviews you?</td>
<td>G5. Some peculiarities about your friends or your family?</td>
</tr>
</tbody>
</table>
Collecting Information

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?
Is it really just that?
RE is difficult, because ...

... software value-chains are getting more and more complex

... and might be geographically distributed world-wide
Many Stakeholders …

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

Internal:
- Software Developers
- Development (Project) Managers
- Product Managers
  - …
Example of a Real World Situation …
(Mobile Phones)

Where do all requirements come from?

... find the right person to talk to …
... get the deep domain knowledge ….
Project setting shapes how RE is done …

**In-house:** intern development for own use

**Product development:** development for open (mass) market (→ marked-driven development)

**Time & materials based:** compensation based on time/effort and material costs

**Components-off-the-shelf:** acquisition of generic (shelved) software (components)

**Tender:** bidding for a contract by a third party
  - Procurement of customer-specific development
  - Procurement of COTS development

**Contract development:** contract-based development with fixed or variable price

**Sub-contracting:** sub-contracted third-party development with fixed or variable price

**Unknown:** pre-study required to determine project type

**Hybrid:** combination of any of the above

<table>
<thead>
<tr>
<th>Project types</th>
<th>Customer</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house</td>
<td>User dept.</td>
<td>IT dept.</td>
</tr>
<tr>
<td>Prod. devel.</td>
<td>Marketing</td>
<td>SW dept.</td>
</tr>
<tr>
<td>Time &amp; materials</td>
<td>Company</td>
<td>SW house</td>
</tr>
<tr>
<td>COTS</td>
<td>Company</td>
<td>(Vendor)</td>
</tr>
<tr>
<td>Tender</td>
<td>Company</td>
<td>Supplier</td>
</tr>
<tr>
<td>Contract devel.</td>
<td>Company</td>
<td>SW house</td>
</tr>
<tr>
<td>Sub-contracting</td>
<td>Supplier</td>
<td>SW house</td>
</tr>
<tr>
<td>Unknown</td>
<td>Inhouse?</td>
<td>COTS?</td>
</tr>
</tbody>
</table>

From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002
Complex Customer – Supplier Relationships

Who has the power?
Who has the knowledge?

Who takes the biggest risk?
Who takes the biggest profit?

In the short term?
In the long run?

Mutual benefit?
RE questions to answer …

• How to Elicit Requirements?
  • Different Project Settings
  • Complex Customer-Supplier Relationships / Value-Chains
  • Different Levels of Requirements
  • Many (Types of) Stakeholders
• How to Describe/Specify Requirements?
  • Different Levels & Types of Requirements
  • Quality of Requirements?
  • Different Representation Styles
• How to Prioritize/Assign/Manage Requirements?
Definition: Requirements

- Requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process.

  (Sommerville, 2010)
The Goal of RE

- What the Customer wants
- What the Customer needs
- What the Software does

Application Domain (User Requirements)

System Domain (System Requirements)
User vs. System Requirements

User requirements

- Statements in natural language plus diagrams of the services the system provides and its operational constraints.
- Written for customers.

System requirements

- A structured document setting out detailed descriptions of the system’s functions, services and operational constraints.
- Defines what should be implemented and thus may be part of a contract between client and contractor.
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
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 compiler?
Software Quality – Definition

- Software quality is the degree of conformance to explicit or implicit requirements and expectations

Explanation:
- **Explicit**: clearly defined and documented
- **Implicit**: not clearly defined and documented but indirectly suggested
- **Requirements**: business/product/software requirements
- **Expectations**: mainly end-user expectations
Software Quality – Dimensions (1)

- **Functionality**: The ability of software to carry out the functions as specified or desired.
- **Accessibility**: The degree to which software can be used comfortably by a wide variety of people, including those who require assistive technologies like screen magnifiers or voice recognition.
- **Compatibility**: The suitability of software for use in different environments like different Operating Systems, Browsers, etc.
- **Concurrency**: The ability of software to service multiple requests to the same resources at the same time.
- **Efficiency**: The ability of software to perform well or achieve a result without wasted energy, resources, effort, time or money.
- **Installability**: The ability of software to be installed in a specified environment.
- **Localizability**: The ability of software to be used in different languages, time zones etc.
Software Quality – Dimensions (2)

- **Maintainability:** The ease with which software can be modified (adding features, enhancing features, fixing bugs, etc).
- **Performance:** The speed at which software performs under a particular load.
- **Portability:** The ability of software to be transferred easily from one location to another.
- **Reliability:** The ability of software to perform a required function under stated conditions for stated period of time without any errors.
- **Scalability:** The measure of software’s ability to increase or decrease in performance in response to changes in software’s processing demands.
- **Security:** The extent of protection of software against unauthorized access, invasion of privacy, theft, loss of data, etc.
- **Testability:** The ability of software to be easily tested.
- **Usability:** The degree of software’s ease of use.
Software Product Quality Model
– ISO 25010:2011 Standard
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)
# Example – Usability Requirements

<table>
<thead>
<tr>
<th>Problem counts</th>
<th>Task time</th>
<th>Keystroke counts</th>
<th>Opinion poll</th>
<th>Score for understanding</th>
<th>Design-level reqs</th>
<th>Product-level reqs</th>
<th>Guideline adherence</th>
<th>Development process 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>R2: Novice users shall perform tasks Q and R in 15 minutes. Experienced users tasks Q, R, S in 2 minutes.</td>
<td>R3: Recording breakfast shall be possible. No mouse.</td>
<td>R4: 80% of users shall find system easy to recommend system to others.</td>
<td>R5: Show 5 users 10 common error messages. Ask for the cause. 80% of the time</td>
<td>R6: System shall use screen pictures in app. xx, buttons work as app. yy.</td>
<td>R7: For all code fields, user shall be able to select value from drop-down list.</td>
<td>R8: System shall follow style guide xx. Menus shall have at most three levels.</td>
<td>R9: Three prototype versions shall be made and usability tested during design.</td>
</tr>
</tbody>
</table>
Representation of Requirements

• Natural Language
• Structured Natural Language
  • e.g., User Stories, Use Case Descriptions
• Semi-Formal Representations
  • e.g. Use Case Models (and other UML diagrams)
• Formal Representations
  • Requirements specification languages with precisely defined syntax and semantics
  • Can be used to automatically check consistency, generate code, generate tests, etc.
NL Requirements (unstructured)
Why are NL Requirements problematic?
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 user intention vs. developer interpretation of 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: 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)

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’

<table>
<thead>
<tr>
<th>S</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Measurable</td>
</tr>
<tr>
<td>A</td>
<td>Attainable (Achievable, Actionable, Appropriate)</td>
</tr>
<tr>
<td>R</td>
<td>Realistic</td>
</tr>
<tr>
<td>T</td>
<td>Time-bound (Timely, Traceable)</td>
</tr>
</tbody>
</table>
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/](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)

In software engineering, you may see the “T” in SMART being used to mark whether a requirement is “traceable”, which is a separate but important topic in developing software.

Source: http://jessica80304.wordpress.com/2008/08/04/smart-requirements/
User Stories
User Story – Example

As a tenant, I can unlock the doors to enter my apartment.

- user-role (benefactor)
- capability (functionality)
- business-value (motivation/rationale)

who - what - why

• Similar to NL requirements, but focus on the user benefits, instead on system characteristics (alone).

• Unfortunately, third element (business-value) is often ommitted

• Preferred technique in agile methods.
‘Normal’ User Story

<Actor/Role> As a user

... 

<Action> I want to narrow down people search results by location

... 

<Value> so I can find the right person more quickly

A good User Story is:
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

INVEST
‘Normal’ User Story

<Actor/Role> As a user

...  

<Action> I want to narrow down people search results by location

...  

<Value> so I can find the right person more quickly

Acceptance test:

Given I am on the search screen
And 'Paula' is on the same indexed page with 'Tartu'

When I search for 'Paula'
Then I see 'Tartu' in the location section of the search results
‘Non-Functional’ User Story

<Actor/Role> As a user
...

<Action> I want error messages to be shown always at the same place of the screen
...

<Value> so I can easily detect them

A good User Story is:
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

INVEST
‘Device’ or ‘System’ User Story

<Actor/System> As a web crawler

... I need a URL dictionary without duplicates or dead links

... so that the crawling process is faster

A good User Story is:
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

INVEST
‘Technical’ User Story

<Action> Implement Smart Distance Algorithm for multi-paragraph web-pages

<Value> so that relevant and random connections between search entities could be distinguished

A good User Story is:

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

INVEST
Release Planning
Why do Release Planning?

• Too many requirements (features) for given time and capacity

• RP goes hand in hand with incremental and agile software development
What is Release Planning?
What is a Good Release Plan?

A good release plan should …

• provide maximum business value by
  • offering the best possible blend of features
  • in the right sequence of releases,

• satisfy the most important stakeholders involved,

• be feasible with available resources, and

• reflect existing dependencies between features.
How to do Release Planning?

Art vs. Science
Baseline: Release planning “on the fly”

- Informal process
- Informal decisions
- Resource situation ignored
- Stakeholders left out
- But: $4^{20} > 1.000.000.000.000$ possibilities already in case of 20 features and three releases
Example

<table>
<thead>
<tr>
<th>Feature</th>
<th>F(a)</th>
<th>F(b)</th>
<th>F(c)</th>
<th>F(d)</th>
<th>F(e)</th>
<th>F(f)</th>
<th>F(g)</th>
<th>F(h)</th>
<th>F(i)</th>
<th>F(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benefit</td>
<td>6</td>
<td>8.5</td>
<td>4.5</td>
<td>7.5</td>
<td>7</td>
<td>6.5</td>
<td>9</td>
<td>5.5</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Cost</td>
<td>10</td>
<td>90</td>
<td>30</td>
<td>23</td>
<td>22</td>
<td>20</td>
<td>100</td>
<td>30</td>
<td>30</td>
<td>25</td>
</tr>
</tbody>
</table>

Planning for two releases ahead of time
Capacity(Release 1) = 100 (K dollar),
Capacity(Release 2) = 90
Greedy release planning …

Begin
// Assume value(1) ≥ value(2) ≥ .. ≥ value(N)
For k = 1…K Do
// k denotes the release index//
    Begin
        ActualCost(k) = 0,
        ActualValue(k) = 0,
        Release(k) = Ø
    For n = 1…N Do
        // n denotes the index of the feature under consideration //
        Begin
            If f(n) in F & ActualCost(k) + cost(n) ≤ TotalCost(k) Then
                Begin
                    Release(k) := Release(k) + {f(n)}, F = F - {f(n)},
                    ActualCost(k) := ActualCost(k) + cost(n)
                    ActualValue(k) := ActualValue(k) + value(n)
                End
            End
        End
    End
End

©Guenther Ruhe
MTAT.03.244 / © Dietmar Pfahl 2016
Why is greedy not good enough?

Planning for two releases ahead of time
Capacity(1) = 100 (K dollar), Capacity(2) = 90

Greedy benefit solution:
Release(1) = \{f(1)\}
Release(2) = \{f(2)\}

Optimized cost-benefit solution:
Release(1) = \{f(3), f(4), f(5), f(6), f(7)\}
Release(2) = \{f(8), f(9) and f(10)\}

Ratio: Greedy/Optimized = 13.25/42.5 = 31.2%

<table>
<thead>
<tr>
<th>Feature</th>
<th>f(1)</th>
<th>f(2)</th>
<th>f(3)</th>
<th>f(4)</th>
<th>f(5)</th>
<th>f(6)</th>
<th>f(7)</th>
<th>f(8)</th>
<th>f(9)</th>
<th>f(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benefit</td>
<td>9</td>
<td>8.5</td>
<td>8</td>
<td>7.5</td>
<td>7</td>
<td>6.5</td>
<td>6</td>
<td>5.5</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>Cost</td>
<td>100</td>
<td>90</td>
<td>25</td>
<td>23</td>
<td>22</td>
<td>20</td>
<td>10</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>
Release Planning – Modeling

Formal definition of objective function and all relevant constraints

This involves:

• Identify and select stakeholders
  ➔ This might include a weighting of stakeholders
• Define criteria for feature selection
  ➔ Some criteria might be more important than others
• Offer stakeholder voting (priority, urgency, risk …)
• Estimate (cost, effort, capacity …)
Stakeholders

ROADMAPPING PROJECT

- Management
- Developers
- Customers
- Quality assurance
- Competitors
- Standards and legislation
- (Legacy) System and product constraints
- Internal accounting, marketing and sales
Criteria for feature prioritization

- Customer satisfaction
- Customer dissatisfaction
- Risk of implementation
- Risk of acceptance
- Financial value
- Cost
- Time to market
- Volatility
- Frequency of use
- Ease of use

Problem:
What to do when we have several (conflicting) prioritisation criteria?
Example

Feature

EFFORT (estimated)
(NB: Only one type of resource = engineers: effort in ph)

Prioritization Engineering wants to minimize the risk
Prioritization Marketing wants to maximize the value
Prioritization Customers want the high value
... the low risk
NB: for risk, higher score means lower risk!

Importance of Release (equal = all set to 5)

Constraint
F01 = F03 (in same release)
Pre-assignment of features to releases
More on feature dependencies

For given features A, B, and C, we distinguish eight types of dependencies:
Starting Situation:
Set of features $F = \{f(1), f(2), \ldots, f(n)\}$
Features may be
- New functionality
- Customer change requests
- Defect corrections

Goal:
Assign the features to a finite set number $K$ of release options.
RP Modeling (2)

Decision Variables:

Each individual release plan describes the strategy for mapping all candidate features to releases.

The mapping is described by a vector of decision variables \( \{x(1), x(2), \ldots, x(n)\} \)

where

- \( x(i) = k \) if feature \( f(i) \) is assigned to release \( k \) 
  
  \( (k \in \{1, \ldots, K\}) \)

- \( x(i) = 0 \) if feature \( f(i) \) is postponed
Feature Dependencies:

Various dependencies between features are possible, and most features are involved in some sort of dependency relationship.

- **Coupling:**
  Coupled features should be released jointly because they depend on each other

- **Precedence:**
  Precedence features should be released in a certain order
RP Modeling (4)

Resource Constraints:

Typically, project managers must consider several kinds of resource constraints.

Usually, these constraints relate to either budget or effort, and there is a maximum capacity available of each resource.

Formalisation:

- Resource types 1, …, T
- Releases 1, …, K
- Resource consumption per feature and type $r(i, t)$
- Resource capacity per release and type $Cap(k, t)$
Prioritisation:

We can use different criteria to prioritise features. These criteria can be conflicting.

Method of prioritisation by stakeholders: e.g., using a 9-point scale

Examples: value, urgency, risk, …

Formalisation:

Assuming 3 releases, the prioritisations

• value(p, i) = 9 and urgency(p, i) = (9, 0, 0) indicate that stakeholder p (from \{1, …, S\}) has assigned the highest possible value to feature f(i) and in addition thinks it is most urgent.

• urgency(p, i) = (3,3,3) indicates that stakeholder p has no urgency preference.
RP Modeling (6)

Objective Function:
Maximise $F(x)$ defined as

$$F(x) = \sum_{k=1,\ldots,K} \sum_{i: x(i) = k} \text{WAS}(i, k)$$

where

$$\text{WAS}(i, k) = \xi(k) \left[ \sum_{p=1,\ldots,q} \lambda(p) \text{value}(p, i) \text{urgency}(p, i, k) \right]$$

with:

- \text{WAS}: weighted average satisfaction of stakeholder priorities
- $\lambda(p)$: relative importance of stakeholder $p$
- $\xi(k)$: relative importance of release $k$
- $\sum_{k=1,\ldots,K} \xi(k) = 1$ and $\sum_{p=1,\ldots,S} \lambda(p) = 1$
EVOLVE II: Three phases

Phase 1 - Modeling:

Formal description of the (changing) real world to make it suitable for computational intelligence based solution techniques.

Phase 2 - Exploration:

Application of computational techniques to explore the solution space, to generate and evaluate solution alternatives.

Phase 3 - Consolidation:

Human decision maker evaluates current solution alternatives.

Match with implicit objectives and constraints.
## Table 1

### Features, resource consumption, and stakeholder feature evaluations

<table>
<thead>
<tr>
<th>Feature $f(i)$</th>
<th>Analyst &amp; designers (hrs) $r(i,1)$</th>
<th>Developers (hrs) $r(i,2)$</th>
<th>QA (hrs) $r(i,3)$</th>
<th>Budget (US$ in thousands) $r(i,4)$</th>
<th>Stakeholder $S(1)$</th>
<th>Stakeholder $S(2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Cost reduction of transceiver</td>
<td>150</td>
<td>120</td>
<td>20</td>
<td>1,000</td>
<td>6</td>
<td>(5, 4, 0)</td>
</tr>
<tr>
<td>2. Expand memory on BTS controller</td>
<td>75</td>
<td>10</td>
<td>8</td>
<td>200</td>
<td>7</td>
<td>(5, 0, 4)</td>
</tr>
<tr>
<td>3. FCC out-of-band emissions</td>
<td>400</td>
<td>100</td>
<td>20</td>
<td>200</td>
<td>9</td>
<td>(9, 0, 0)</td>
</tr>
<tr>
<td>4. Software quality initiative</td>
<td>450</td>
<td>100</td>
<td>40</td>
<td>0</td>
<td>5</td>
<td>(2, 7, 0)</td>
</tr>
<tr>
<td>5. USEast Inc., Feature 1</td>
<td>100</td>
<td>500</td>
<td>40</td>
<td>0</td>
<td>3</td>
<td>(7, 2, 0)</td>
</tr>
<tr>
<td>6. USEast Inc., Feature 2</td>
<td>200</td>
<td>400</td>
<td>25</td>
<td>25</td>
<td>9</td>
<td>(7, 2, 0)</td>
</tr>
<tr>
<td>7. China Feature 1</td>
<td>50</td>
<td>250</td>
<td>20</td>
<td>500</td>
<td>5</td>
<td>(9, 0, 0)</td>
</tr>
<tr>
<td>8. China Feature 2</td>
<td>60</td>
<td>120</td>
<td>19</td>
<td>200</td>
<td>7</td>
<td>(8, 1, 0)</td>
</tr>
<tr>
<td>9. 12-carrier BTS for China</td>
<td>280</td>
<td>150</td>
<td>40</td>
<td>1,500</td>
<td>6</td>
<td>(9, 0, 0)</td>
</tr>
<tr>
<td>10. Pole-mount packaging</td>
<td>200</td>
<td>300</td>
<td>40</td>
<td>500</td>
<td>2</td>
<td>(5, 4, 0)</td>
</tr>
<tr>
<td>11. Next-generation BTS</td>
<td>250</td>
<td>375</td>
<td>50</td>
<td>150</td>
<td>1</td>
<td>(8, 1, 0)</td>
</tr>
<tr>
<td>12. India BTS variant</td>
<td>100</td>
<td>300</td>
<td>25</td>
<td>50</td>
<td>3</td>
<td>(9, 0, 0)</td>
</tr>
<tr>
<td>13. Common feature 01</td>
<td>100</td>
<td>250</td>
<td>20</td>
<td>50</td>
<td>7</td>
<td>(9, 0, 0)</td>
</tr>
<tr>
<td>14. Common feature 02</td>
<td>0</td>
<td>100</td>
<td>15</td>
<td>0</td>
<td>8</td>
<td>(9, 0, 0)</td>
</tr>
<tr>
<td>15. Common feature 03</td>
<td>200</td>
<td>150</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>(0, 0, 9)</td>
</tr>
<tr>
<td><strong>Total resource consumption</strong></td>
<td><strong>2,615</strong></td>
<td><strong>3,225</strong></td>
<td><strong>392</strong></td>
<td><strong>4,375</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Available capacity, Release 1</td>
<td>1,300</td>
<td>1,450</td>
<td>158</td>
<td>2,200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Available capacity, Release 2</td>
<td>1,046</td>
<td>1,300</td>
<td>65</td>
<td>1,750</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**EVOLVE II: Example (Part 2)**

**Table 2**

Two qualified release plan alternatives, listing the release to which each feature is assigned and each weighted average satisfaction

<table>
<thead>
<tr>
<th>Feature $f(i)$</th>
<th>Release Plan $x_1$</th>
<th>Release Plan $x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1(i)$</td>
<td>WAS($i, k$)</td>
</tr>
<tr>
<td>1. Cost reduction of transceiver</td>
<td>1</td>
<td>84.0</td>
</tr>
<tr>
<td>2. Expand memory on BTS controller</td>
<td>1</td>
<td>287.0</td>
</tr>
<tr>
<td>3. FCC out-of-band emissions</td>
<td>1</td>
<td>252.0</td>
</tr>
<tr>
<td>4. Software quality initiative</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>5. USEast, feature 1</td>
<td>1</td>
<td>134.4</td>
</tr>
<tr>
<td>6. USEast, feature 2</td>
<td>2</td>
<td>516.6</td>
</tr>
<tr>
<td>7. China feature 1</td>
<td>2</td>
<td>277.2</td>
</tr>
<tr>
<td>8. China feature 2</td>
<td>2</td>
<td>43.2</td>
</tr>
<tr>
<td>9. 12-carrier BTS for China</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>10. Pole-mount packaging</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>11. Next-generation BTS</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>12. India BTS variant</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>13. Common feature 01</td>
<td>1</td>
<td>37.8</td>
</tr>
<tr>
<td>14. Common feature 02</td>
<td>1</td>
<td>8.4</td>
</tr>
<tr>
<td>15. Common feature 03</td>
<td>2</td>
<td>54.0</td>
</tr>
</tbody>
</table>

| Objective function value $F(x)$            | 1,694.6            | 1,708.0            |
Tool Demo: ReleasePlanner 2.0
Homework: Assignment 3