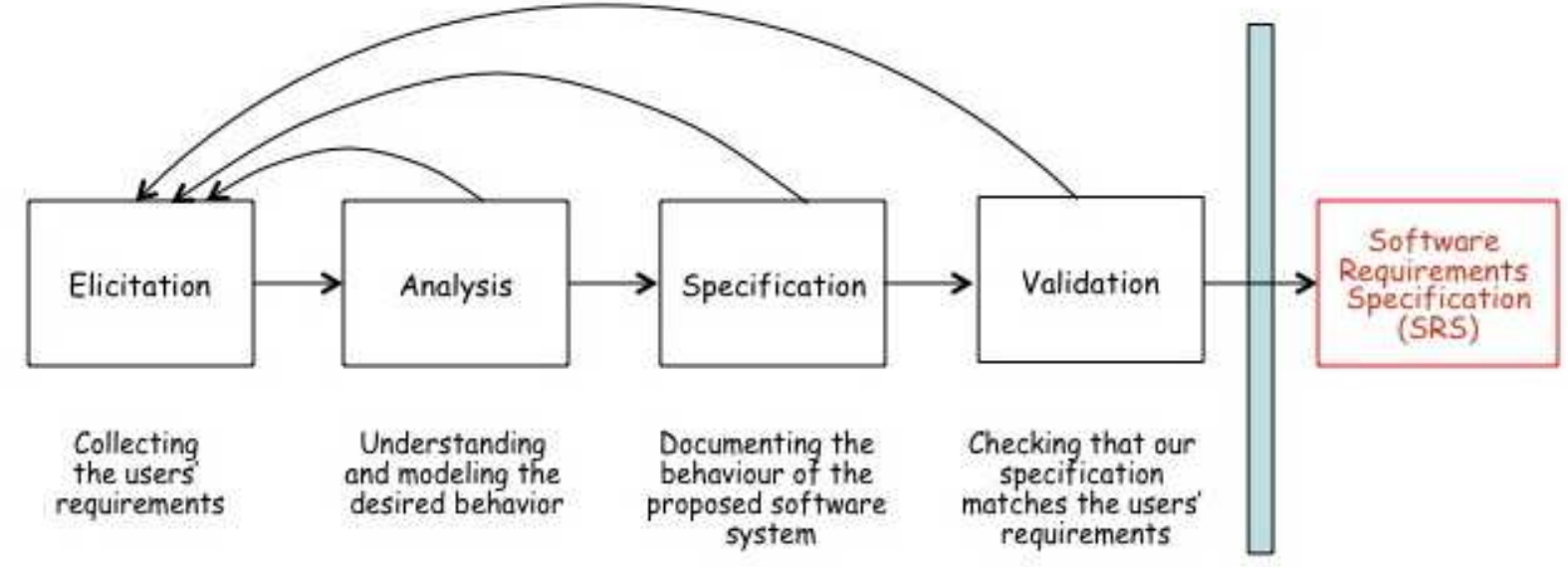
SE1: Software Requirements Specification and Analysis

Winter 2010

Software Requirements Specification

Review: Requirements Process



Today's Agenda

Software Requirements Specifications (SRS)

- IEEE standard for organizing an SRS
- User manual as SRS

Required Reading: IEEE Recommended Practice for SRSs, 1998
(available from an on-campus machine via the course web page)

Lecture includes some excerpts from
Requirements document for an automated teller machine network
http:/www.cs.umd.edu/projects/SoftEng/ ESEG/manual/error_abstraction/docs/atm.ps

SRS Contents

The main issues that the SRS should address are

- Functionality
- External interfaces
- Performance
- Quality attributes
- Design constraints

Not process requirements
Not design decisions

IEEE SRS Organization

Table of Contents

Table of Figures

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, acronyms, abbreviations
    - 1.4 References
    - 1.5 Overview
2. Overall description
    - 2.1 Product perspective
    - 2.2 Product functions
    - 2.3 User characteristics
    - 2.4 Constraints
    - 2.5 Assumptions and dependencies
3. Specific requirements /* variable organization */

Appendices

Index

Section 1: Introduction

This section is an introduction to the SRS document: scope of project, audience, background knowledge of reader, additional information needed to read rest of document.

Table of Contents

Table of Figures

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, acronyms, abbreviations
    - 1.4 References
    - 1.5 Overview

1.1 Purpose

This document describes the software requirements and specification for an automated teller machine (ATM) network. The document is intended for the customer and the developer (designers, testers, maintainers).

The reader is assumed to have basic knowledge of banking accounts and account services. Knowledge and understanding of UML diagrams is also required.

1.2 Scope

The software supports a computerized banking network called YouBank. The network enables customers to complete simple bank-account services via automated teller machines (ATMs) that may be located off premise and that need not be owned and operated by the customer's bank. The ATM identifies a customer by a cash card and password. It collects information about a simple account transaction (e.g., deposit, withdrawal, transfer, bill payment), communicates the transaction information to the customer's bank, and dispenses cash to the customer. The banks provide their own software for their own computers. The YouBank software requires appropriate record keeping and security provisions. The software must handle concurrent accesses to the same account correctly.

1.3 Definition vs. Abbreviation

Definition:

- Account
  A single account at a bank against which transactions can be applied. Accounts may be of various types with at least checking and savings. A customer can hold more than one account.

Abbreviation:

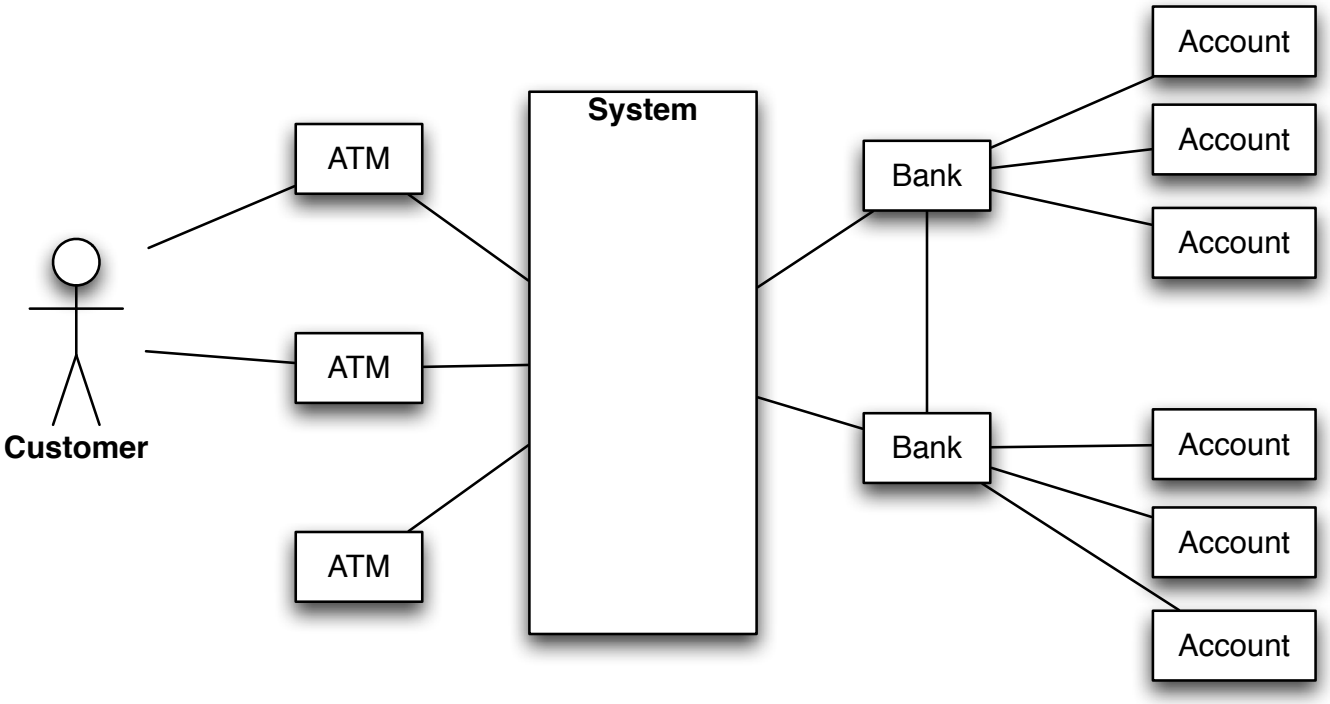
- maxDailyWD
  The maximum amount of cash that a customer can withdraw from an account in a day (from 00:00 AM to 23:59 PM) via ATMs.

Section 2: Overall Description

This section gives an overview description of the system under development, including general factors that affect the product and its requirements.

2. Overall description
    - 2.1 Product perspective
    - 2.2 Product functions
    - 2.3 User characteristics
    - 2.4 Constraints
    - 2.5 Assumptions and dependencies

2.1 Product Perspective



2.3 User Characteristics

- Document any assumptions you make about the user and any assumptions you make about the background or how much training the user will need to use the system.

  For example, you could build different user interfaces for knowledgeable and novice users.

- Only consider user characteristics that affect the software requirements

2.3 User Characteristics

There are several users of the ATM network:

- Customers are simply members of the general public with no special training.
- Bank security personnel need have no special education or experience.
- Maintainers must be experienced network administrators, to be able to connect new ATMs to the network.

2.4 General Constraints

- Sources of other constraints on requirements
  - regulatory policies
  - hardware limitations
  - parallel operation
  - audit functions
  - control functions
  - criticality of the application
  - safety and security considerations
  - standards
  - laws

2.5 Assumptions and Dependencies

- Assumptions about input, or environmental behavior
  - Ex: hardware never fails
  - Ex: ATM casing is impenetrable
  - Ex: limited number of transactions per day (sufficient paper for receipts)
  - Ex: limited amount of money withdrawn per day (sufficient money)
- What conditions could cause the system to fail?
- What changes in the environment could cause changes to the software requirements?

Section 3: Specific Requirements

This section of the SRS should contain all of the software requirements and specification.

At a minimum, it should include descriptions of

- All interfaces to the system
  - Every input (stimulus) into the system
  - Every output (response) from the system
- All functions performed by the system
  - validity checks on inputs
  - relationship of outputs to inputs
  - responses to abnormal situations (e.g., overflow, error handling)

Input and output definitions should be consistent among use cases, functional specifications, state machine diagrams, and UIs.

3.1 External Interfaces

- Detailed descriptions of all inputs and outputs
  - Name of input (or output)
  - Description of purpose
  - Source of input or destination of output
  - Valid range, accuracy, and/or tolerance
  - Units of measure
  - Timing
  - Relationships to other inputs/outputs
  - Screen formats/organization
  - Window formats/organization
  - Data formats
  - Command formats

Output variable

Output Data Item: Steering Error

Acronym: //STERROR//

Hardware: Attitude Direction Indicator (ADI)

Description: //STERROR// controls the position of the vertical needle on the ADI. A positive value moves the pointer to the right when looking at the display. A value of zero centers the needle.

Characteristics of Values

Unit: Degrees

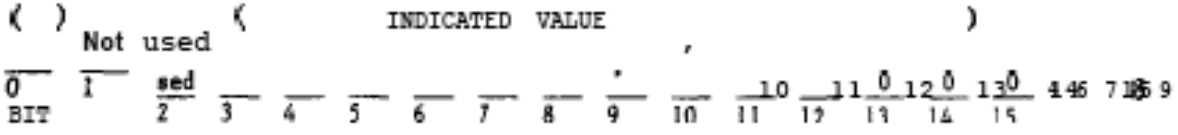
Range: -2.5 to +2.5

Accuracy: ± .1

Resolution: .00122

Instruction Sequence: WRITE 229 (Channel 7)
Test Carry Bit = 0 for request acknowledged
If not, restart

Data Representation: 11-bit two's complement number, bit 0 and bits 3-12
scale = 512/1.25 = 409.6
offset = 0



Timing Characteristics: Digital to DC voltage conversion. See Section 1.5.7.

Comments: The pointer is driven by a stepper motor.

T. Alspaugh et al., *Software Requirements for the A-7E Aircraft,* NRL technical report, 1988.

3.2 Functional Requirements

- Use case descriptions
- Sequence diagrams
- Domain Model
- Functional Specifications
- StateMachine model
- Constraints

Section 3: Specific Requirements

3.3 Performance Requirements

- number of terminals to be supported
- number of simultaneous users to be supported
- amount and type of information to be handled
- number of transactions to be processed within a set time period
  - normal workload conditions
  - peak workload conditions

3.4 Design Constraints

3.5 Quality Attributes

- nonfunctional properties (besides performance), expressed as testable constraints

Appendices

Glossary: The glossary serves as a central place to give a brief description of each term (class, attribute, function, variable). The glossary is a simplified version of what is often called a data dictionary in requirements.

Index: The index maps each important word or phrase to the numbers of all pages in which it appears. When combined with the glossary, cross-referencing is easy.

Section 3 Organization

A.3 Template of SRS Section 3 organized by user class

3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 User class 1
    - 3.2.1.1 Functional requirement 1.1
    - .
    - .
    - .
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 User class 2
  - .
  - .
  - .
  - 3.2.*m* User class *m*
    - 3.2.*m*.1 Functional requirement *m*.1
    - .
    - .
    - .
    - 3.2.*m*.*n* Functional requirement *m*.*n*
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

IEEE 830-1998 Recommended Practice for Software Requirements Specification

Section 3 Organization

A.5 Template of SRS Section 3 organized by feature

3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 System features
  - 3.2.1 System Feature 1
    - 3.2.1.1 Introduction/Purpose of feature
    - 3.2.1.2 Stimulus/Response sequence
    - 3.2.1.3 Associated functional requirements
      - 3.2.1.3.1 Functional requirement 1
      - .
      - .
      - .
      - 3.2.1.3.*n* Functional requirement *n*
  - 3.2.2 System feature 2
  - .
  - .
  - .
  - 3.2.*m* System feature *m*
  - .
  - .
  - .
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

IEEE 830-1998 Recommended Practice for Software Requirements Specification

Section 3 Organization

A.6 Template of SRS Section 3 organized by stimulus

3. Specific requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Stimulus 1
    - 3.2.1.1 Functional requirement 1.1
    - .
    - .
    - .
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 Stimulus 2
  - .
  - .
  - .
  - 3.2.*m* Stimulus m
    - 3.2.*m*.1 Functional requirement *m*.1
    - .
    - .
    - .
    - 3.2.*m*.*n* Functional requirement *m*.*n*
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

IEEE 830-1998 Recommended Practice for Software Requirements Specification

Dilbert on Requirements Documentation

Today's Agenda

Software Requirements Specifications (SRS)

- IEEE standard for organizing an SRS
- User manual as SRS

Other Artifacts that Express Requirement

- User manual
- Test cases
- Help system

These are artifacts that any project for commercial or contract software must produce.

They are a representation of the system's requirements.

They are typically produced late in the development process, but if produced earlier, they could serve as a requirements document and help to identify requirements errors early.

Fred Brook's Observation

In 1975, in MM-M, Fred Brooks equated the user manual with the written SRS:

"The manual must not only describe everything the user does see, including all interfaces; it must also refrain from describing what the user does not see. That is the implementer's business, and there his design freedom must be unconstrained. The architect must always be prepared to show an implementation for any feature he describes, but he must not attempt to dictate the implementation."

DeMarco and McConnell

- Tom DeMarco also suggests using user manuals as SRSs, most notably in *The Deadline*.
- In Software Project Survival Guide, Steve McConnell says:

  "Prior to placing the prototype under change control, work can begin on a detailed user documentation (called the User Manual/Requirements Specification). This is the documentation that will eventually be delivered to the software's end users. Typically, this documentation is developed at the end of the project, but in this book's approach, it is developed near the beginning."

Lisa & MacIntosh

- It is said that the user manual for the Lisa and Macintosh computers were written completely before implementation of their software began.
- The user manuals were given to systems programmers as the SRS of the user interfaces (UIs) and of the underlying systems.

Good User Manuals

Without meaning to be formal documentation, good user manuals have the following elements in common:

- Lexicon! Descriptions of underlying and fundamental concepts of the software
  - A good way to organize the lexicon is around the abstractions identified in the domain model
- Use Cases! A graduated set of examples each showing
  - a problem situation the user faces
  - some possible user actions to the problem, in the form of commands to the software
  - the software's response to these commands
- Reference Manual! A systematic summary of all the commands

When it Works, When it Doesn't

Only works for systems where the user manual describes all but trivially explained requirements.

- If there are multiple kinds of users, can write a user manual for each user view
- But then need to keep the manuals consistent

User manuals won't be a good SRS for

- Autonomous systems that have no real human users
- Systems where the algorithms are important and the UI is less of an issue
- Systems with nontrivial NFRs

User Manuals vs. SRSs

User Manuals

- Favours users
- Is use-case centric
- Needs to be created as part of product
- More likely to be kept up to date
- Can be input to test-case generation

SRSs

- Favours developers
- Is feature/function centric
- Might not be looked at after coding starts
- Harder to "cheat" and handwave over details

In summary, a well-written user manual can effectively serve as an SRS because it is a requirements view of system.

Summary

Software Requirements Specifications (SRS)

- IEEE standard for organizing an SRS
- User manual as SRS

Next Lecture: Cost Estimation

Readings: none