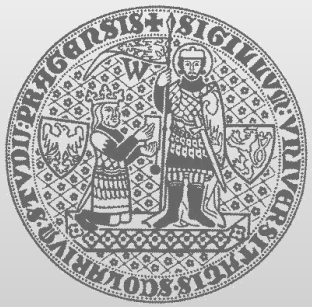
Requirements Engineering 2

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Software Engineering for Dependable Systems

Tomas Bures

bures@d3s.mff.cuni.cz

Racap

Types of 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 so may be part of a contract between client and contractor.

User and Systems Requirements

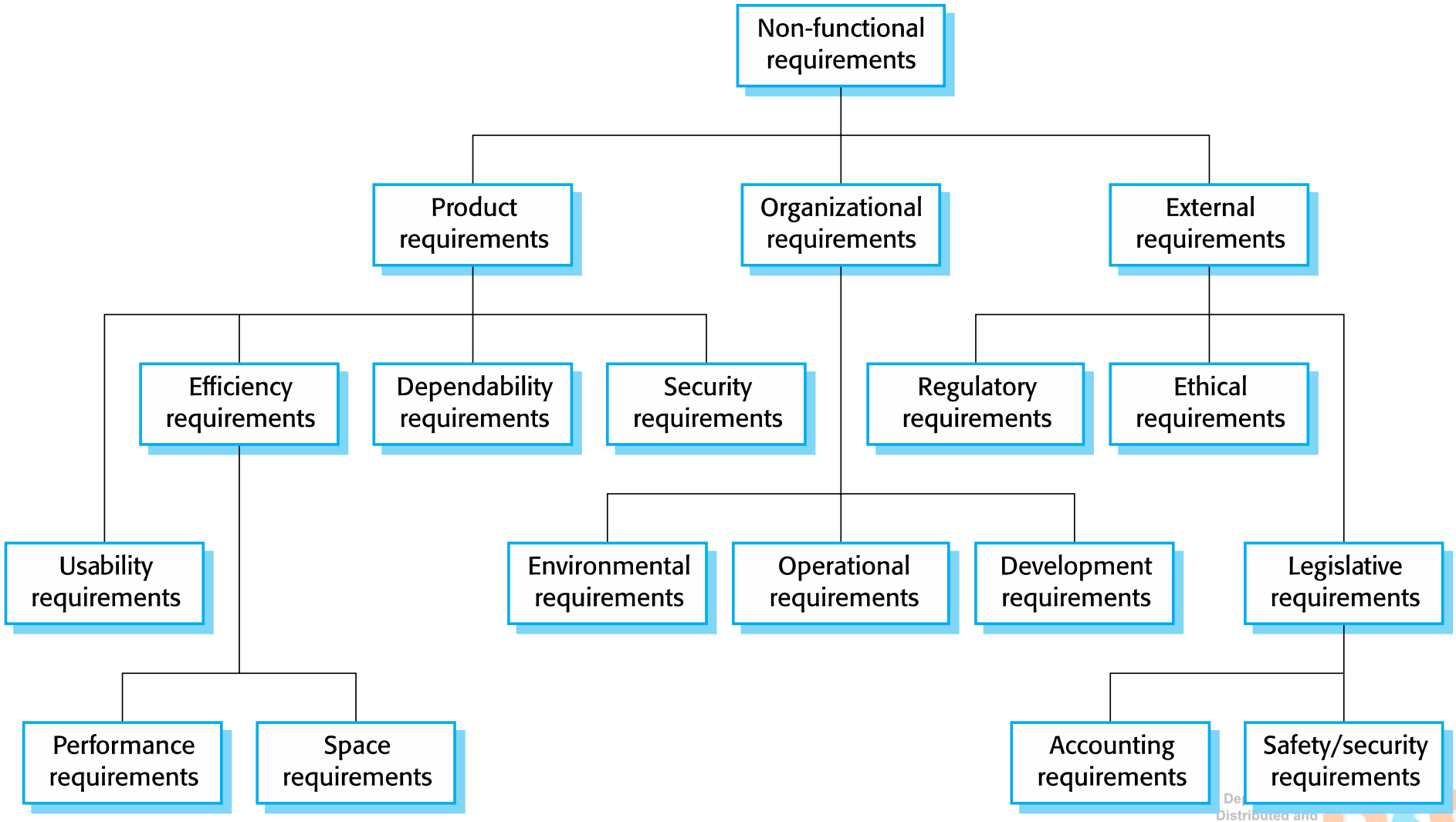
User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Types of non-functional requirements



Examples of nonfunctional requirements

Product requirement

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement

Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement

The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Requirements engineering processes

- Requirements elicitation & analysis
  - Requirements discovery
  - Requirements classification and organization
  - Requirements prioritization and negotiation
  - Requirements specification
- Requirements validation
  - Ensures
    - Validity – does the system provide the functions which best support the customer's needs?
    - Consistency – are there any requirements conflicts?
    - Completeness – are all functions required by the customer included?
    - Realism – can the requirements be implemented given available budget and technology
    - Verifiability – can the requirements be checked?
  - By requirement reviews, prototyping, test-case generation
- Requirements management

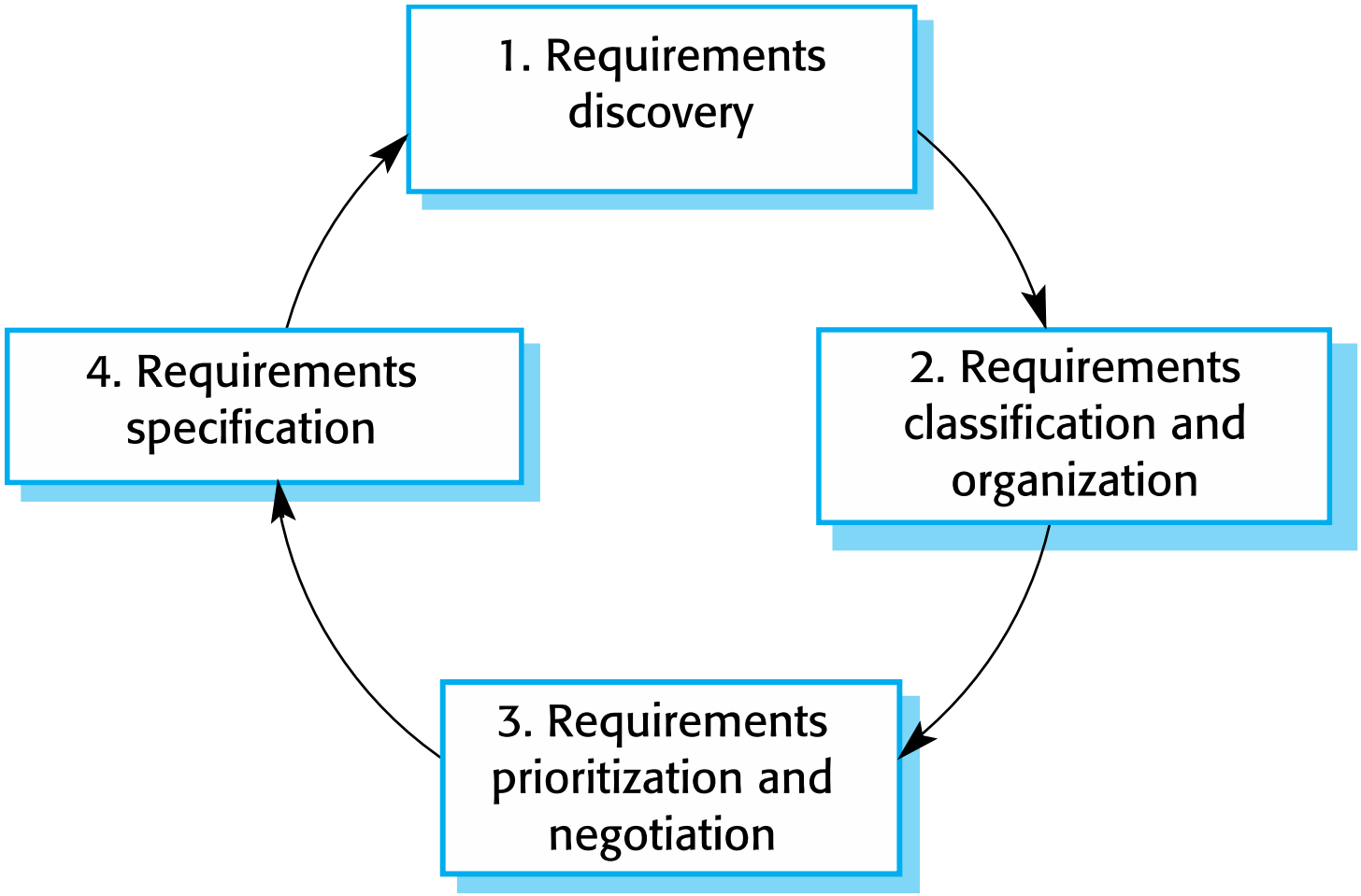
Requirements Elicitation

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

Problems of requirements elicitation

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

Requirements elicitation and analysis process



- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
  - Requirements are documented and input into the next round of the spiral.

Requirements Discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Interaction is with system stakeholders from managers to external regulators.
- Systems normally have a range of stakeholders.

Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Interviews in Practice

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- You need to prompt the user to talk about the system by suggesting requirements rather than simply asking them what they want.

Stories and Scenarios

- Scenarios and user stories are real-life examples of how a system can be used.
- Stories and scenarios are a description of how a system may be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Photo sharing in the classroom (iLearn)

Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, pupils are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo sharing site as he wants pupils to take and comment on each others' photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers group, which he is a member of to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

Scenarios

- A structured form of user story
- Scenarios should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Example: iLearn

- **Initial assumption:** A user or a group of users have one or more digital photographs to be uploaded to the picture sharing site. These are saved on either a tablet or laptop computer. They have successfully logged on to KidsTakePics.
- **Normal:** The user chooses upload photos and they are prompted to select the photos to be uploaded on their computer and to select the project name under which the photos will be stored. They should also be given the option of inputting keywords that should be associated with each uploaded photo. Uploaded photos are named by creating a conjunction of the user name with the filename of the photo on the local computer.
- On completion of the upload, the system automatically sends an email to the project moderator asking them to check new content and generates an on-screen message to the user that this has been done.

Example: iLearn

- **What can go wrong:**
- No moderator is associated with the selected project. An email is automatically generated to the school administrator asking them to nominate a project moderator. Users should be informed that there could be a delay in making their photos visible.
- Photos with the same name have already been uploaded by the same user. The user should be asked if they wish to re-upload the photos with the same name, rename the photos or cancel the upload. If they chose to re-upload the photos, the originals are overwritten. If they chose to rename the photos, a new name is automatically generated by adding a number to the existing file name.

- **Other activities:** The moderator may be logged on to the system and may approve photos as they are uploaded.

- **System state on completion:** User is logged on. The selected photos have been uploaded and assigned a status 'awaiting moderation'. Photos are visible to the moderator and to the user who uploaded them.

Requirements Specification

Ways of writing a system req. specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Requirements and design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
  - This may be the consequence of a regulatory requirement.

Natural language specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
- Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Guidelines for writing requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.
- Include an explanation (rationale) of why a requirement is necessary.

Problems with natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read.
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up.
- Requirements amalgamation
  - Several different requirements may be expressed together.

Example – Insulin pump (natural lang.)

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

Structured specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Form-based specifications

- Definition of the function or entity.
- Description of inputs and where they come from.
- Description of outputs and where they go to.
- Information about the information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

Example – Insulin pump (structured)

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2); the previous two readings (r0 and r1).

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

Action

CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requirements

Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition

The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition r0 is replaced by r1 then r1 is replaced by r2.

Side effects None.

Tabular specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Example – Insulin pump (tabular)

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

Use-cases

- Requirements that capture interaction with a user can be described by use-cases
- A use case is a collection of related success and failure scenarios that describe actors using a system to support a goal
- A scenario is a specific sequence of actions and interactions between actors and the system under discussion
  - also called a use case instance.
  - a particular story of using a system, or one path through the use case
  - Examples:
    - successfully purchasing items with cash
    - failing to purchase items because of a credit card transaction denial
- A use-case describes interaction of actors (e.g. a person – identified by role, computer system, or organization) and is supposed to yield an observable result of value to a particular actor
- Different level of detail – brief, casual, fully-dressed

Use-case Diagrams

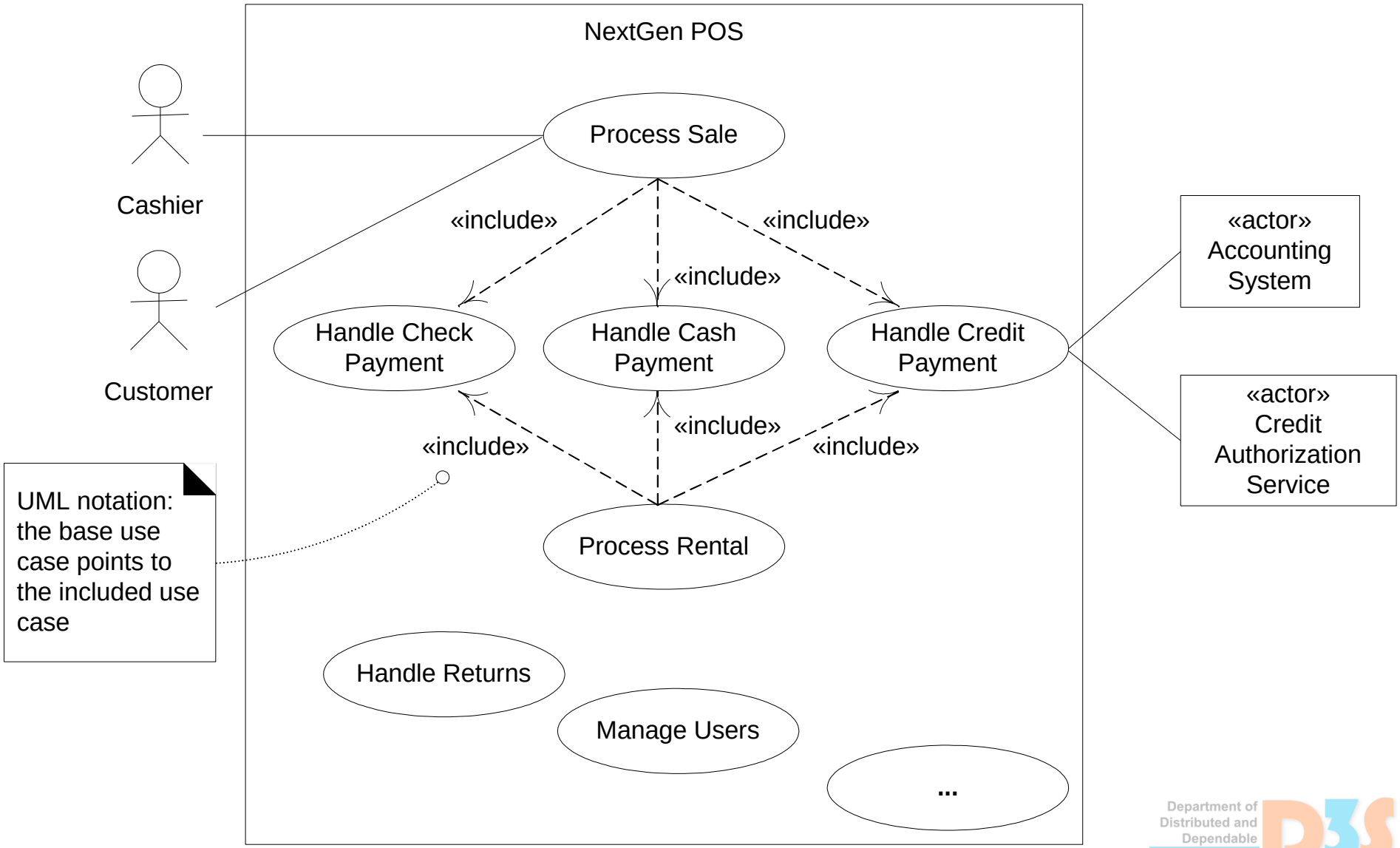


Figure taken from C. Larman: Applying UML and Patterns (Second Edition), Prentice Hall

Example of a *Brief Format* Use Case

Process Sale:

A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

Example of *Casual Format* Use-Case

Handle Returns

Main Success Scenario: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item ...

Alternate Scenarios:

If the credit authorization is reject, inform the customer and ask for an alternate payment method.

If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).

If the system detects failure to communicate with the external tax calculator system, ...

Fully Dressed Example 1/9

www.usecases.org format

Primary Actor: Cashier

Stakeholders and Interests:

- *Cashier*: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.
- *Salesperson*: Wants sales commissions updated.
- *Customer*: Wants purchase and fast service with minimal effort. Wants proof of purchase to support returns.
- *Company*: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
- *Government Tax Agencies*: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.
- *Payment Authorization Service*: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.

Fully Dressed Example 1/9

www.usecases.org format

Primary Actor: Cashier

Stakeholders and Interests:

- *Cashier*: Wants accurate, fast entry
  shortages are deducted from his/h
- *Salesperson*: Wants sales commissions updated.
- *Customer*: Wants purchase and fast service with minimal effort. Wants proof of purchase to support returns.
- *Company*: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
- *Government Tax Agencies*: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.
- *Payment Authorization Service*: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.

Primary Actor

The principal actor that calls upon system services to fulfill a goal

Department of Distributed and Dependable Systems D3S

Fully Dressed Example 1/9

www.usecases.org format

Primary Actor: Cashier

Stakeholders and Interests:

- *Cashier*: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.
- *Salesperson*: Wants sales commissions updated.
- *Customer*: Wants purchase and fast service with minimal effort. Wants proof of purchase to support returns.
- *Company*: Wants
  interests. Wants t
  receivables are re
  even if server con
  Wants automatic
- *Government Tax A*
  agencies, such as
- *Payment Authoriz*
  in the correct format and protocol. Wants to accurately account for their payables to the store.

Stakeholders and Interests

Suggests and bounds what the system must do

The [system] operates a contract between stakeholders, with the use cases detailing the behavioral parts of that contract... The use case, as the contract for behavior, captures all and only the behaviors related to satisfying the stakeholders' interests *[Cockburn0l]*

Department of Distributed and Dependable Systems D3S

Fully Dressed Example 2/9

www.usecases.org format

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.
Payment authorization approvals are recorded.

Main Success Scenario (or Basic Flow):

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.

Fully Dressed Example 2/9

www.usecases.org format

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.
Payment authorizati

Main Success Scenar

1. Customer arrive
2. Cashier starts a
3. Cashier enters it
4. System records s
   total. Price calcu

Cashier repeats steps

5. System presents
6. Cashier tells Cus
7. Customer pays a

Preconditions

State what must always be true before beginning a scenario in the use case.

Not tested in the use-case, rather assumed to be true.

Typically, a precondition implies a scenario of another use case that has successfully completed, such as logging in, or the more general "cashier is identified and authenticated."

Note that there are conditions that must be true, but are not of practical value to write, such as "the system has power." Preconditions communicate noteworthy assumptions that the use case writer thinks readers should be alerted to.

Fully Dressed Example 2/9

www.usecases.org format

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.
Payment authorizati

Success guarantees (Postconditions)

State what must be true on successful completion of the use case – either the main success scenario or some alternate path.

Main Success Scenar

1. Customer arrive
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.

Department of Distributed and Dependable Systems D3S

Fully Dressed Example 2/9

www.usecases.org format

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.
Payment authorization approvals are recorded.

Main Success Scenario (or Basic Flow):

1. Customer arrives
2. Cashier starts a
3. Cashier enters it
4. System records s
   total. Price calcu

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.

Main Success Scenario

Basic flow. Typical success path that satisfies the interests of the stakeholders.

Often does *not* include any conditions or branching – deferred to Extensions section.

Department of Distributed and Dependable Systems D3S

Fully Dressed Example 2/9

www.usecases.org format

Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.
Payment authorization approvals are recorded.

Main Success Scenario (or Basic Flow):

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a
3. Cashier enters it
4. System records s
   total. Price calcu

Cashier repeats steps

5. System presents
6. Cashier tells Cus
7. Customer pays a

Scenario

The scenario records the steps, of which there are three kinds:

1. An interaction between actors
2. A validation (usually by the system)
3. A state change by the system (for example, recording or modifying something).

Step one of a use case does not always fall into this classification, but indicates the trigger event that starts the scenario.

Fully Dressed Example 3/9

www.usecases.org format

8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

*a. At any time, System fails:

To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.

1. Cashier restarts System, logs in, and requests recovery of prior state.
2. System reconstructs prior state.

2a. System detects anomalies preventing recovery:

1. System signals error to the Cashier, records the error, and enters a clean state.
2. Cashier starts a new sale.

Fully Dressed Example 3/9

www.usecases.org format

8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

*a. At any time, System fails:

To support recov
state and eve
1. Cashier restar
2. System recon
2a. System detects anomalies preventing recovery:
1. System signals error to the Cashier, records the error, and enters a clean state.
2. Cashier starts a new sale.

Extensions (Alternate Flows)

Indicate all the other scenarios or branches, both success and failure.

Department of Distributed and Dependable Systems D3S

Fully Dressed Example 4/9

www.usecases.org format

3a. Invalid identifier:
    1. System signals error and rejects entry.

3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
    1. Cashier can enter item category identifier and the quantity.

3-6a. Customer asks Cashier to remove an item from the purchase:
    1. Cashier enters item identifier for removal from sale.
    2. System displays updated running total.

3-6b. Customer tells Cashier to cancel sale:
    1. Cashier cancels sale on System.

3-6c. Cashier suspends the sale:
    1. System records sale so that it is available for retrieval on any POS terminal.

4a. The system generated item price is not wanted (e.g., Customer complained about something and is offered a lower price):
    1. Cashier enters override price.
    2. System presents new price.

Fully Dressed Example 4/9

www.usecases.org format

3a. Invalid identifier:
    1. System signals error and rejects entry.
3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
    1. Cashier can enter item category identifier and the quantity.
3-6a. Customer asks Cashier to remove an item from the purchase:
    1. Cashier enters item identifier for removal from sale.
    2. System displ
3-6b. Customer tells C
    1. Cashier canc
3-6c. Cashier suspend
    1. System recor
4a. The system gene
about something
    1. Cashier ente
    2. System prese

Extensions (Alternate Flows)

Extension scenarios are branches from the main success scenario, and so can be notated with respect to it. For example, at Step 3 of the main success scenario there may be an invalid item identifier, either because it was incorrectly entered or unknown to the system. An extension is labeled "3a"; it first identifies the condition and then the response. Alternate extensions at Step 3 are labeled "3b“ and so forth.

Fully Dressed Example 4/9

www.usecases.org format

3a. Invalid identifier:
    1. System signals error and rejects entry.
3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
    1. Cashier can enter item category identifier and the quantity.
3-6a. Customer asks Cashier to remove an item from the purchase:
    1. Cashier enters item identifier for removal from sale.
    2. System displays updated running total
3-6b. Customer tells C
    1. Cashier canc
3-6c. Cashier suspend
    1. System recor
4a. The system generated item price is not wanted (e.g., Customer complained about something and is offered a lower price):
    1. Cashier enters override price.
    2. System presents new price.

Extensions (Alternate Flows)

Extension handling can be summarized in *one step*, or include *a sequence* (see 3-6a) which also illustrates notation to indicate that a condition can arise within a range of steps.

Fully Dressed Example 5/9

www.usecases.org format

5a. System detects failure to communicate with external tax calculation system service:
    1. System restarts the service on the POS node, and continues.
    1a. System detects that the service does not restart.
        1. System signals error.
        2. Cashier may manually calculate and enter the tax, or cancel the sale.

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):
    1. Cashier signals discount request.
    2. Cashier enters Customer identification.
    3. System presents discount total, based on discount rules.

5c. Customer says they have credit in their account, to apply to the sale:
    1. Cashier signals credit request.
    2. Cashier enters Customer identification.
    3. Systems applies credit up to price=0, and reduces remaining credit.

Fully Dressed Example 5/9

www.usecases.org format

5a. System detects failure to communicate with external tax calculation system service:
  1. System restarts the service on the POS node, and continues.
  1a. System detects that the service does not restart.
    1. System signals error.
    2. Cashier may manually calculate and enter the tax, or cancel the sale.

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):
  1. Cashier signa
  2. Cashier ente
  3. System prese

5c. Customer says th
  1. Cashier signa
  2. Cashier ente
  3. Systems app

Extensions (Alternate Flows)

An extension has two parts: the condition and the handling.

Guideline: Write the condition as something that can be detected by the system or an actor. Compare:

5a. System detects failure to communicate with external tax calculation system service **[better – can be detected]**

5a. External tax calculation system not working **[worse – needs inference]**

Fully Dressed Example 6/9

www.usecases.org format

6a. Customer says they intended to pay by cash but don't have enough cash:
    1a. Customer uses an alternate payment method.
    1b. Customer tells Cashier to cancel sale. Cashier cancels sale on System.
7a. Paying by cash:
    1. Cashier enters the cash amount tendered.
    2. System presents the balance due, and releases the cash drawer.
    3. Cashier deposits cash tendered and returns balance in cash to Customer.
    4. System records the cash payment.
7b. Paying by credit:
    1. Customer enters their credit account information.
    2. System sends payment authorization request to an external Payment Authoriza tion Service System, and requests payment approval.
        2a. System detects failure to collaborate with external system:
            1. System signals error to Cashier.
            2. Cashier asks Customer for alternate payment.

Fully Dressed Example 6/9

www.usecases.org format

6a. Customer says they intended to pay by cash but don't have enough cash:
    1a. Customer uses an alternate payment method.
    1b. Customer tells Cashier to cancel sale. Cashier cancels sale on System.
7a. Paying by cash:
    1. Cashier ente
    2. System prese
    3. Cashier depo
    4. System recor
7b. Paying by credit:
    1. Customer en
    2. System sends payment authorization request to an external Payment Authoriza tion Service System, and requests payment approval.
        2a. System detects failure to collaborate with external system:
            1. System signals error to Cashier.
            2. Cashier asks Customer for alternate payment.

Extensions (Alternate Flows)

At the end of extension handling, by default the scenario merges back with the main success scenario, unless the extension indicates otherwise (such as by aborting the use-case – see "Cashier cancels sale on System").

Fully Dressed Example 7/9

www.usecases.org format

3. System receives payment approval and signals approval to Cashier.
   3a. System receives payment denial:
      1. System signals denial to Cashier.
      2. Cashier asks Customer for alternate payment.
4. System records the credit payment, which includes the payment approval.
5. System presents credit payment signature input mechanism.
6. Cashier asks Customer for a credit payment signature. Customer enters signature.

7c. Paying by check...

7d. Paying by debit...

7e. Customer presents coupons:

1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
   1a. Coupon entered is not for any purchased item:
      1. System signals error to Cashier.

Fully Dressed Example 7/9

www.usecases.org format

3. System receives payment approval and signals approval to Cashier.
   3a. System receives payment denial:
      1. System signals denial to Cashier.
      2. Cashier asks Customer for alternate payment.
4. System records the credit payment, which includes the payment approval.
5. System presents credit payment signature input mechanism.
6. Cashier asks C
   signature.

7c. Paying by check..
7d. Paying by debit...
7e. Customer presen
   1. Before handl
      reduces pric
      accounting r
      1a. Coupon entered is not for any purchased item:
         1. System signals error to Cashier.

Extensions (Alternate Flows)

Sometimes, a particular extension point is quite complex, as in the "paying by credit" extension. This can be a motivation to express the extension as a separate use case.

This example demonstrates also the notation to express failures within extensions.

Fully Dressed Example 8/9

www.usecases.org format

9a. There are product rebates:
    1. System presents the rebate forms and rebate receipts for each item with a rebate.

9b. Customer requests gift receipt (no prices visible):
    1. Cashier requests gift receipt and System presents it.

Special Requirements:

- Touch screen Ul on a large flat panel monitor. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Somehow, we want robust recovery when access to remote services such the inventory system is failing.
- Language internationalization on the text displayed.
- Pluggable business rules to be insertable at steps 3 and 7.

Fully Dressed Example 8/9

www.usecases.org format

9a. There are product rebates:
    1. System presents the rebate forms and rebate receipts for each item with a rebate.

9b. Customer requests gift receipt (no prices visible):
    1. Cashier requests gift receipt and System presents it.

Special Requirements:

- Touch screen Ul on a large flat panel monitor. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Somehow, we want robust recovery when access to remote services such the inventory system is failing.
- Language internationalization on the text displayed.
- Pluggable busines

Special Requirements

Non-functional requirement, quality attribute, or constraint related directly to the use-case.

If it applies globally, it can be recorded in a separate document – Supplementary Specification

Fully Dressed Example 9/9

www.usecases.org format

Technology and Data Variations List:

3a. Item identifier entered by bar code laser scanner (if bar code is present) or keyboard.
3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
7a. Credit account information entered by card reader or keyboard.
7b. Credit payment signature captured on paper receipt. But within two years, we predict many customers will want digital signature capture.

Open Issues:

- What are the tax law variations?
- Explore the remote service recovery issue.
- What customization is needed for different businesses?
- Must a cashier take their cash drawer when they log out?
- Can the customer directly use the card reader, or does the cashier have to do it?

Fully Dressed Example 9/9

www.usecases.org format

Technology and Data Variations List:

3a. Item identifier entered by bar code laser scanner (if bar code is present) or keyboard.

3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.

7a. Credit account information entered by card reader or keyboard.

7b. Credit payment signature captured on paper receipt. But within two years, we predict many cus

Technology and Data Variation List

Captures technical variations in *how* something must be done, but not *what*.

Represents early design decisions, which however in some cases are unavoidable.

Open Issues:

- What are the tax l
- Explore the remot
- What customizati
- Must a cashier take their cash drawer when they log out?
- Can the customer directly use the card reader, or does the cashier have to do it?

IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)

IEEE recommended practice for specifying requirements

IEEE 830-1998

Basic issues to be addressed:

- Functionality.
  - What is the software supposed to do?
- External interfaces.
  - How does the software interact with people, the system's hardware, other hardware, and other software?
- Performance.
  - What is the speed, availability, response time, recovery time of various software functions, etc.?
- Attributes.
  - What are the portability, correctness, maintainability, security, etc. considerations?
- Design constraints imposed on an implementation.
  - Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

SRS should be …

- Correct
  - Every requirement is one that the software shall meet
- Unambiguous
  - Every requirement has only one interpretation
  - Glossary if terms used can be differently understood
- Complete
  - All significant requirements (functionality, external interfaces, …)
  - Definition of responses to all realizable classes of input data in all realizable classes of situations (including invalid inputs if they are possible)
  - If something is marked as TBD (to be determined), it should be accompanied with
    - A description of the conditions causing the TBD (e.g., why an answer is not known) so that the situation can be resolved;
    - A description of what must be done to eliminate the TBD, who is responsible for its elimination, and by when it must be eliminated.

SRS should be …

- Consistent
  - No subset of individual requirements described in it conflict
  - The specified characteristics of real-world objects may conflict. For example,
    - The format of an output report may be described in one requirement as tabular but in another as textual.
    - One requirement may state that all lights shall be green while another may state that all lights shall be blue.
  - There may be logical or temporal conflict between two specified actions. For example,
    - One requirement may specify that the program will add two inputs and another may specify that the program will multiply them.
    - One requirement may state that "A" must always follow "B", while another may require that "A and B" occur simultaneously.
  - Two or more requirements may describe the same real-world object but use different terms for that object.
    - For example, a program's request for a user input may be called a "prompt" in one requirement and a "cue" in another. The use of standard terminology and definitions promotes consistency.

SRS should be …

- Ranked for importance and/or stability
  - Each requirement has an identifier to indicate either its importance or stability
- Verifiable
  - There exists some finite cost-effective process with which a person or machine can check that the software meets the requirements
  - Verifiable statement: *"Output of the program shall be produced within 20 s of event 60% of the time; and shall be produced within 30 s of event 100% of the time."*
- Modifiable
  - Any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style
  - Generally this requires:
    - Have a coherent and easy-to-use organization with a table of contents, an index, and explicit crossreferencing;
    - Not be redundant (i.e., the same requirement should not appear in more than one place in the SRS);
    - Express each requirement separately, rather than intermixed with other requirements.
- Traceable
  - The origin of each requirement is clear and can be easily referenced from the future development or enhancement documentation
  - *Backward traceability (i.e., to previous stages of development)*
    - Each requirement explicitly references the source
  - *Forward traceability (i.e., to all documents spawned by the SRS)*
    - Each requirement has a unique name and reference number

Parts of SRS

- Introduction
  - Purpose
  - Scope
  - Definitions, acronyms, and abbreviations
  - References
  - Overview
- Overall description
  - Product perspective
  - Product functions
  - User characteristics
  - Constraints
  - Assumptions and dependencies
  - Apportioning of requirements
- Specific requirements
  - Sources of requirements
    - External interfaces
    - Functions
    - Performance requirements
    - Logical database requirements
    - Design constraints
    - Standards compliance
    - Software systems attributes – Reliability, availability, security, maintainability, portability
  - Organization of requirements
    - By system mode
    - By user class
    - By objects
    - By feature
    - By stimulus
    - By response
    - By functional hierarchy
- Supporting information
  - Table of contents, appendices

Derivation from KAOS

- Definitions, acronyms, and abbreviations
  - Derived from the KAOS object model
- Assumptions and dependencies
  - Assumptions used in the KAOS goal model
  - Obstacles that the system is not expected to deal with
- Product functions
  - Replaced by “User requirements”
  - Contains the goal models (traversed from top-down) with description
- Apportioning of requirements
  - provides a table containing the list of all the requirements presented in the User requirements section sorted by priority level

Derivation from KAOS

- Specific requirements
  - Replaced by “System requirements”
  - System architecture
    - Decomposition of the system by KAOS agents. For each agent, the list of the requirements he/she/it is responsible for is listed
  - Object model.
    - Each diagram of the KAOS object model with explanation
  - Operation model
    - Each diagram of the KAOS operation model with explanation
    - At the end of the section, a table showings the operations performed by each agent and the requirements these operations contribute to satisfy

SRS_001: Robot stops when the emergency switch is pressed

Description: The robot has an emergency switch. When this switch is pressed, the robot must immediately stop any of its movement. To resume the operation, the emergency switch has to be released and the robot has to be switched on (from the power off state) by the power switch on its side.

Rationale: This is a rule of the Line Follower Challenge (sect. 2, ln. 4)

Dependencies: SRS_xxx (Periodic reading of sensors)