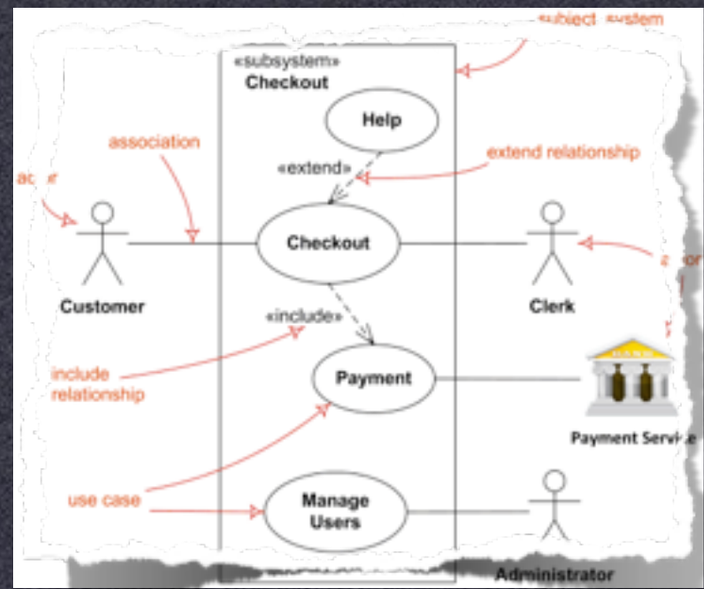


Week 4: Specifying Requirements

400220 Software Requirements Engineering

Jim Buchan

The aims for Today

- Review of Elicitation technique
- SRS and standard
- Use cases
- Scenarios
- User Stories
- Personas
- Research and empirical evidence

Requirements *Discovery*

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?
- Rephrase the following questions to be context free and open-ended.

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?
- Rephrase the following questions to be context free and open-ended.
  - Do you think the user should have to enter a password?

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?
- Rephrase the following questions to be context free and open-ended.
  - Do you think the user should have to enter a password?
  - Should the system automatically save the user's work every 15 minutes?

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?
- Rephrase the following questions to be context free and open-ended.
  - Do you think the user should have to enter a password?
  - Should the system automatically save the user's work every 15 minutes?
  - Can one user see database entries saved by another user??

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Describe how the system should safeguard sensitive data.

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?
- Rephrase the following questions to be context free and open-ended.
  - Do you think the user should have to enter a password?
  - Should the system automatically save the user's work every 15 minutes?
  - Can one user see database entries saved by another user??

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Describe how the system should safeguard sensitive data.

What would a user do if the system crashed while she was using it?

Requirements *Discovery*

- Why not rely on questionnaires for all requirements trawling?
- Interviews (group or 1-1) - What kind of questions NOT to ask
- Why is it best to ask open-ended, context-free questions?
- Rephrase the following questions to be context free and open-ended.
  - Do you think the user should have to enter a password?
  - Should the system automatically save the user's work every 15 minutes?
  - Can one user see database entries saved by another user??

Questionnaires can take a long time to turn around so the project will take longer to complete. Someone will need to aggregate and interpret the results, which means there will be some degree of misinterpretation. Because questionnaires do not provide for true two-way communication, it will be extremely hard for a team to get feedback as to whether it's on the right track.

Leading questions that steer customers to agree with your own intentions.

Also avoid overruling a customer's idea just because it doesn't agree with your own point of view.

Context-free questions do not imply an answer so the respondent does not feel a need to give the "right" answer. Open-ended questions allow detailed responses that go beyond a simple yes or no. Open-ended, context-free questions are best because they do not influence the response and they allow a broader range of responses than yes or no.

Describe how the system should safeguard sensitive data.

What would a user do if the system crashed while she was using it?

Tell me about the accessibility of data saved by a user..

Questions to Ask

- What business problem are you trying to solve?
- What's the motivation for solving this problem?
- What would a highly successful solution do for you?
- How can we judge the success of the solution?
- Which business activities and events should be included in the solution? Which should not?
- Can you think of any unexpected or adverse consequences that the new system could cause?
- What's a successful solution worth
- Who are the individuals or groups that could influence this project or be influenced by it?
- Are there any related projects or systems that could influence this one or that this project could affect?

AUT COMPUTING + MATHEMATICAL SCIENCES

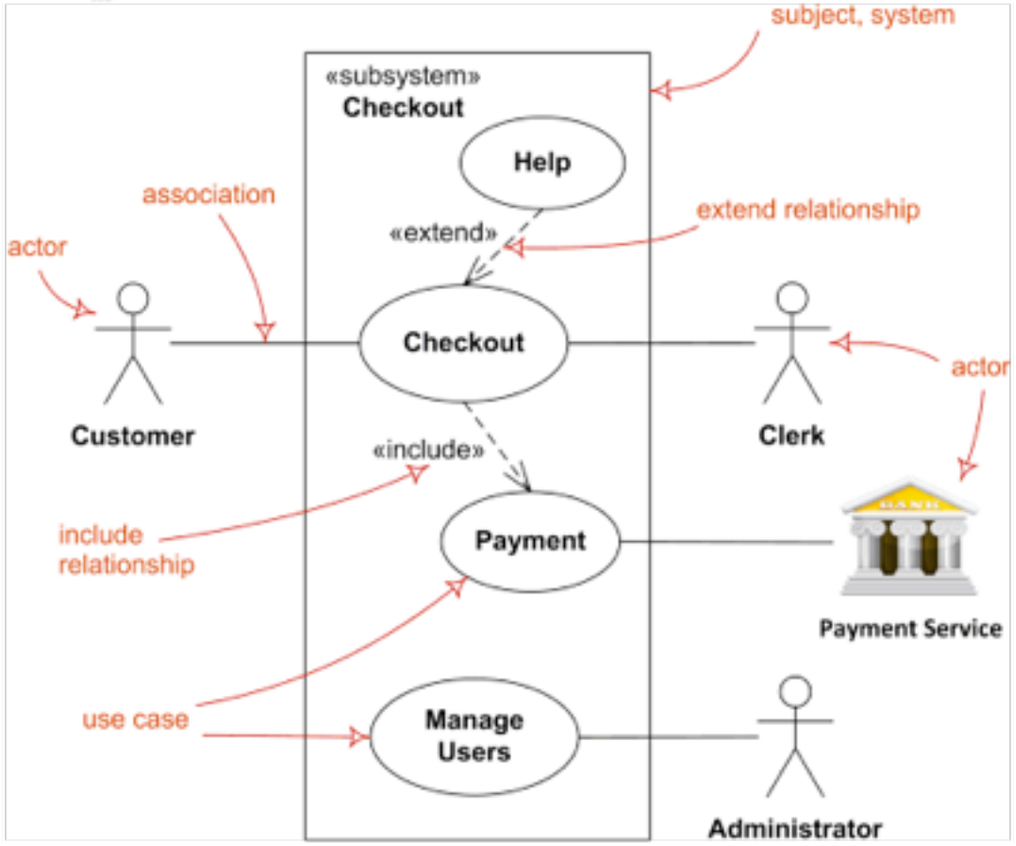
Requirements Specification

- Software Requirements Specification (eg STANDARD IEEE 29148:2011)
- Use Case
- Scenarios
- Feature List
- User Stories

INTERNATIONAL **ISO/IEC/** STANDARD **IEEE 29148**

First edition 2011-12-01

Systems and software engineering — Life cycle processes — Requirements engineering



ISO/IEC/IEEE 29148:2011

5 Concepts
5.1 Introduction
5.2 Requirements fundamentals
5.2.1 General
5.2.2 Stakeholders
5.2.3 Transformation of needs into requirements
5.2.4 Requirements construct
5.2.5 Characteristics of individual requirements
5.2.6 Characteristics of a set of requirements
5.2.7 Requirement language criteria
5.2.8 Requirements attributes
5.3 Practical considerations
5.3.1 Iteration and recursion of processes
5.3.2 Iteration and recursion in requirements engineering
5.4 Requirement information items

6 Processes
6.1 Requirement processes
6.1.1 Guidelines for Processes
6.2 Stakeholder requirements definition process
6.2.1 Purpose
6.2.2 Outcomes
6.2.3 Activities and tasks
6.3 Requirements analysis process
6.3.1 Purpose
6.3.2 Outcomes
6.3.3 Activities and tasks
6.4 Requirements engineering activities in other technical processes
6.4.1 Requirements in architectural design
6.4.2 Requirements in verification
6.4.3 Requirements in validation
6.5 Requirements management
6.5.1 Management Overview
6.5.2 Change management
6.5.3 Measurement for requirements

8 Guidelines for information items 42
8.1 Requirements information item outlines 42
8.2 Stakeholder requirements specification document 42
8.2.1 Introduction 42
8.2.2 StRS example outline 42
8.3 System requirements specification document 43
8.3.1 Introduction 43
8.3.2 SyRS example outline 44
8.4 Software requirements specification document 45
8.4.1 Introduction 45
8.4.2 SRS example outline 45

9 Information item content 46
9.1 Introduction 46
9.2 General content 46
9.2.1 Identification 46
9.2.2 Front matter 47
9.2.3 Definitions 47
9.2.4 References 47
9.2.5 Acronyms and abbreviations 47
9.3 Stakeholder requirements specification (StRS) document 47
9.3.1 Business purpose 47
9.3.2 Business scope 48
9.3.3 Business overview 48
9.3.4 Stakeholders 48
9.3.5 Business environment 48
9.3.6 Goal and Objective 48
9.3.7 Business model 48
9.3.8 Information environment 48
9.3.9 Business processes 49
9.3.10 Business operational policies and rules 49
9.3.11 Business operational constraints 49
9.3.12 Business operation modes 49
9.3.13 Business operational quality 49
9.3.14 Business structure 49
9.3.15 User requirements 49
9.3.16 Operational concept 50
9.3.17 Operational scenarios 50
9.3.18 Project constraints 50
9.4 System requirements specification (SyRS) document 50
9.4.1 System purpose 50
9.4.2 System scope 50
9.4.3 System overview 51
9.4.4 Functional requirements 51
9.4.5 Usability requirements 51
9.4.6 Performance requirements 51
9.4.7 System interfaces 51
9.4.8 System Operations 52
9.4.9 System modes and states 52
9.4.10 Physical characteristics 52
9.4.11 Environmental conditions 53
9.4.12 System security 53
9.4.13 Information management 53
9.4.14 Policies and regulations 53
9.4.15 System life cycle sustainment 53
9.4.16 Packaging, handling, shipping and transportation 53
9.4.17 Verification 54
9.4.18 Assumptions and dependencies 54
9.5 Software requirements specification (SRS) document 54

ENCES

ISO/IEC/IEEE 29148:2011

5 Concepts
5.1 Introduction
5.2 Requirements fundamentals
5.2.1 General
5.2.2 Stakeholders
5.2.3 Transformation of needs into requirements
5.2.4 Requirements construct
5.2.5 Characteristics of individual requirements
5.2.6 Characteristics of a set of requirements
5.2.7 Requirement language criteria
5.2.8 Requirements attributes
5.3 Practical considerations
5.3.1 Iteration and recursion of processes
5.3.2 Iteration and recursion in requirements engineering
5.4 Requirement information items

6 Processes
6.1 Requirement processes
6.1.1 Guidelines for Processes
6.2 Stakeholder requirements definition process
6.2.1 Purpose
6.2.2 Outcomes
6.2.3 Activities and tasks
6.3 Requirements analysis process
6.3.1 Purpose
6.3.2 Outcomes
6.3.3 Activities and tasks
6.4 Requirements engineering activities in other technical processes
6.4.1 Requirements in architectural design
6.4.2 Requirements in verification
6.4.3 Requirements in validation
6.5 Requirements management
6.5.1 Management Overview
6.5.2 Change management

8 Guidelines for information items 42
8.1 Requirements information item outlines 42
8.2 Stakeholder requirements specification document 42
8.2.1 Introduction 42
8.2.2 StRS example outline 42
8.3 System requirements specification document 43
8.3.1 Introduction 43
8.3.2 SyRS example outline 44
8.4 Software requirements specification document 45
8.4.1 Introduction 45
8.4.2 SRS example outline 45

9 Information item content 46
9.1 Introduction 46
9.2 General content 46
9.2.1 Identification 46
9.2.2 Front matter 47
9.2.3 Definitions 47
9.2.4 References 47
9.2.5 Acronyms and abbreviations 47
9.3 Stakeholder requirements specification (StRS) document 47
9.3.1 Business purpose 47
9.3.2 Business scope 48
9.3.3 Business overview 48
9.3.4 Stakeholders 48
9.3.5 Business environment 48
9.3.6 Goal and Objective 48
9.3.7 Business model 48
9.3.8 Information environment 48
9.3.9 Business processes 49
9.3.10 Business operational policies and rules 49
9.3.11 Business operational constraints 49
9.3.12 Business operation modes 49
9.3.13 Business operational quality 49
9.3.14 Business structure 49
9.3.15 User requirements 49
9.3.16 Operational concept 50
9.3.17 Operational scenarios 50
9.3.18 Project constraints 50
9.4 System requirements specification (SyRS) document 50
9.4.1 System purpose 50
9.4.2 System scope 50
9.4.3 System overview 51
9.4.4 Functional requirements 51
9.4.5 Usability requirements 51
9.4.6 Performance requirements 51
9.4.15 System life cycle sustainment 53
9.4.16 Packaging, handling, shipping and transportation 53
9.4.17 Verification 54
9.4.18 Assumptions and dependencies 54
9.5 Software requirements specification (SRS) document 54

THROUGH AUT LIBRARY
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6146379

ENCES

IEEE 830-1998 (superseded SRS standard)

4.3 Characteristics of a good SRS

An SRS should be

a) Correct;
b) Unambiguous;
c) Complete;
d) Consistent;
e) Ranked for importance and/or stability;
f) Verifiable;
g) Modifiable;
h) Traceable.

SRS Outline

1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

2. References

3. Specific requirements
- 3.1 External interfaces
- 3.2 Functions
- 3.3 Usability Requirements
- 3.4 Performance requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

4. **Verification**
(parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

Figure 8 — Example SRS Outline

AUT COMPUTING + MATHEMATICAL SCIENCES

SRS Outline

1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

2. References

3. Specific requirements
- 3.1 External interfaces
- 3.2 Functions
- 3.3 Usability Requirements
- 3.4 Performance requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

4. **Verification**
(parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

Figure 8 — Example SRS Outline

1. Introduction
- 1.1 System purpose
- 1.2 System scope
- 1.3 System overview
  - 1.3.1 System context
  - 1.3.2 System functions
  - 1.3.3 User characteristics
- 1.4 Definitions

2. References

3. System requirements
- 3.1 Functional requirements
- 3.2 Usability requirements
- 3.3 Performance requirements
- 3.4 System interface
- 3.5 System operations
- 3.6 System modes and states
- 3.7 Physical characteristics
- 3.8 Environmental conditions
- 3.9 System security
- 3.10 Information management
- 3.11 Policies and regulations
- 3.12 System life cycle sustainment
- 3.13Packaging, handling, shipping and transportation

4. Verification
(parallel to subsections in Section 3)

5. Appendices
- Assumptions and dependencies
- Acronyms and abbreviations

Figure 7 — Example SyRS Outline

Example of Specific requirements in IEEE 830-1998 (section 9.5.10 in newer standard)

5.3.1 External interfaces

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in 5.2 and should not repeat information there.

It should include both content and format as follows:

a) Name of item;
b) Description of purpose;
c) Source of input or destination of output;
d) Valid range, accuracy, and/or tolerance;
e) Units of measure;
f) Timing;
g) Relationships to other inputs/outputs;
h) Screen formats/organization;
i) Window formats/organization;
j) Data formats;
k) Command formats;
l) End messages.

Example of Specific requirements in IEEE 830-1998 (section 9.5.10 in newer standard)

5.3.1 External interfaces

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in 5.2 and should not repeat information there.

It should include both content and format as follows:

a) Name of item;
b) Description of purpose;
c) Source of input or destination of outpu
d) Valid range, accuracy, and/or toleranc
e) Units of measure;
f) Timing;
g) Relationships to other inputs/outputs;
h) Screen formats/organization;
i) Window formats/organization;
j) Data formats;
k) Command formats;
l) End messages.

- Functions
- Performance Requirements
- Logical Database requirements
- Design Constraints (e.g. Standards Compliance)
- Software System Attributes (e.g. Reliability, Availability, Security, Maintainability, Portability)

Example Functional requirements

Example Functional requirements

4.6) ***The system shall*** allow a company to pay for a job posting with a credit card.

4.6.1) The system shall accept Visa, MasterCard and American Express cards.

4.6.2) The system shall charge the credit card before the job posting is placed on the site.

4.6.3) The system shall give the user a unique confirmation number.

SRS E-Commerce example

ACS560

Electronic Commerce System Case Study

Minhad Keserović

Software Requirements Specification

Document

Version: (1.0) **Date: (12/06/2009)**

AUT UNIVERSITY COMPUTER + MATHEMATICAL SCIENCES

False sense of Certainty? Constrain options early?

- **Documenting a system's requirements to this level is:**

very time-consuming->tedious->error-prone.

Lose track. Take shortcuts. Forget.

- **Reading this document to verify requirements is:**

very time-consuming->tedious->error-prone

Skim/skip read. Lose track. Forget.

- May lose the big picture if get lost in the detail.

Cohn, M. (2004). User stories applied: For agile software development.

How and Why Will the Feature Be Used?

- SRS- lists is that the items on the lists describe the behavior of the software, not the behavior or goals of a user. ***A requirements list rarely answers "But how and why will someone use this feature?***"

- Steve Berczuk, XP developer and author of Software Configuration Management Patterns, points out the importance of this question: "I can't count the number of times that I saved a whole lot of work by taking a feature list and asking a customer to create scenarios that use those features. Often the customer will realize that the feature really isn't needed, and that you should spend time building things that add value."[3]

[3] Steve Berczuk on extremeprogramming@yahoogroups.com, February 20, 2003.

Cohn, M. (2004). User stories applied: For agile software development.

How and Why Will the Feature Be Used?

- SRS- lists is that the items on the lists describe the behavior of the software, not the behavior or goals of a user. ***A requirements list rarely answers "But how ill someone use this feature?***"

- Steve Berczuk, XP developer viewpoint Configuration Managem user stories"! importance of this question: "I es that I saved a whole lot of and asking a custom WHAT ARE THE BENEFITS? those features. Often the cu ature really isn't needed, and tha building things that add value."[3]

This is one viewpoint
-from the "father of user stories"!

[3] Steve B ming@yahoogroups.com, February 20, 2003.

Cohn, M. (2004). Use d. For agile software development.

Use Cases

A generalized description of a set of interactions between the system and one or more actors, where an actor is either a user or another system.

- Commonly Associated with RUP
- Freeform text or template
- First introduced by Ivor Jacobsen in 1992, templates introduced by Alistair Cockburn in 2001
- prioritising
- checking understanding

Cohn, M. (2004). User stories applied: For agile software development.

Use Cases

A generalized description of a set of interactions between the system and one or more actors, where an actor is either a user or another system.

- Commonly Associated with RUP
- Freeform text or template
- First introduced by Ivor Jacobsen in 1992, templates introduced by Alistair Cockburn in 2001
- prioritising
- checking understanding

Use Case Title: Pay for a job posting
Primary Actor: Recruiter
Level: Actor goal
Precondition: The job information has been entered but is not viewable.
Minimal Guarantees: None
Success Guarantees: Job is posted; recruiter's credit card is charged.
Main Success Scenario:

1. Recruiter submits credit card number, date, and authentication information.
2. System validates credit card.
3. System charges credit card full amount.
4. Job posting is made viewable to Job Seekers.
5. Recruiter is given a unique confirmation number.

Extensions:

2a: The card is not of a type accepted by the system:
2a1: The system notifies the user to use a different card.
2b: The card is expired:
2b1: The system notifies the user to use a different card.
2c: The card is expired:
2c1: The system notifies the user to use a different card.
3a: The card has insufficient available credit to post the ad.
3a1: The system charges as much as it can to the current credit card.
3a2: The user is told about the problem and asked to enter a second credit card for the remaining charge. The use case continues at Step 2.

Cohn, M. (2004). User stories applied: For agile software deve

Use Cases

- Extensions are usually error handling or alternative paths
- Larger scope than user story
- Main success criteria are like the user stories
- User stories plus acceptance tests very similar to use case
- Use cases are usually kept for the lifetime of the product - different to user stories

Use Case Title: Pay for a job posting
Primary Actor: Recruiter
Level: Actor goal
Precondition: The job information has been entered but is not viewable.
Minimal Guarantees: None
Success Guarantees: Job is posted; recruiter's credit card is charged.
Main Success Scenario:

1. Recruiter submits credit card number, date, and authentication information.
2. System validates credit card.
3. System charges credit card full amount.
4. Job posting is made viewable to Job Seekers.
5. Recruiter is given a unique confirmation number.

Extensions:

2a: The card is not of a type accepted by the system:
2a1: The system notifies the user to use a different card.
2b: The card is expired:
2b1: The system notifies the user to use a different card.
2c: The card is expired:
2c1: The system notifies the user to use a different card.
3a: The card has insufficient available credit to post the ad.
3a1: The system charges as much as it can to the current credit card.
3a2: The user is told about the problem and asked to enter a second credit card for the remaining charge. The use case continues at Step 2.

Cohn, M. (2004). User stories applied: For agile software development.

Essential Use Cases

Use Case Title: Compose and send email message

Main Success Scenario:

1. User selects the "New Message" menu item.
2. System presents the user with the "Compose New Message" dialog.
3. User edits email body, subject field and recipient lines.
4. User clicks the Send button.
5. System sends the message.

Cohn, M. (2004). User stories applied: For agile software development.

Essential Use Cases

Use Case Title: Compose and send email message

Main Success Scenario:

1. User selects the "New Message" menu item.
2. System presents the user with the "Compose New Message" dialog.
3. User edits email body, subject field and recipient lines.
4. User clicks the Send button.
5. System sends the message.

What's wrong with this?

Cohn, M. (2004). User stories applied: For agile software development.

Essential Use Cases

Use Case Title: Compose and send email message

Main Success Scenario:

1. User selects the "New Message" menu item.
2. System presents the user with the "Compose New Message" dialog.
3. User edits email body, subject field and recipient lines.
4. User clicks the Send button.
5. System sends the message.

What would the User Story be?

Cohn, M. (2004). User stories applied: For agile software development.

Essential Use Cases

Use Case Title: Compose and send email message

Main Success Scenario:

1. User selects the "New Message" menu item.
2. System presents the user with the "Compose New Message" dialog.
3. User edits email body, subject field and recipient lines.
4. User clicks the Send button.
5. System sends the message.

What would the User Story be?

User Story:
As a user I want to compose and send email messages.

Cohn, M. (2004). User stories applied: For agile software development.

Essential Use Cases

Use Case Title: Compose and send email message

Main Success Scenario:

1. User selects the "New Message" menu item.
2. System presents the user with the "Compose New Message" dialog.
3. User edits email body, subject field and recipient lines.
4. User clicks the Send button.
5. System sends the message.

What would the User Story be?

User Story:
As a user I want to compose and send email messages.

An essential use case (Constantine and Lockwood 1999), sometimes called a business use case, is a **simplified**, abstract, generalized use case that captures the **intentions of a user** in a **technology and implementation independent** manner. A fully documented essential use case is a structured narrative, **expressed in the language of the application domain and of users**, comprising a simplified, abstract, technology-free and implementation-independent description of one task or interaction. An essential use case is complete, meaningful, and well designed from the point-of-view of users in some role or roles in relation to a system and that embodies the purpose or intentions underlying the interaction.

http://agilemodeling.com/artifacts/essentialUseCase.htm

Cohn, M. (2004). User stories applied: For agile software development.

Essential Use Cases

Use Case Title: Compose and send email message

Main Success Scenario:

1. User selects the "New Message" menu item.
2. System presents the user with the "Compose New Message" dialog.
3. User edits email body, subject field and recipient lines.
4. User clicks the Send button.
5. System sends the message.

What would the User Story be?

User Story:
As a user I want to compose and send email messages.

User Intention	System Responsibility
Compose email message	
Indicate recipient(s)	
	Collect email content and recipient(s)
Send email message	
	Send the message

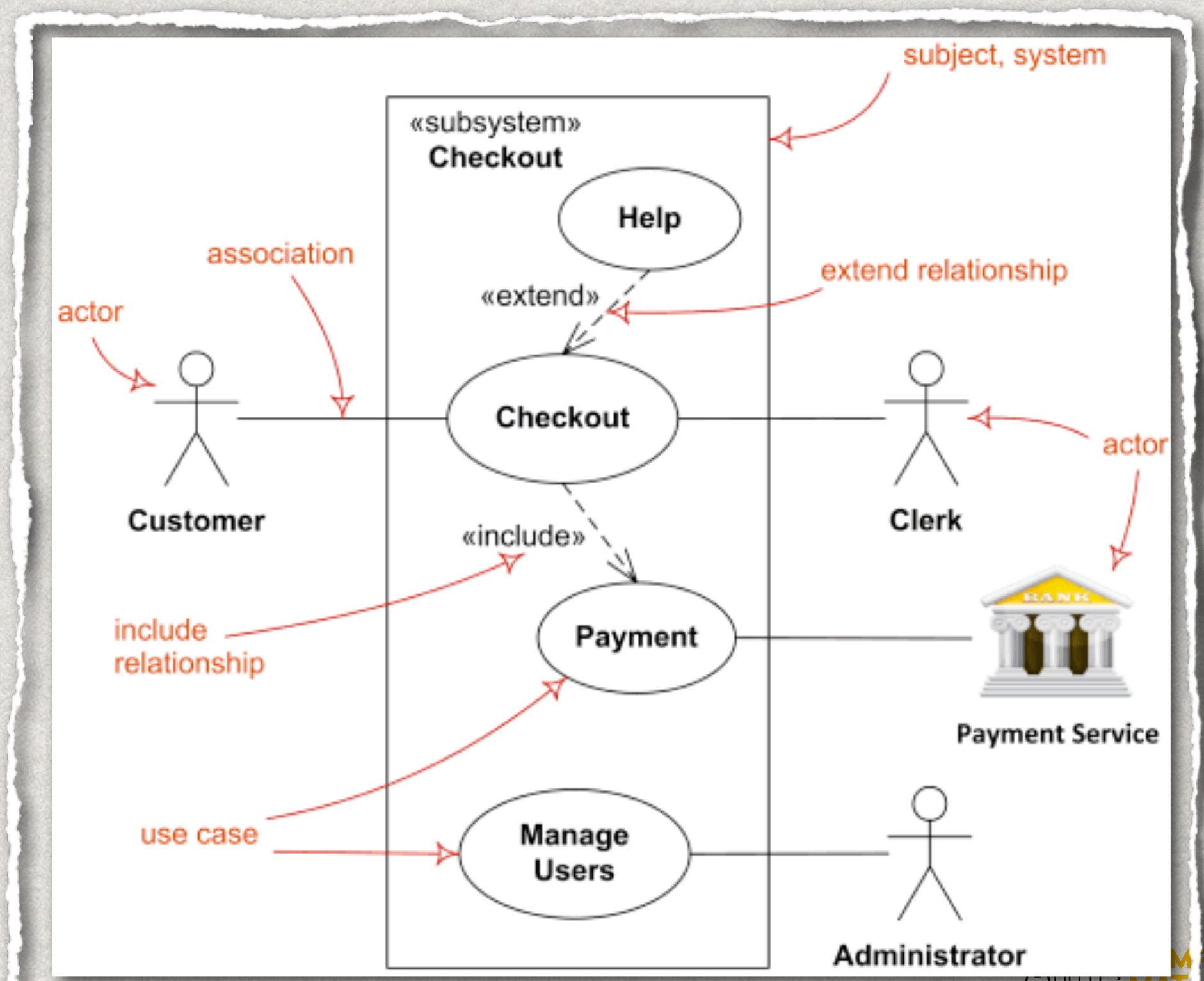
User stories?

An essential use case (Constantine and Lockwood 1999), sometimes called a business use case, is a **simplified**, abstract, generalized use case that captures the **intentions of a user** in a **technology and implementation independent** manner. A fully documented essential use case is a structured narrative, **expressed in the language of the application domain and of users**, comprising a simplified, abstract, technology-free and implementation-independent description of one task or interaction. An essential use case is complete, meaningful, and well designed from the point-of-view of users in some role or roles in relation to a system and that embodies the purpose or intentions underlying the interaction.

http://agilemodeling.com/artifacts/essentialUseCase.htm

Cohn, M. (2004). User stories applied: For agile software development.

Use Case Diagrams



Scenarios

Maria is thinking about making a career change. Since the glory days of the dot-com boom she has worked as a tester at BigTechCo. A former high school math teacher, Maria decides she'll be happier if she returns to teaching. Maria goes to the BigMoneyJobs.com website. She creates a new account with a user name and password. She then creates her resume. She wants to find a job as a math teacher anywhere in Auckland but preferably near her current job in Albany. Maria finds a handful of jobs that match her search criteria. The job that intrigues her most is with the North Shore School, a private high school in Browns Bay. Maria has a friend, Jessica, in Browns Bay whom she hopes may know someone at North Shore. Maria enters Jessica's email address and forwards the job link to her with a note asking if she knows anyone at the school. The next morning Maria gets an email from Jessica saying that she doesn't know anyone at the school, but she knows of the North Shore School and it has a wonderful reputation. Maria clicks on a button that submits her resume to North Shore.

Scenarios

List some user stories.

Maria is thinking about making a career change. Since the glory days of the dot-com boom she has worked as a tester at BigTechCo. A former high school math teacher, Maria decides she'll be happier if she returns to teaching. Maria goes to the BigMoneyJobs.com website. She creates a new account with a user name and password. She then creates her resume. She wants to find a job as a math teacher anywhere in Auckland but preferably near her current job in Albany. Maria finds a handful of jobs that match her search criteria. The job that intrigues her most is with the North Shore School, a private high school in Browns Bay. Maria has a friend, Jessica, in Browns Bay whom she hopes may know someone at North Shore. Maria enters Jessica's email address and forwards the job link to her with a note asking if she knows anyone at the school. The next morning Maria gets an email from Jessica saying that she doesn't know anyone at the school, but she knows of the North Shore School and it has a wonderful reputation. Maria clicks on a button that submits her resume to North Shore.

Scenarios

List some user stories.

Maria is thinking about making a career change. Since the glory days of the dot-com boom she has worked as a tester at BigTechCo. A former high school mat happier if she returns to teaching. Mar m website. She creates a new ac sword. She then creates her resu math teacher anywhere in Auc rent job in Albany. Maria finds a ha h criteria. The job that intrigues her mo , a private high school in Browns Bay. M wns Bay whom she hopes may know enters Jessica's email address and for te asking if she knows anyone at the sc s an email from Jessica saying t e school, but she knows of the North Shore School and it has a wonderful reputation. Maria clicks on a button that submits her resume to North Shore.

A Scenario has...

- a setting
- actors
- goals or objectives
- actions and events

AUT COMPUTING + MATHEMATICAL SCIENCE

User Stories

- Card
- Conversation
- Confirmation

As a <role>,
I can <activity>
So that <business value>

Details in discussion between PO and team

- *A list of what will make the story acceptable to the product owner*

User Stories

Conversation

- Prior to the iteration
- During the Iteration planning meeting
- During development

As a <role>,
I can <activity>
So that <business value>

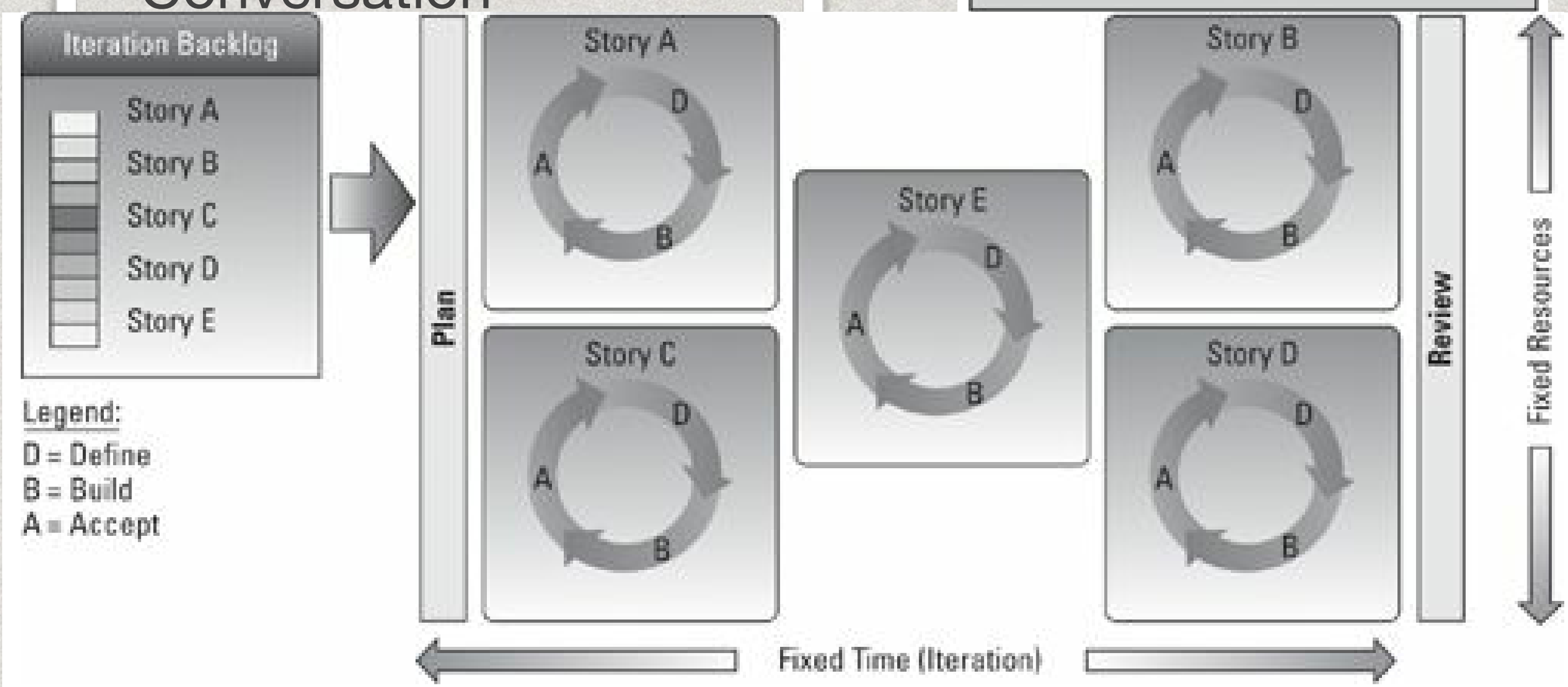
Details in discussion between PO and team

- *A list of what will make the story acceptable to the product owner*

User Stories

Conversation

As a <role>,
I can <activity>
So that <business value>



Which are ok or not?

- The user can run the system on Windows XP and Linux.
- All graphing and charting will be done using a third-party library.
- The user can undo up to fifty commands.
- The software will be released by June 30.
- The software will be written in Java.
- The user can select her country from a drop-down list.
- The system will use Log4J to log all error messages to a file.
- The user will be prompted to save her work if she hasn't saved it for 15 minutes.
- The user can select an "Export to XML" feature.
- The user can export data to XML.

As a <role>,
I can <activity>
So that <business value>

Details in discussion between PO and team

- *A list of what will make the story acceptable to the product owner*

AUT COMPUTING + MATHEMATICAL SCIENCES

Why User Stories

- User stories emphasize verbal communication.
- User stories are comprehensible by everyone.
- User stories are the right size for planning.
- User stories work for iterative development.
- User stories encourage deferring detail.
- User stories support opportunistic design.
- User stories encourage participatory design.
- User stories build up tacit knowledge.

As a <role>,
I can <activity>
So that <business value>

Details in discussion between PO and team

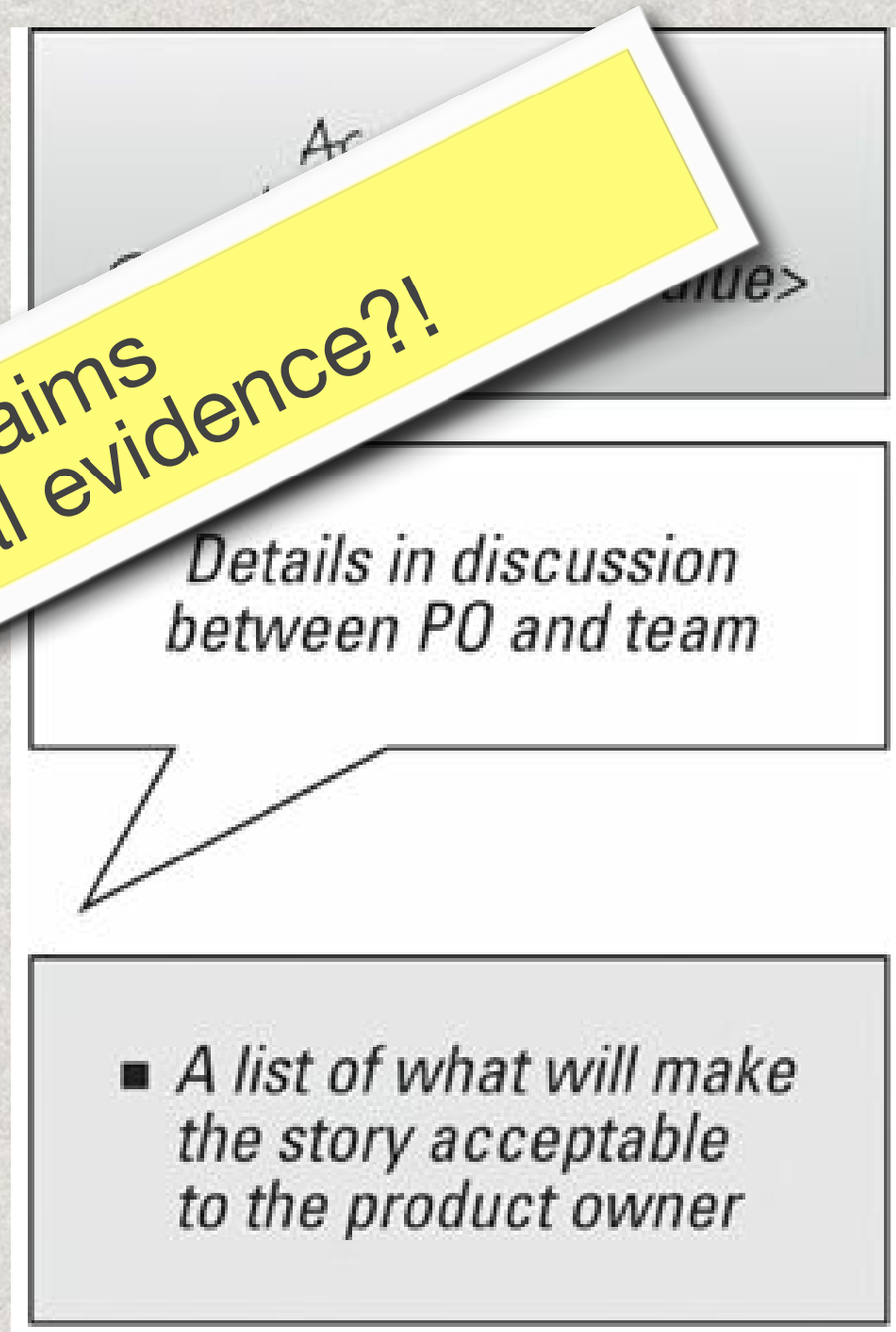
- *A list of what will make the story acceptable to the product owner*

AUT COMPUTING + MATHEMATICAL SCIENCES

Why User Stories

- User stories emphasize verbal communication.
- User stories are comprehensible by everyone.
- User stories are the right size for planning.
- User stories work for iterati
  development.
- User stories detail.
- User st portunistic
  design.
- User stories encourage participatory design.
- User stories build up tacit knowledge.

These are the claims
-where is the empirical evidence?!



Characteristics of Good User Stories

- **I**ndependent
- **N**egotiable
- **V**aluable to the users or customers
- **E**stimable
- **S**mall
- **T**estable

Characteristics of Good User Stories

INDEPENDENT

- Dependency can cause prioritisation and planning problems
  - A company can pay for a job posting with a Visa card.
  - A company can pay for a job posting with a MasterCard.
  - A company can pay for a job posting with an American Express card.

Characteristics of Good User Stories

INDEPENDENT

- Dependency can cause prioritisa
planning problems
  - A company can pay for a job posting with a Visa card.
  - A company can pay for a job posting with a MasterCard.
  - A company can pay for a job posting with an American Express card.

- A customer can pay with one type of credit card.
- A customer can pay with two additional types of credit cards.

Characteristics of Good User Stories

- Independent

NEGOTIABLE

- A User Story is NOT a contract

Characteristics of Good User Stories

VALUABLE

- Why is the system being proposed
- What is the economic, social or legal value and to whom?

- Independent
- Negotiable

Characteristics of Good User Stories

ESTIMABLE

What if not estimable?

Challenges estimating when-

- Developers lack domain knowledge.
- Developers lack technical knowledge.
- The story is too big.

- Independent
- Negotiable
- Valuable

Characteristics of Good User Stories

SMALL

- Will take 1 day (or <2 weeks) to implement-IT IS CONSTRAINED
- Can be too small?
- What if it's very big
- Are epic user stories ok?

- Independent
- Negotiable
- Valuable
- Estimable

AUT COMPUTING + MATHEMATICAL SCIENCES

Characteristics of Good User Stories

TESTABLE

- What if not testable
- Acceptance tests, success criteria, specification by examples etc..

- Independent
- Negotiable
- Valuable
- Estimable
- Small

Characteristics of Good User Stories

TESTABLE

- What if not test
- Accep
  s
  uccess criteria,
  y examples etc..

You need to understand INVEST so it becomes invisible!

- Independent
- Negotiable
- Valuable
- Estimable
- Small

User Story Smells

- Stories Are Too Small
- Interdependent Stories
- Goldplating
- Too Many Details
- Including User Interface Detail Too Soon
- Thinking Too Far Ahead
- Splitting Too Many Stories
- Customer Has Trouble Prioritizing
- Customer Won't Write and Prioritize the Stories

User Story Smells

- Stories Are Too Small
- Interdependent Stories
- Goldplating
- Too Many Details
- Including User Int Soon
- Thinking
- Sp any Stories
- Custo ner Has Trouble Prioritizing
- Customer Won't Write and Prioritize the Stories

See Cohn's book on User Stories, Chapter 14
(deliberately vague reference-see if you can find it)

Cohn's Hints

- To identify stories, start by considering the goals of each user role in using the system.
- When splitting a story, try to come up with stories that cut through all layers of the application.
- Try to write stories that are of a size where the user feels justified in taking a coffee break after completing the story.
- Augment stories with other requirements gathering or documenting techniques as necessary for the project's domain and environment.
- Create constraint cards and either tape them to a shared wall or write tests to ensure the constraints are not violated.
- Write smaller stories for functionality the team will implement soon, and write broad, high- level stories for functionality further into the future.
- Keep the user interface out of the stories for as long as possible.

Bertrand Meyer's View

User stories requirement elicitation but not a fundamental requirement technique. They cannot define the requirements:

- Not abstract enough
- Too specific
- Describe current processes

- Do not support evolution

User stories are to requirements what tests are to software specification

Major application: for validating requirements

BM’s example

Consider “busy treatments” in telephony, such as call forwarding, callee interruption, delayed retry and voice mail. Suppose that we have a tool for specifying and composing such features. Features can fail to come into action, even if their individual description says they should, when they start interacting with other features :

- Bob has enabled the “call-forwarding” feature, to Carol. Carol has “do-not- disturb”. Alice calls Bob: the call is forwarded to Carol; her phone rings.

- Alice calls a sales group. A sales-group feature forwards the call to the salesperson on duty, Bob. His cellphone is off, so the caller gets Bob’s personal Voice Mail message. It would be better to reactivate the sales-group feature to find another salesperson.

*Abridged, see full citation in his book or original at http://public.research.att.com/~pamela/faq.html

Breaking Up Epic or compound User Stories-some patterns

1. Workflow Steps

Identify specific steps that a user takes to accomplish a specific workflow, and then implement the workflow in incremental stages.

As a utility, I want to update and publish pricing programs to my customer.	...I can publish pricing programs to the customer's in-home display. ...I can send a message to the customer's web portal. ...I can publish the pricing table to a customer's smart thermostat.

2. Business Rule Variations

At first glance, some stories seem fairly simple. However, sometimes the business rules are more complex or extensive than the first glance revealed. In this case, it might be useful to break the story into several stories to handle the business rule complexity.

As a utility, I can sort customers by different demographics.	...sort by ZIP code. ...sort by home demographics. ...sort by energy consumption.

3. Major Effort

Sometimes a story can be split into several parts where most of the effort will go toward implementing the first one. In the example shown next, processing infrastructure should be built to support the first story; adding more functionality should be relatively trivial later.

As a user, I want to be able to select/change my pricing program with my utility through my web portal.	...I want to use time-of-use pricing. ...I want to prepay for my energy. ...I want to enroll in critical-peak pricing.

4. Simple/Complex

When the team is discussing a story and the story seems to be getting larger and larger ("What about x? Have you considered y?"), stop and ask, "What's the simplest version that can possibly work?" Capture that simple version as its own story, and then break out all the variations and complexities into their own stories.

As a user, I basically want a fixed price, but I also want to be notified of critical-peak pricing events.	...respond to the time and the duration of the critical-peak pricing event. ...respond to emergency events.

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed.). Addison-Wesley Professional. p135-136

Breaking Up Epic or compound User Stories-some patterns

5. Variations in Data

Data variations and data sources are another source of scope and complexity. Consider adding stories just-in-time after building the simplest version. A localization example is shown here:

As a utility, I can send messages to customers.	...customers who want their messages.
	...in Spanish
	...in Arabic, and so on.

6. Data Entry Methods

Sometimes complexity is in the user interface rather than the functionality itself. In that case, split the story to build it with the simplest possible UI, and then build the richer UI later.

As a user, I can view my energy consumption in various graphs.	...using bar charts that compare weekly consumption.
	...in a comparison chart, so I can compare my usage to those who have the same or similar household demographics.

7. Defer System Qualities

Sometimes, the initial implementation isn't all that hard, and the major part of the effort is in making it fast or reliable or more precise or more scalable. However, the team can learn a lot from the base implementation, and it should have some value to a user, who wouldn't otherwise be able to do it all. In this case, break the story into successive "ilities."

As a user, I want to see real-time consumption from my meter.	...interpolate data from the last known reading.
	...display real-time data from the meter.

Breaking Up Epic or compound User Stories-some patterns

8. Operations (Example: Create Read Update Delete (CRUD))

Words like *manage* or *control* are a giveaway that the story covers multiple operations, which can offer a natural way to split the story.

As a user, I can manage my account.

...I can sign up for an account.

...I can edit my account settings.

...I can cancel my account.

...I can add more devices to my account.

9. Use-Case Scenarios

If use cases have been developed to represent complex user-to-system or system-to-system interaction, then the story can often be split according to the individual scenarios of the use case.*

I want to enroll in the energy savings program through a retail distributor.

Use case/story #1 (happy path): Notify utility that consumer has equipment.

Use case/story #2: Utility provisions equipment and data and notifies consumer.

Use case/story #3 (alternate scenario): Handle data validation errors.

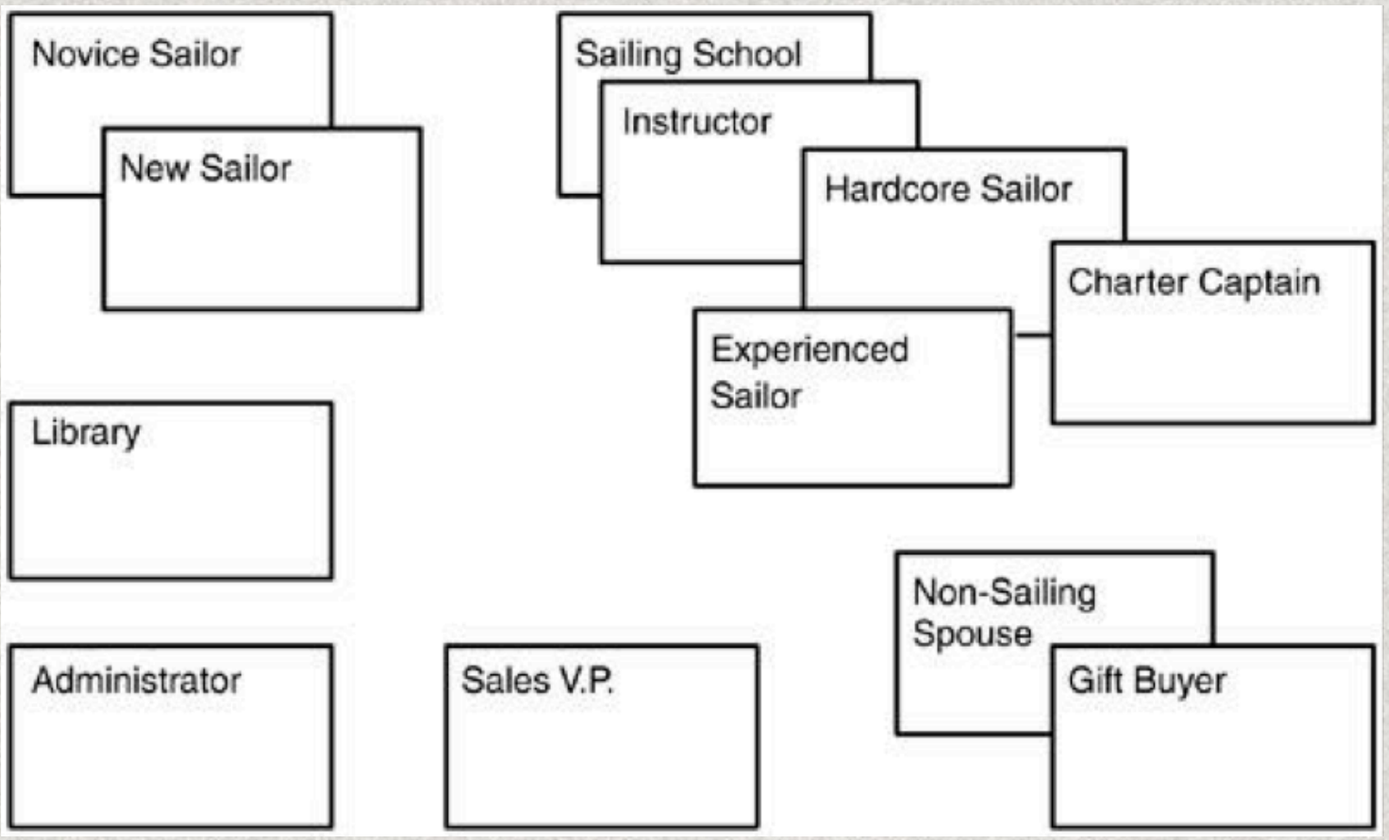
10. Break Out a Spike

In some cases, a story may be too large or overly complex, or perhaps the implementation is poorly understood. In that case, build a technical or functional spike to figure it out; then split the stories based on that result. (See the "Spikes" section.)

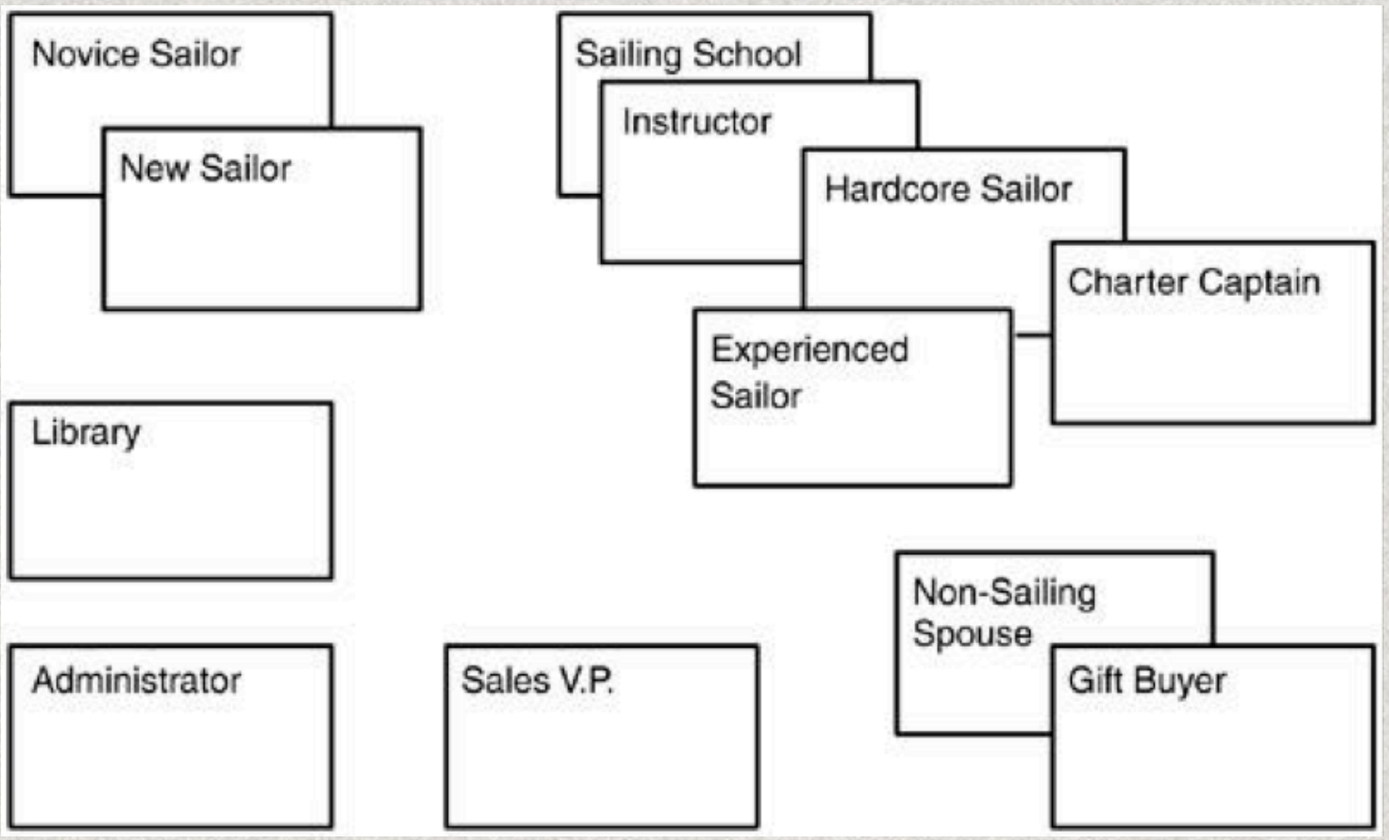
Example: 5 mins

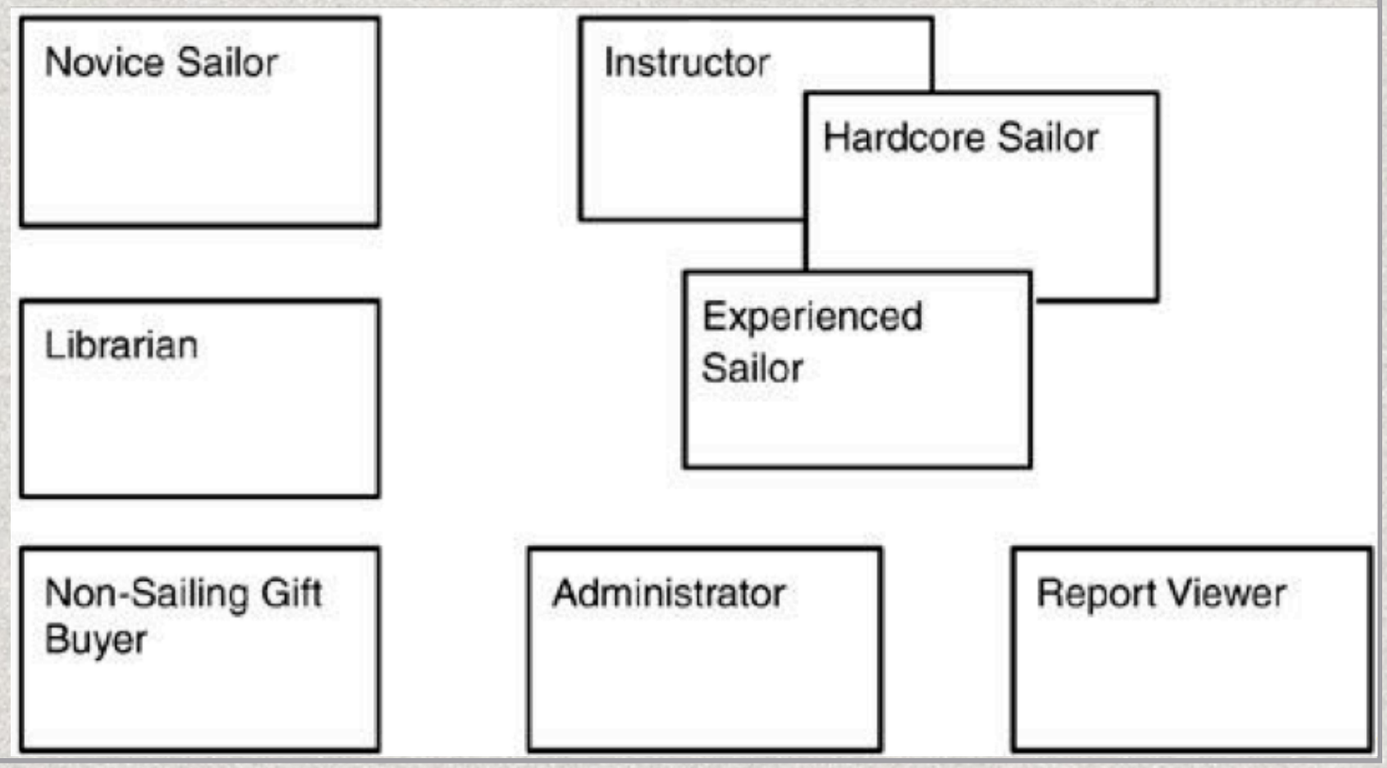
- An eCommerce system selling books and equipment related to sailing online.
- Write a description of 2 users
- Write 2 user stories

Roles


Novice Sailor
New Sailor
Sailing School
Instructor
Hardcore Sailor
Charter Captain
Experienced Sailor
Library
Non-Sailing Spouse
Gift Buyer
Administrator
Sales V.P.

Roles


Novice Sailor
New Sailor
Sailing School
Instructor
Hardcore Sailor
Charter Captain
Experienced Sailor
Library
Non-Sailing Spouse
Gift Buyer
Administrator
Sales V.P.


Novice Sailor
Instructor
Hardcore Sailor
Experienced Sailor
Librarian
Non-Sailing Gift Buyer
Administrator
Report Viewer

Role Details

- **Novice Sailor**: Experienced web shopper. Expected to make 6 purchases during first 3 months of sailing. Is sometimes referred to a specific title; other times needs help selecting the right book. Wants more help in selecting appropriate books (good content written at the appropriate level) than she gets in a physical bookstore.

- **Instructor**: Expected to use the website frequently, often once a week. Through the company's telephone sales group, an Instructor frequently places similar orders (for example, 20 copies of the same book). Proficient with the website but usually somewhat nervous around computers. Interested in getting the best prices. Not interested in reviews or other "frills."

- **Hard Core Sailor**: Generally inexperienced with computers. A large purchaser of items from the company's catalog but not a large purchaser of books. Buys lots of additional equipment from us. Usually knows exactly what he's looking for. Doesn't want to feel stupid while trying to use the site.

- **Experienced Sailor**: Proficient with computers. Expected to order once or twice per quarter, perhaps more often during the summer. Knowledgeable about sailing but usually only for local regions. Very interested in what other sailors say are the best products and the best places to sail.

- **Non-Sailing Gift Buyer**: Usually proficient with computers (or would not chose to purchase a gift online). Not a sailor and will have passing familiarity at best with sailing terms. Will usually be after a very specific book but may be looking for a book on a given topic. Librarian: Proficient with computers. Knows exactly what she is looking for and prefers to order by ISBN rather than author or even title. Not particularly interested in frills such as gift wrapping or tracking shipments. Will generally place a small number of orders per year but each order will be for more books than a typical individual order.

- **Administrator**: Highly proficient with computers. Has at least a passing familiarity with sailing. Accesses the back-end of the system daily as part of her job. Interested in quickly learning the software but will want advanced user shortcuts later.
  Report Viewer: Moderate proficiency with computers, mostly with business programs such as spreadsheets and word processors. Interested in highly detailed data on how the system is working, what visitors are buying or not buying, and how they are navigating or searching the site. Very willing to trade speed for power and depth.

Personas

Teresa has been sailing for four years. She is the CEO of a publicly-traded biotech company and is completely comfortable ordering online. Teresa sails mainly during the summer so she'll only use the site during spring or summer, when preparing for a cruise. She's very busy and is interested in using our site to save time and to find books she hasn't seen before. Teresa is married to Tom, who doesn't sail himself but has accompanied Teresa on on two cruises through the Mediterranean.

Personas

Teresa has been sailing for four years. She is the CEO of a publicly-traded biotech company and is completely comfortable ordering online. Teresa sails mainly during the summer so she'll only use the site during spring or summer, when preparing for a cruise. She's very busy and is interested in using our site to save time and to find books she hasn't seen before. Teresa is married to Tom, who doesn't sail himself but has accompanied Teresa on on two cruises through the Mediterranean.

Captain Ron has been sailing for 40 years and runs a sailing school out of San Diego. He retired from teaching high school five years ago and has been a sailing instructor ever since. He's been a loyal catalog customer for ten years. He's still a little intimidated by the computer in his office but he's intrigued enough by ordering on the web that we expect him to give it a try.