CS 4320 / 7320 Software Engineering

Advanced Software Requirements Specifications

Requirements Validation -- Framework

(Possible) GOAL Requirements

User needs & objectives within an organizational/ environmental context (organizational requirements)

e.g.

Develop Better Plans

(Possible) TASK Requirements

Task activities and sub-objectives: where improvements can be made to achieve goal reqs. implies range of information and decision requirements.

e.g.

- **COA Development**
- **COA Wargaming** ⟷
- **COA Selection & Justification**

(Possible) FUNCTIONAL Requirements

General system functions and capabilities for helping users achieve their task requirements

e.g.

- **Attrition Projector**
- **Combat Power Ratio Calculator**
- **Time and Distance Calculator**
- **Battlefield Synchronization Tool**

⟷

(Possible) SYSTEM Requirements

What the system needs to do to implement the required capabilities to achieve task & goal requirements. Included here are knowledge-base requirements

e.g.

- **System Specs. for Attrition Projector**
- **Combat Power Ratio Calculator**
- **Time & Distance Calculator**
- **Battlefield Sync. Tool**

Requirements Validation -- Framework

(Possible) GOAL Requirements

User needs & objectives within an organizational/ environmental context (organizational requirements)

e.g.

(Possible) TASK Requirements

Task activities and sub-objectives: where improvements can be made to achieve goal reqs. implies range of information and decision requirements.

e.g.

(Possible) FUNCTIONAL Requirements

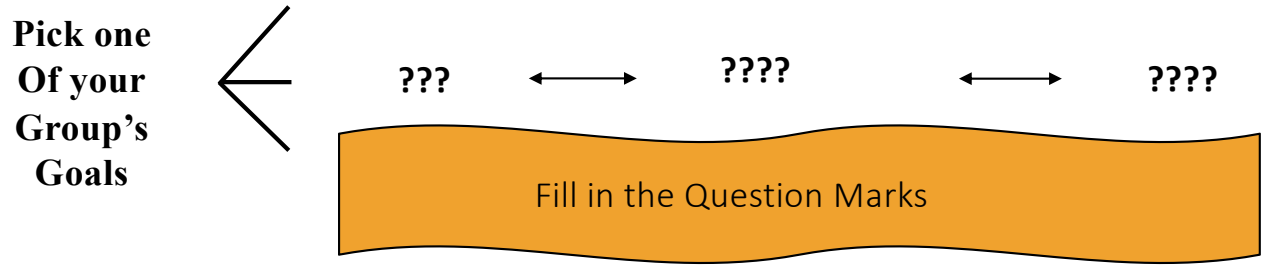
General system functions and capabilities for helping users achieve their task requirements

e.g.

(Possible) SYSTEM Requirements

What the system needs to do to implement the required capabilities to achieve task & goal requirements. Included here are knowledge-base requirements

e.g.



Rotate & Peer Review

Do you understand the progression?

Can you explain what you are seeing from the other group in both directions?

What questions do you need answered in order to be able to confidently explain the relationship in both directions? (You should really have a lot of questions for another team's diagram. Think of it this way: You don't really know the answer! These questions will help the other group know what else needs to be in the SRS, and to make these connections more clear!)

Types of requirements

Business goals and business logic

Purposeful requirements → system goals

Task-support requirements → business-logic usually analyzed as a business use-case or activity-sequence model

Business events and system feature definitions

System events are workflow tasks that require system support
→ system feature definitions

System features can be modeled using system use-cases
→ incorporate high-level system functional requirements.

Types of requirements

User support requirements and use-constraints

Consider the unique roles played by your system users and what this means for how they need to work, e.g.

Order of functions used (navigation → fit with expected task order)

Need to change or revisit functions, rather than working through pre-defined "scripts" (degree of autonomy allowed in configuring how they use the system)

Consider how they wish to have information presented and inputs validated. What does " "usability" mean to *each* set of users?

These elements provide non-functional requirements, that define how the system will provide the functionality required

Environment constraints and technical environment considerations

Supplementary requirements (see reading provided) specify technical aspects of system quality (compatibility, performance, reliability, maintainability, etc.).

Why We Need A Complete Set of System Features

Business logic is modeled through analyzing the information-processing needs of people

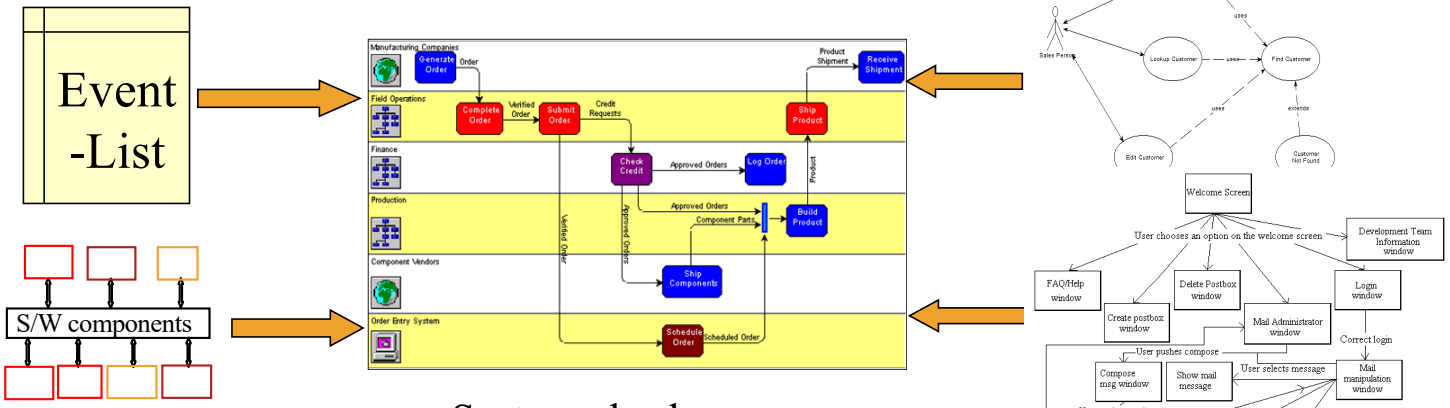
- Workflows (activity-sequences)
- Business use-cases (showing decision-points)

System logic is modeled through analyzing the data-processing needs of the IT system

- Data-flow or object domain models
- System use-cases (showing data-processing logic)

- Feature-definitions act as the ***interface*** between business logic and system logic
- Defining IT data-processing needs in terms of the things that people need to do with the system means that there is some connection between business logic and system logic.
- We have no other way to manage the disconnect.

From Event-Lists
To System Features & System Functions



Business logic
- Event-handling
- Decision support

System plus human information-processing
- **System Features**

Functional specification:
- System Use-cases
- Activity-sequence diagrams (navigation)

External event handling

IT System
Functions

System interfaces

Lets Try to Reconcile an Example

1. (Check internal consistency of your logic). Provide a DFD and use this to verify that you have all the functional processes required to “join the dots.”

2. (Check external consistency of your logic). Walk through the event-list against the high-level (activity-level workflow) swimlane model, that shows non-automated as well as automated user tasks. Use this walkthrough to verify that you have all the business events that are relevant to your system scope supported by system functions; then use the software component model to suggest all the administrative reporting and system management tasks that you have forgotten to include(!).

Non-Functional Requirements

Non-functional requirements specify *how* the system will perform the functions required.

A non-functional requirement is only of use if it can be tested in some way:

- If possible try to assign a measure – e.g. if fast performance is required, state that the system must respond in XX milliseconds.
- If not possible to assign a measure, try to explain how this can be tested – e.g. a store clerk must be able to recall the sequence of commands for this function after a single training session, or 90% of users must be able to locate the record for a member using on-screen hints.

Specifying Non-Functional Requirements From Functional Requirements

Functional Requirements (Use-Case Functions):

UC1.1: The system will provide the store location of the video, when its title or ID is entered

UC1.2: The system will permit the user to check if a member has unpaid or overdue rentals .

UC1.3: The system will check if the video requested is already reserved.

Non-Functional Requirements:

General Non-Functional Requirements for this use-case are:

NF1 The system will display a response within 30 milliseconds.

The system will provide function UC1.1, in the following way:

NF1.1-1 The video will have a bar-code label, that indicates where it is shelved This information is stored in the video record.

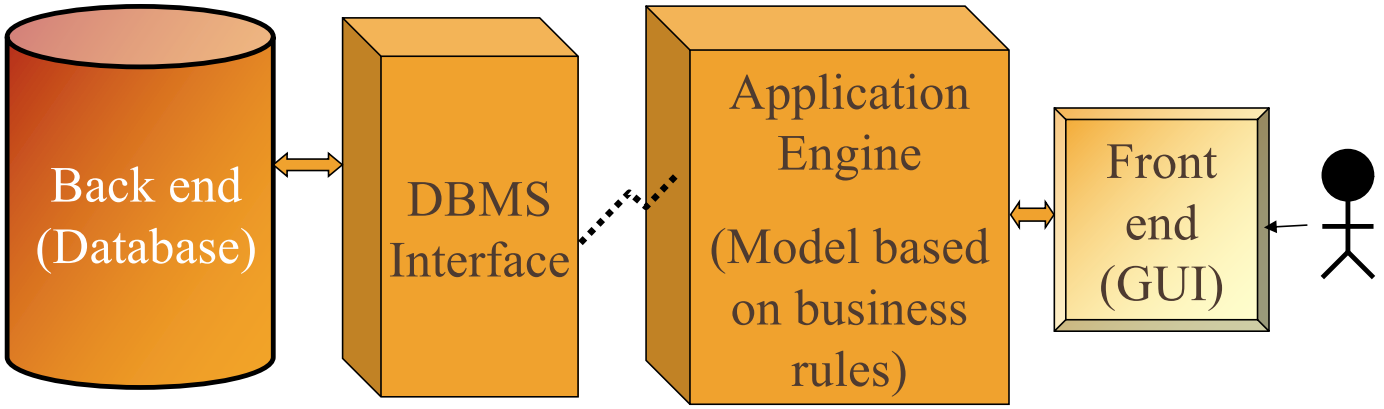
NF1.1-2 The location assigned to any bar-code can be configured by the user (requires additional function: UC 1.8).

The system will provide function UC1.2, in the following way:

NF1.2-1 the system will automatically flash a warning when the member ID is entered, if the member has overdue or unpaid rentals outstanding.

NF1.2-2 Management reports will be printed automatically on at a time specified by Managers and subject to change easily. (see function UC 1.6). *Etc.*

Supplementary Requirements – System Data Use



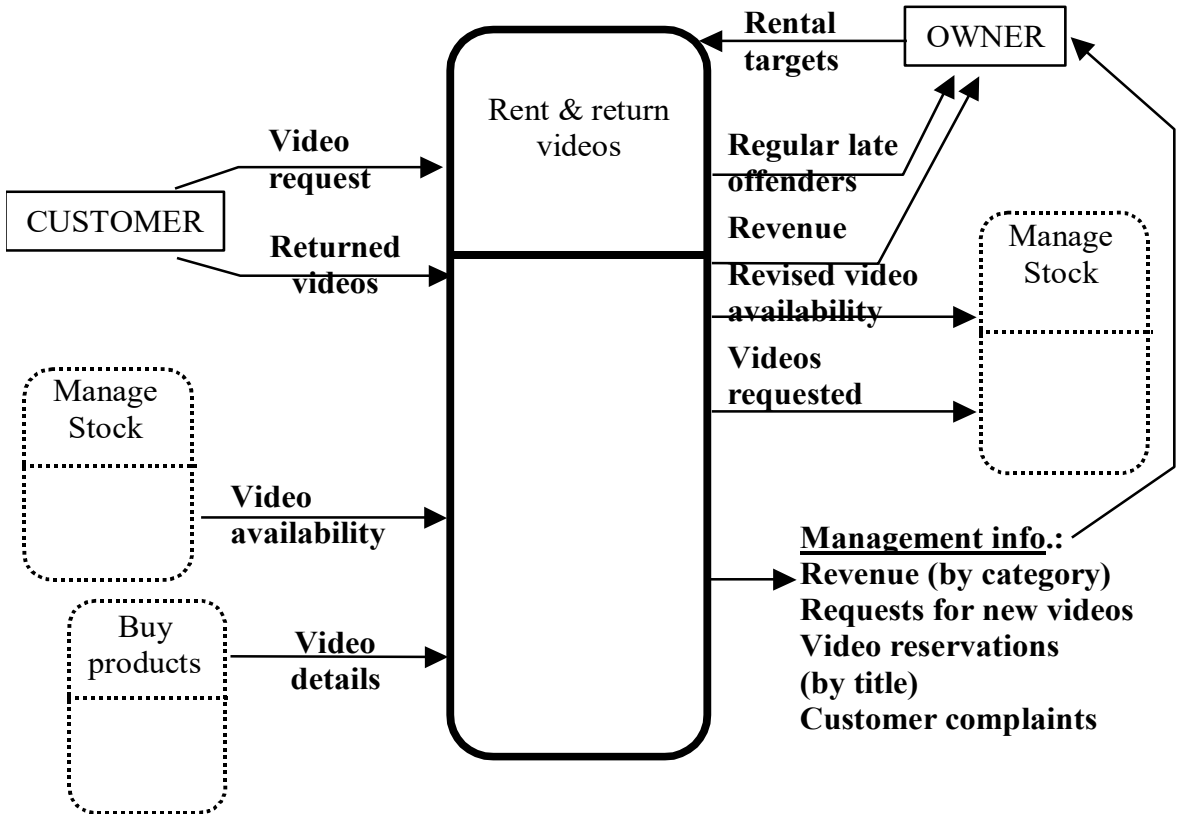
As databases and data stores are a fundamental part of any system architecture, so data-use specification is a fundamental part of any system specification.

Need to define:

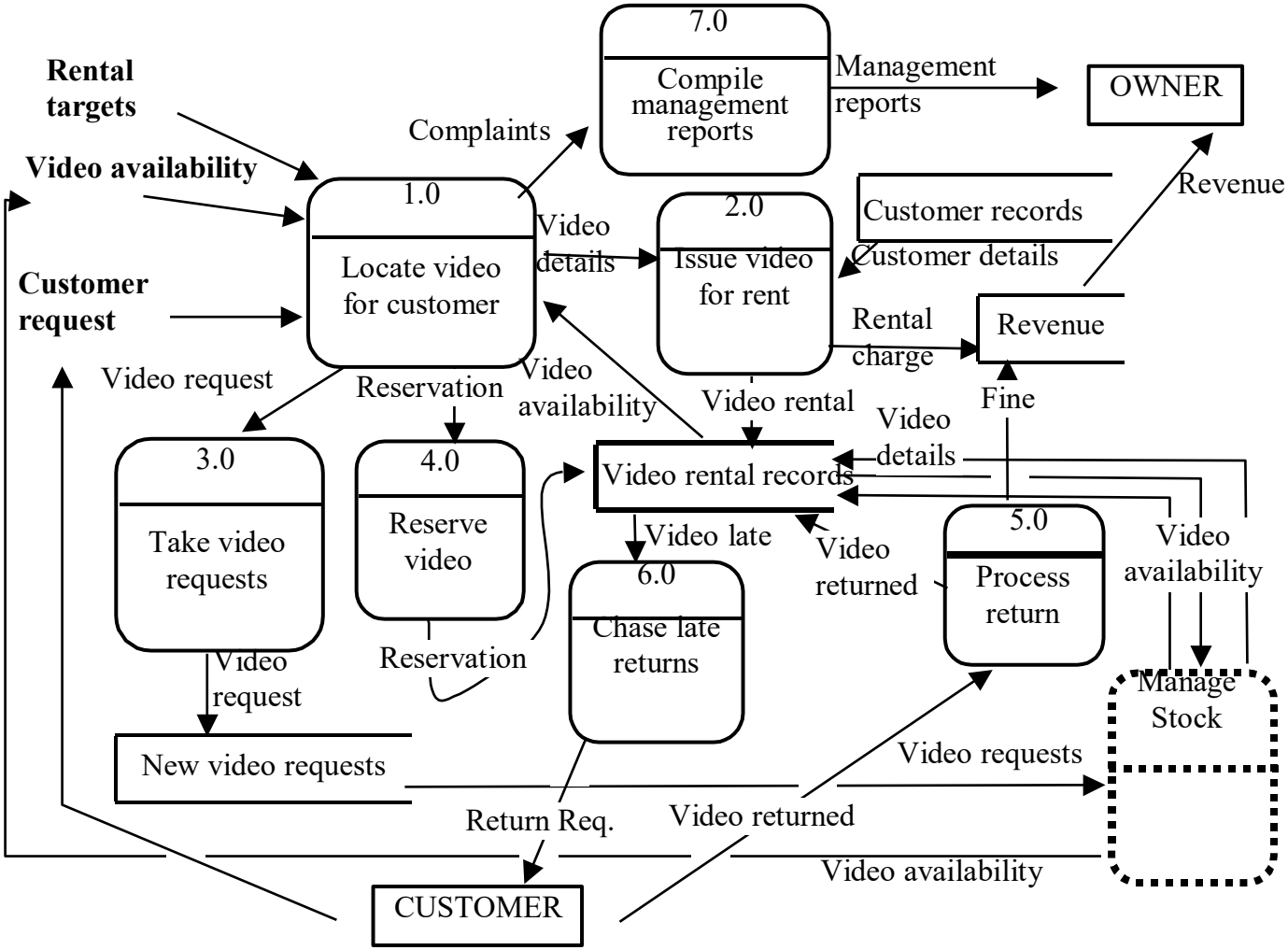
- External data sources and external data users (context diagram);
- Internal data flows and uses (DFD);
- Internal data types/relationships (ERD or class domain model).

The Context Diagram

A high-level context diagram places the system within the external data flows that constitute the information supply chain.

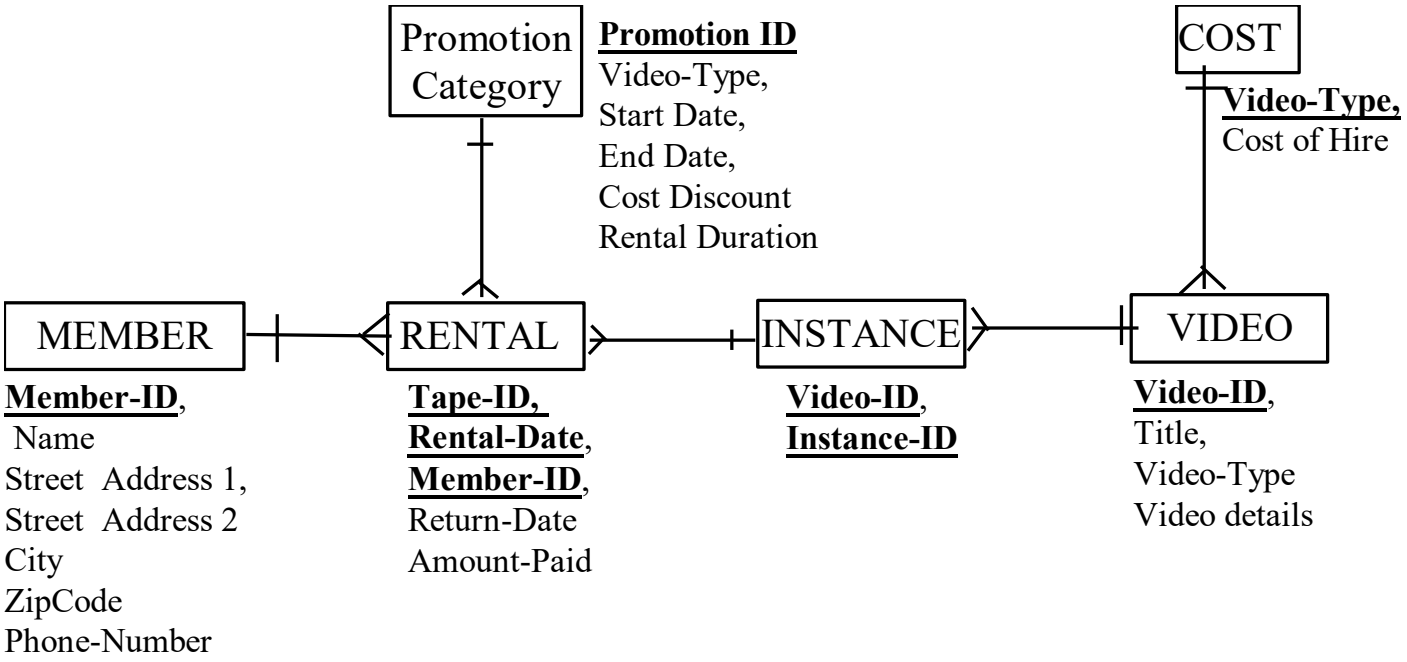


A Data-Flow Diagram Validates The Use-Case List



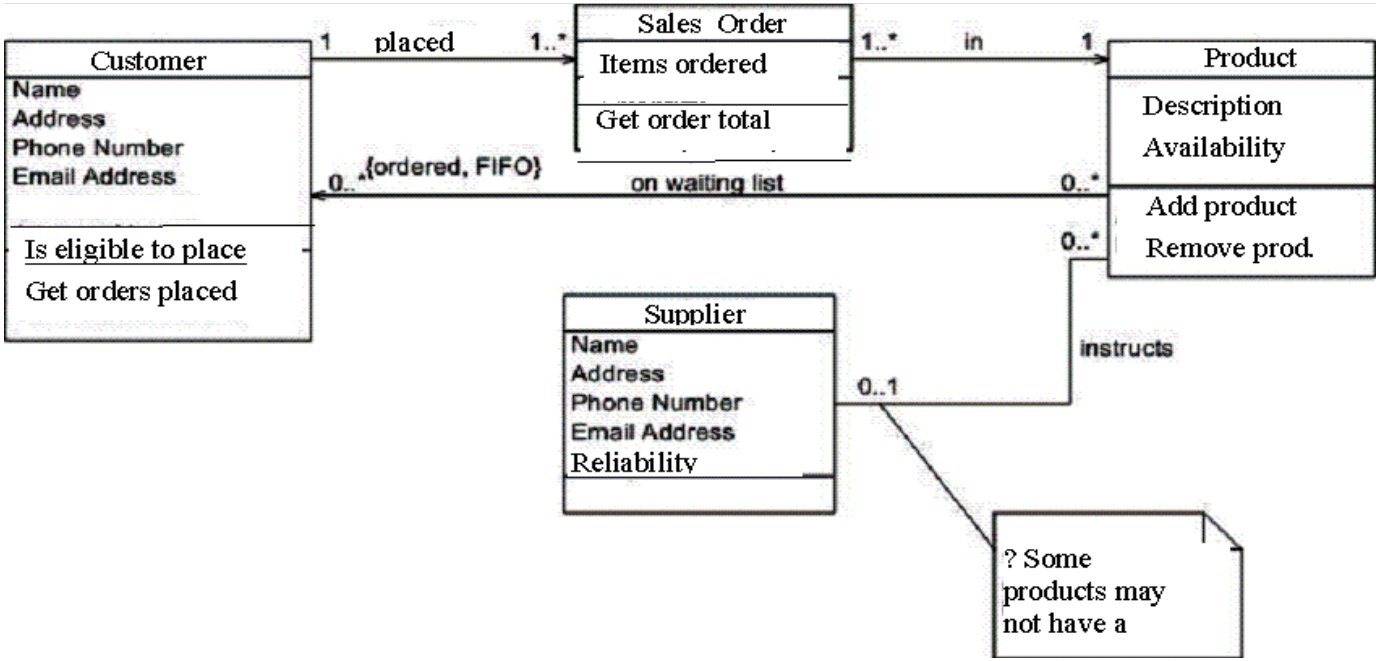
Entity-Relationship Diagram

Section 3.2.3: An ERD shows the main data entities that are used in system processing:



A Class-Domain Model

- This model relates system (data) objects to each other and to the methods (derived from use-case functions) associated with each object. This allows you to check that all use-cases have been included in the spec. (but also requires you to engage in system design, in advance).
- Can use ERD models as an alternative to this.



System Validation Vs. Verification

Verification requires confirmation by testing and the provision of objective evidence that the specified system requirements have been fulfilled.

Validation means establishing by various means of user/stakeholder evaluation that system (product) specifications conform with user needs and intended use(s).

IEEE 830: Characteristics of a Good SRS

Nine Quality Measures:

Correct

Unambiguous

Complete

Consistent

Ranked for importance and stability

Verifiable

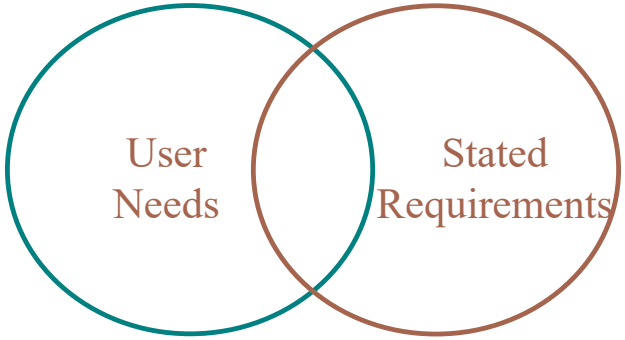
Modifiable

Traceable

Understandable

Correct

"A statement of requirements is correct if and only if every requirement stated therein represents something required of the system to be built." (Davis, 1993)



Some ideas for assessing correctness:

- Walkthroughs with other analysts/designers
- Validation from users (need a user-friendly representation)
- Observe the new system in an operational setting! *(Planned version rollout)*

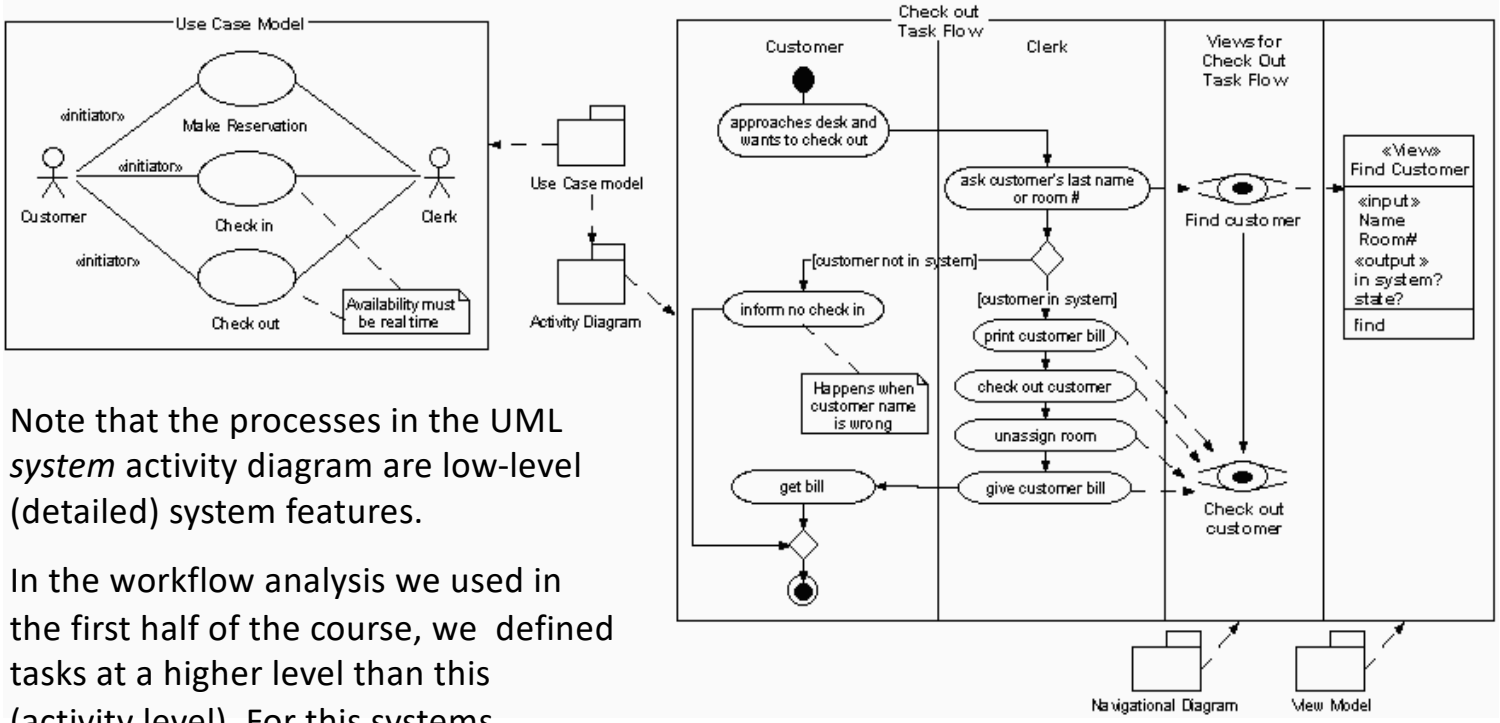
Complete

Completeness can be *assessed* (but not proven) using:

- Interaction with users to understand implicit requirements
  (observations, scenarios, workflow simulations)
- User validation of the "big picture" (walkthroughs with users, prototype evaluation)
- Walkthroughs with other developers.
- Checklists of issues regarding usability, reliability, performance, supportability and system constraints.

Consistent

Use-case and swim-lane views of a Hotel Reservation System:

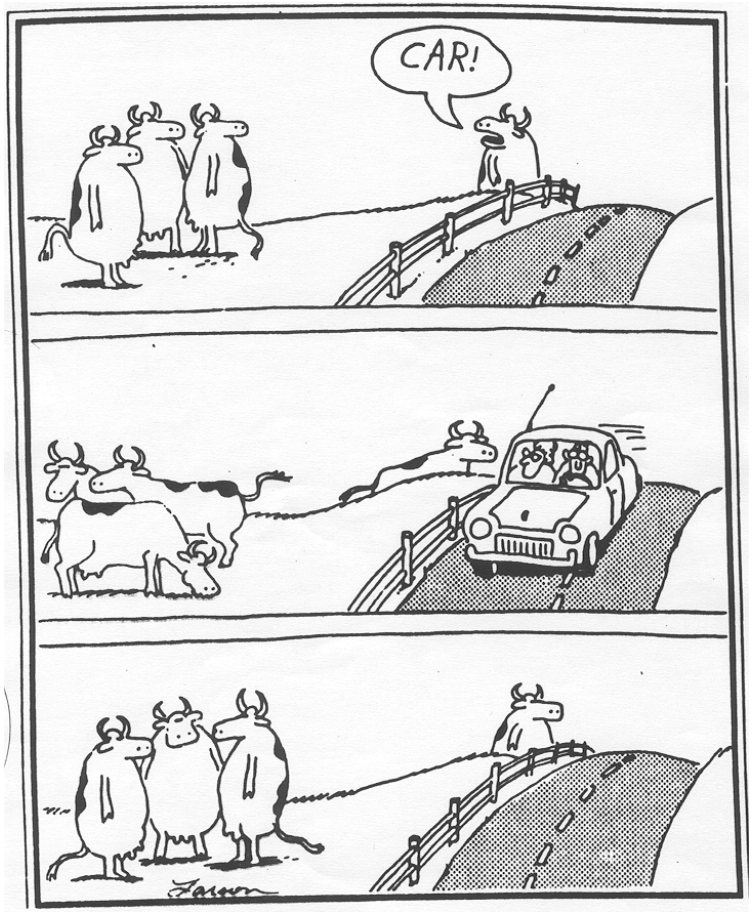


Note that the processes in the UML *system* activity diagram are low-level (detailed) system features.

In the workflow analysis we used in the first half of the course, we defined tasks at a higher level than this (activity level). For this systems analysis, we define ***data-processing*** tasks.

Source: Nuno Jardim Nunes and João Falcão e Cunha, Detailing Use-Cases with Activity Diagrams and Object Views

Consistency …. With What?



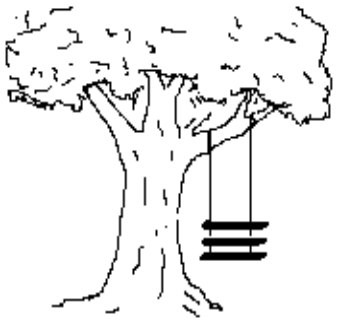
Unambiguous

How many of you understand everything that is required, for my assignments, without additional explanation?

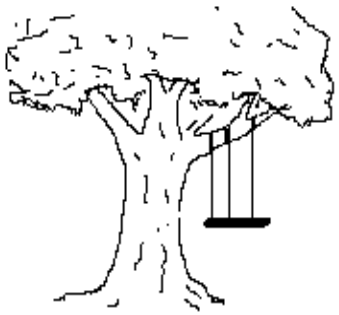
(I'm an experienced systems requirements analyst!)

- See the *techniques for disambiguation* discussed by Leffingwell & Widrig (Chapter 26)

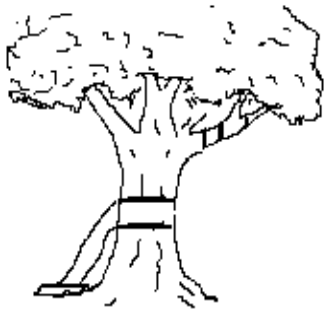
Understandable



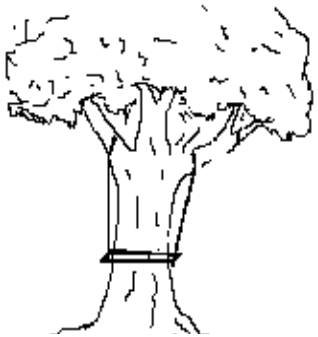
what the user asked for ....



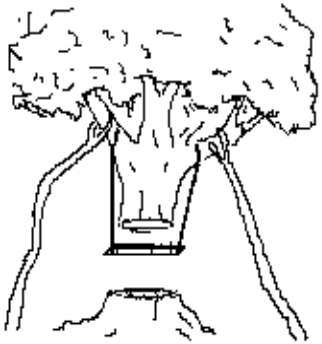
what the feasibility study specified



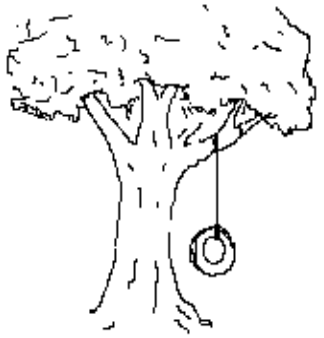
what the system analyst specified



What the programmer inadvertently revised the spec. to



What the user agreed to (acceptance testing)



what the user really wanted !

Ranking For Importance & Stability

Feature:/Use-Case	Stability	Priority	Risk
1. Forecast the popularity of books and items that are offered by suppliers and automatically preorder books for sale, based on popularity forecast.	Low	Critical	Low
2. Assist in placing a customer order, checking the predicted dispatch date and issuing an order confirmation to the customer.	High	Critical	High
4. Directly place a customer order online, generating an order confirmation and a predicted dispatch date.	High	Critical	High
5. Assign a dispatch date for an order and automatically arrange delivery of items in an order, based on availability of items in inventory.	Medium	Critical	Medium
6. Manually schedule a delivery, linked to the availability of items in the order.	Low	Critical	Medium
7. Confirm that inventory will be in stock, to fill an order by a specified date.	Medium	Important	Medium
8. Track the status of an order and communicate order priority and progress	Low	Important	Low
9. Easily check in received goods, so that inventory records are accurate.	Medium	Useful	Low
10. Keep track of what is in stock and when more stock is due, for specific items.	High	Critical	High
11. Manage and track an appropriate sequence of order-processing activities (workflow processes)	High	Critical	High
12. Record what has been done, manually or automatically, to process each step of an order.	High	Critical	Low

Verifiable

Some requirements are more verifiable than others …

- Number of users
- Speed of response
- Aesthetic beauty
- User-friendliness
- Data validity (e.g. date must be in form MM/DD/YY)

Verifiable

System use-cases must be specified from the user point of view.

Functional requirements should result from a thorough *scenario analysis* of how the system will be used.

This requires prototypes and discussions with real users!

Non-functional requirements should be made specific to the context of use.

Non-functional requirements include user-preferred use-case sequences, interaction considerations and performance considerations.

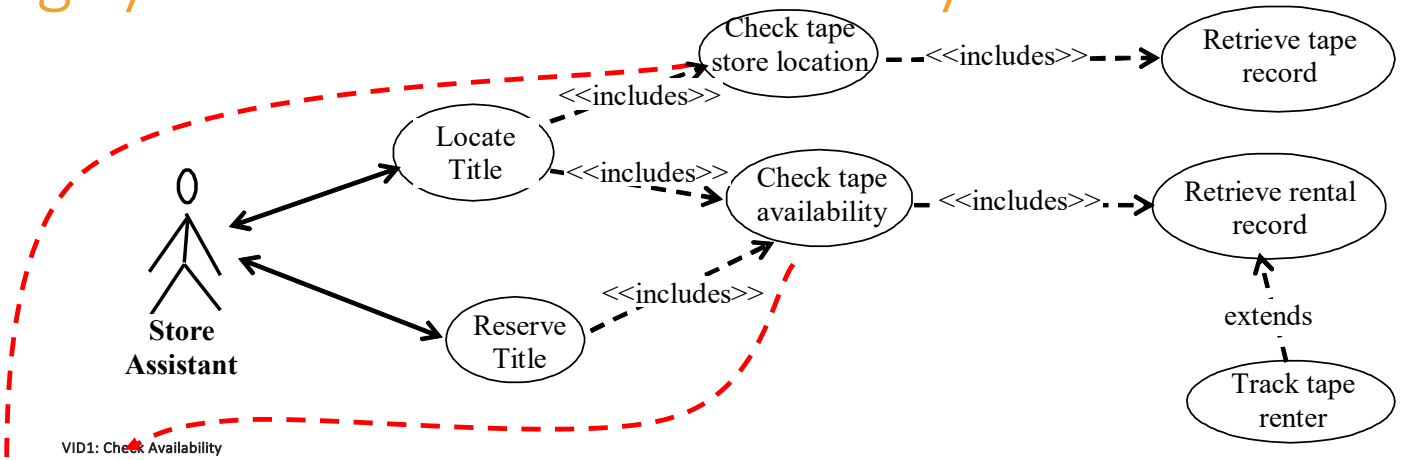
Vague statements about the system being "user-friendly", "efficient" or "effective" serve no purpose. You cannot measure "user-friendly".

What does a user understand by "user-friendly"? How do they define this?

How can you specify *their* definition, in terms of elements such as:
* levels of menus, * number of keystrokes, * number of windows to be navigated to get back to a previous function, * acceptable response-time, * availability of context-specific help, * etc.?

HOW WILL YOU MEASURE SUCCESS, IN ACCEPTANCE TESTS?

Defining System Functions – Traceability and Modifiability



VID1: Check Availability

Indicates if a Title is available for rental, or issued (rented).

Input(s): ID – Video ID of video **instance** requested.

Output(s): Video availability (available, rented)

Processing: The video availability is checked, by determining whether the video is available for rent or is registered as rented in the rental database.

VID2: Locate Video

Provides details of where a video is shelved and whether it is available for rent.

Input(s): *One of:* Name - name of Movie or Program or ID – Video ID of video requested.

Output(s): *May return multiple instances of:* Video availability (available, rented)
Usual video (shelf) location in store

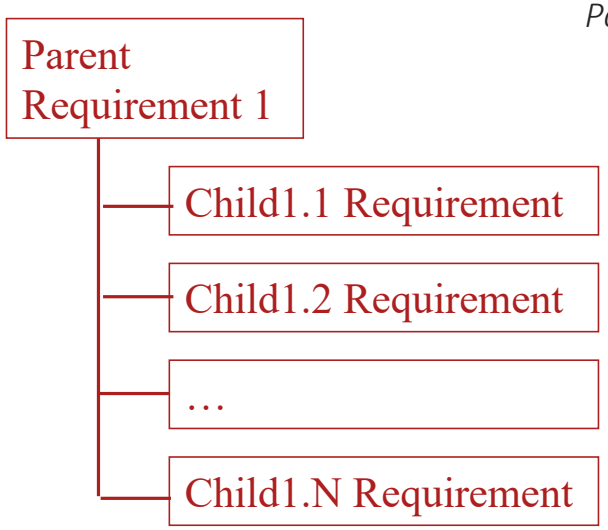
Processing: The video is located, using either a video ID or a Movie name.
The video availability is checked **(Uses Vid1: Check_Avail)**
The video location is checked, by returning the location field in the video record.

Requirements Traceability Matrix

RELATIONSHIPS	UC1	UC2	UC3	UC4	UC5	UC6	UC7
FEA1: Clerk can locate tape from partial movie details					✓		
FEA2: Clerk can track who has which tape		✓	✓	✓			
FEA3: Clerk can locate member info from rental info			✓	✓			
FEA4: Clerk can check-in returned tape easily and quickly	✓	✓					✓
FEA5: Clerk can reserve a tape for rental once it is returned.	✓					✓	✓

UC1: System identifies tape store-location
UC2: System identifies tape availability for rental
UC3: System retrieves tape rental record
UC4: System retrieves member details associated with a rental
UC5: System locates tape identity based on partial movie details.
UC6: System associates tape with member details to create reservation record
UC7: System will show tape as unavailable if rental or reservation record is current.

Using parent-child requirements for traceability



Parent: Use-Case 1 – Rent_Video
[Description] The system will permit a store clerk to rent a video to a store member.

UC1.1: The system will provide the store location of the video, when its title or ID is entered

UC1.2: The system will permit the user to check if a member has unpaid or overdue rentals .

UC1.3: The system will check if the video requested is already reserved.

UC1.4: The system will record the rental of a specific video by a specific member for specific dates.

Non-Functional Requirements: Identifying & Numbering

System Functions:

UC1.1: The system will provide the store location of the video, when its title or ID is entered

UC1.2: The system will permit the user to check if a member has unpaid or overdue rentals .

UC1.3: The system will check if the video requested is already reserved.

Etc.

System Non-Functional Requirements:

General Non-Functional Requirements for this use-case are:

NF1 The system will be easy to use, in that it is designed for someone with little or no computer experience. …

The system will provide function UC1.1, in the following way:

NF1.1-1 The video will have a bar-code label, that indicates where it is shelved This information is stored in the video record. The location assigned to any bar-code can be configured by the user (Function UC1.8).

The system will provide function UC1.2, in the following way:

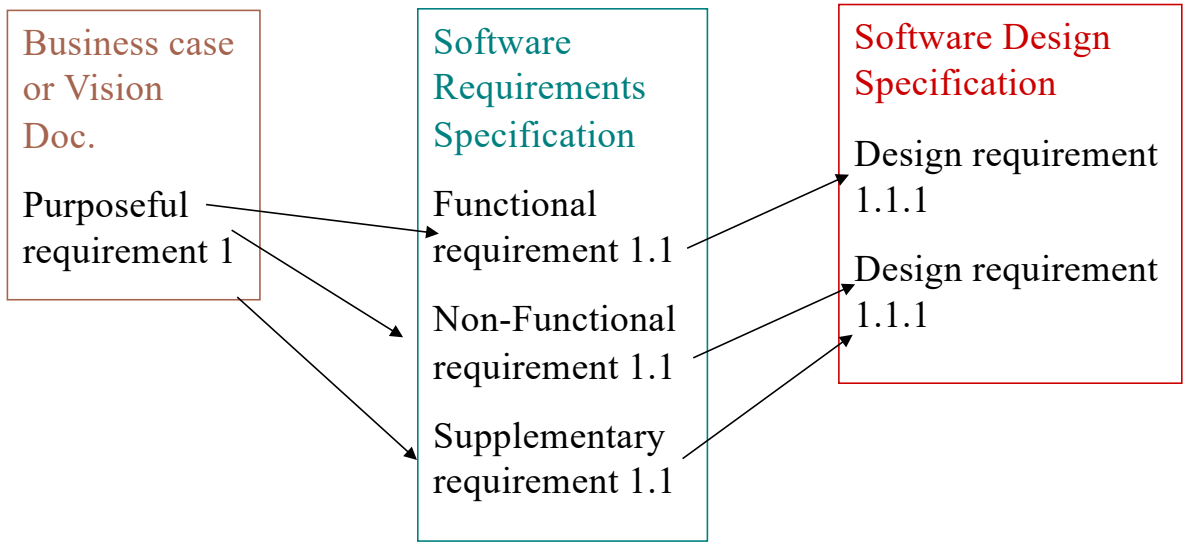
NF1.2-1 the system will automatically flash a warning when the member ID is entered, if the member has overdue or unpaid rentals outstanding.

NF1.2-2 Management reports will be printed automatically on at a time specified by Managers and subject to change easily. (see function UC 1.6) *Etc.*

Traceable

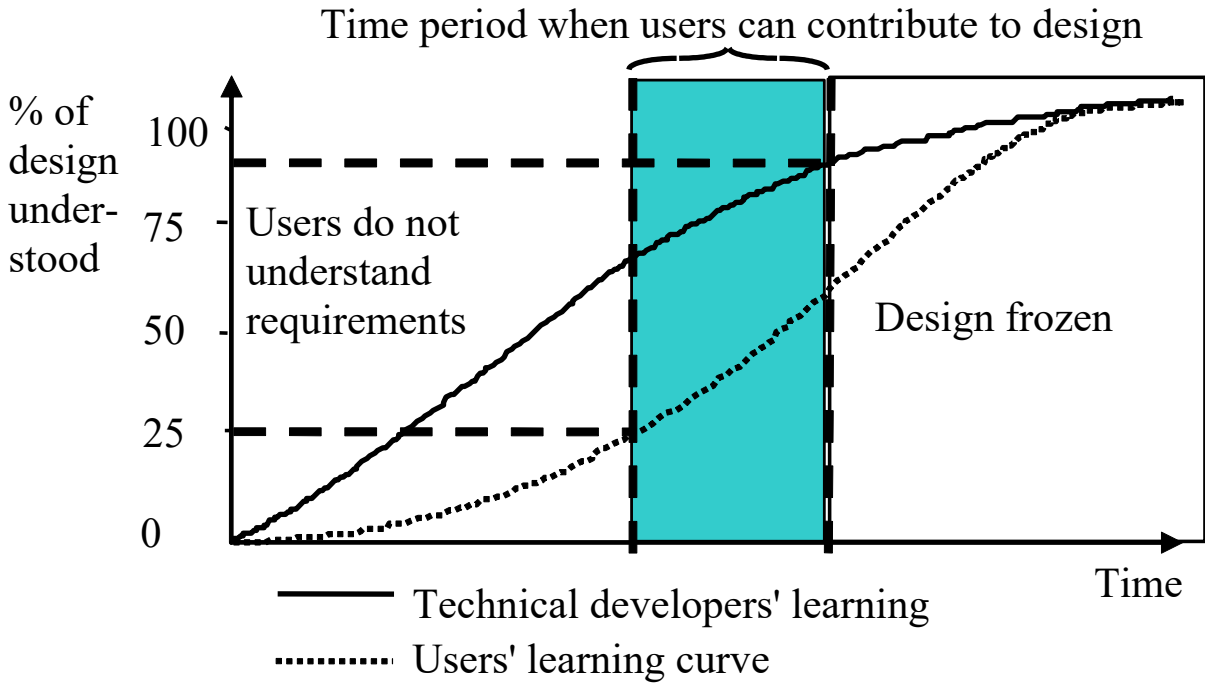
Number requirements

Define a consistent relationship between numbering schema used at different “levels” of requirements analysis and design:



Modifiable

Time-lag Between Developer And User Understanding Of New System



Alternate Ways Of Representing Software
Software *Form & Function*

Scenarios:

Tell a story. Walk through the sequence of events and processing, in words or pictures.

Storyboards:

Show the interfaces to the system, in sequence, with a brief verbal description of triggers and context.

Particularly useful in website design.

Sequence diagrams (e.g SRS Example)

Can represent any sequence that you like: dependent events, screens (to show navigation routes through a website), etc.

Prototypes

Throw-away vs. evolutionary.

Business Process Diagrams

High-level DFDs and a HIGH-LEVEL context diagram (critical)

Information or document diagrams

Can use high-level DFDs, hierarchical diagrams or high-level ERDs.

Critical Success Factors For Software Requirements

Goals:

Must be intelligible, meaningful and useful to three groups of document users:

- Project management and sponsors
- IT implementers
- System users.

Must achieve a **high-quality** representation of system requirements *(nine quality indicators).*

Critical Success Factors:

Use models that different users of the SRS will understand.

Ensure that requirements are validated by key document stakeholders.

Check your requirements against quality "checklist".

Use multiple representations of system as cross-checks, to ensure internal consistency, completeness, etc.