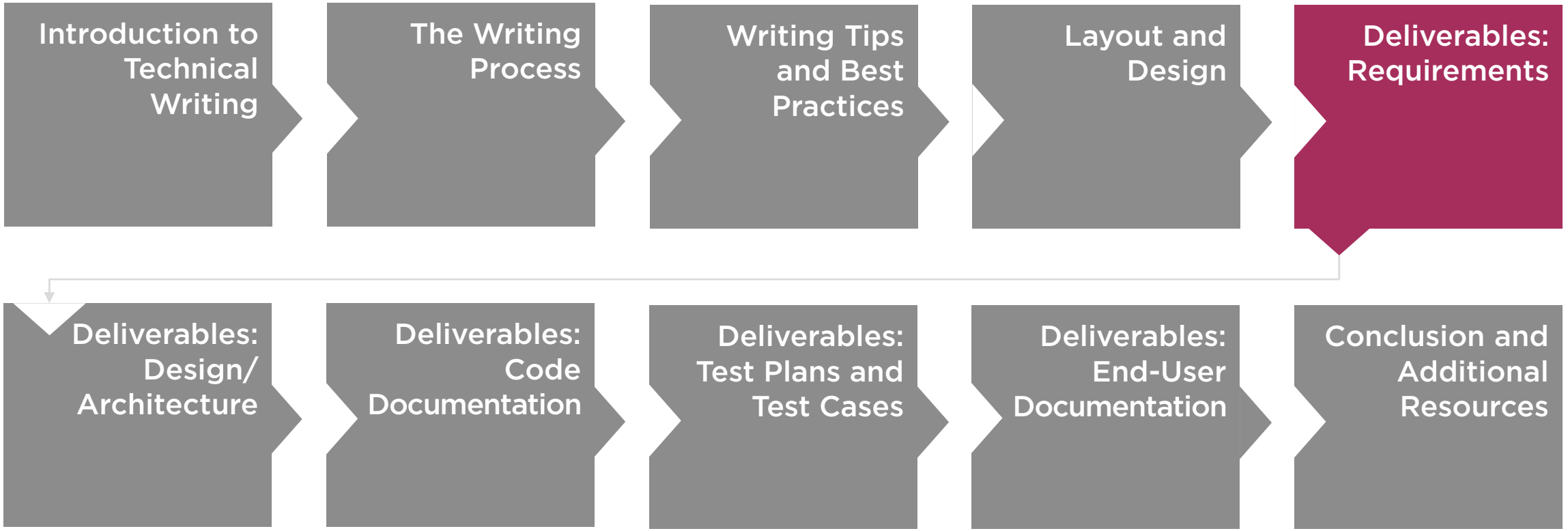
Deliverables: Requirements

Amber Israelsen
DEVELOPER, AUTHOR, TRAINER

www.amberisraelsen.com

Course Outline



I was looking through some of the requirements that Justin started. They’re a bit of a mess.

The application will do stuff.

The system will be easy to maintain.

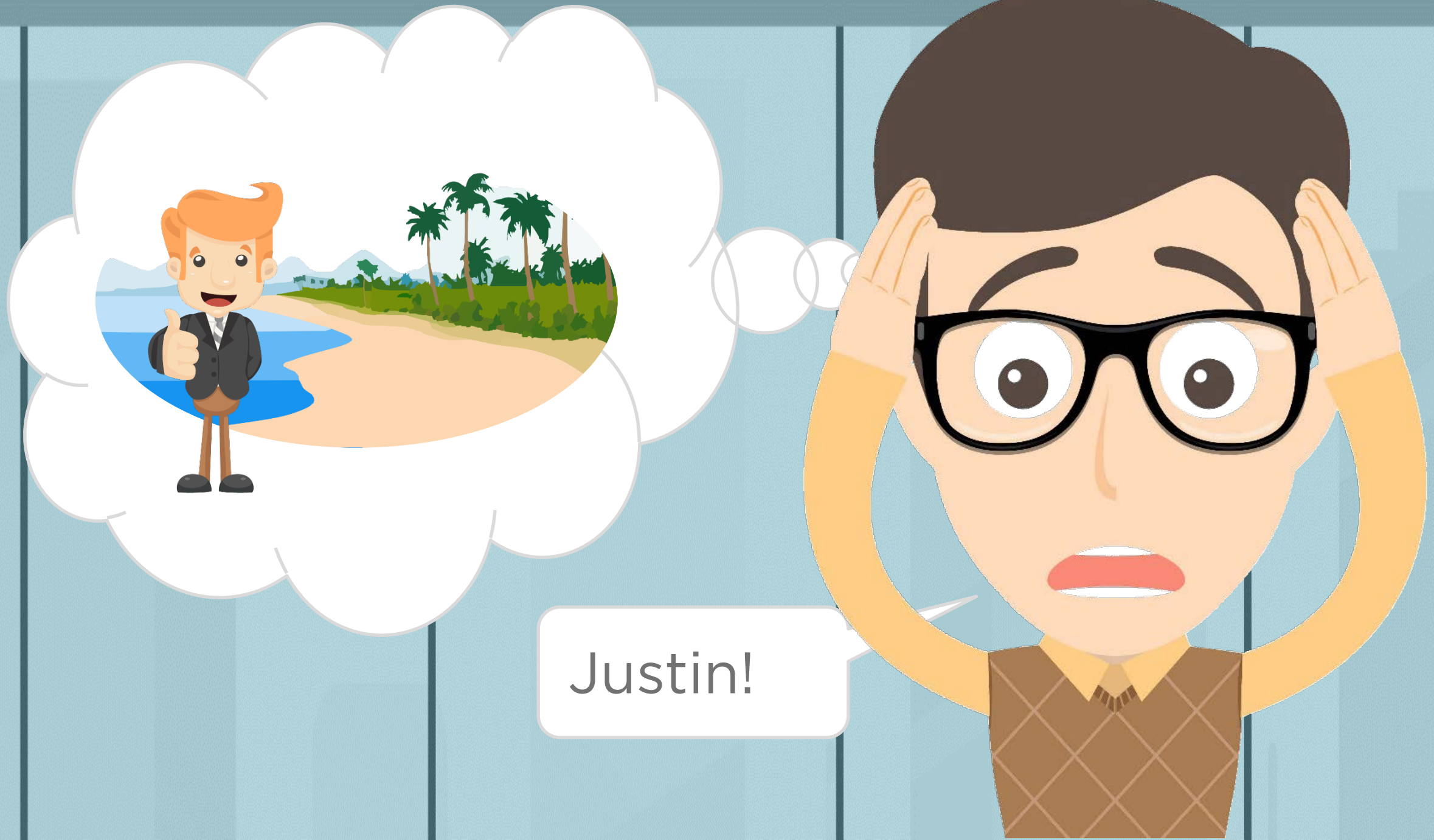
The users will love everythi about the system.

The system will be totally reliable.

The system should be fast.

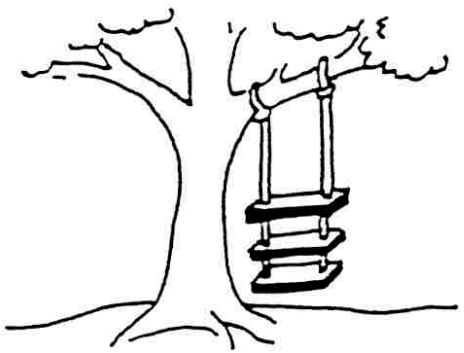
The application ought to be built in a modular way.

The stakeholders better provide sign-off without asking any questions.

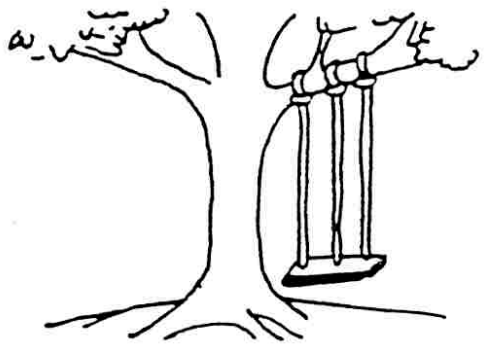

Justin!

Poor Requirements = Project Failure

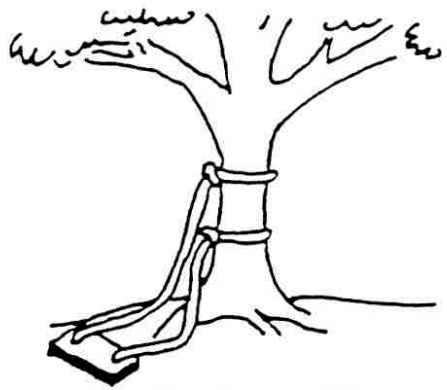
“Problem solving is an art form not fully appreciated by some”



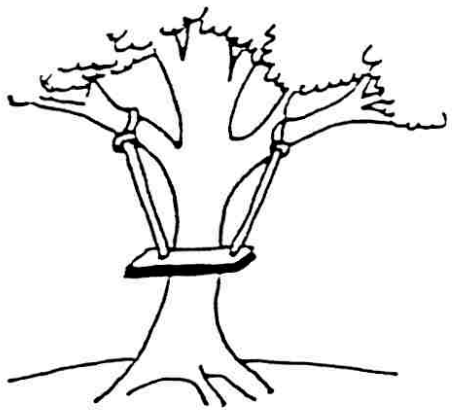
As proposed by the project sponsors



As specified in the project request

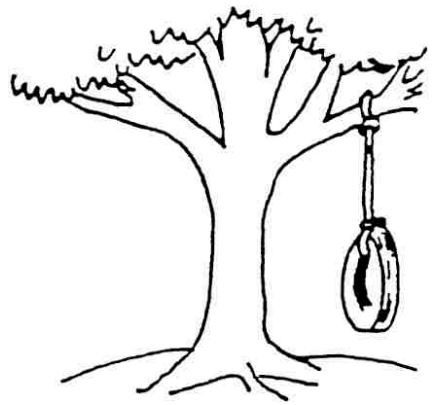


As designed by the senior analyst



As produced by the programmers

As installed at the user's site



What the user wanted

Tree Swing graphic by S Høgh 1993 - from Businessballs.com/treeswing.htm 2013

Standish Group 2015 CHAOS Report

	2011	2012	2013	2014	2015
SUCCESSFUL (on time, on budget, with satisfactory result)	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

Source: Standish Group 2015 CHAOS Report

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

Source: Standish Group 2015 CHAOS Report

Project Impaired Factors	% of Responses
1. Incomplete Requirements	13.1%
2. Lack of User Involvement	12.4%
3. Lack of Resources	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. Changing Requirements & Specifications	8.7%
7. Lack of Planning	8.1%
8. Didn't Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

Source: Standish Group 2015 CHAOS Report

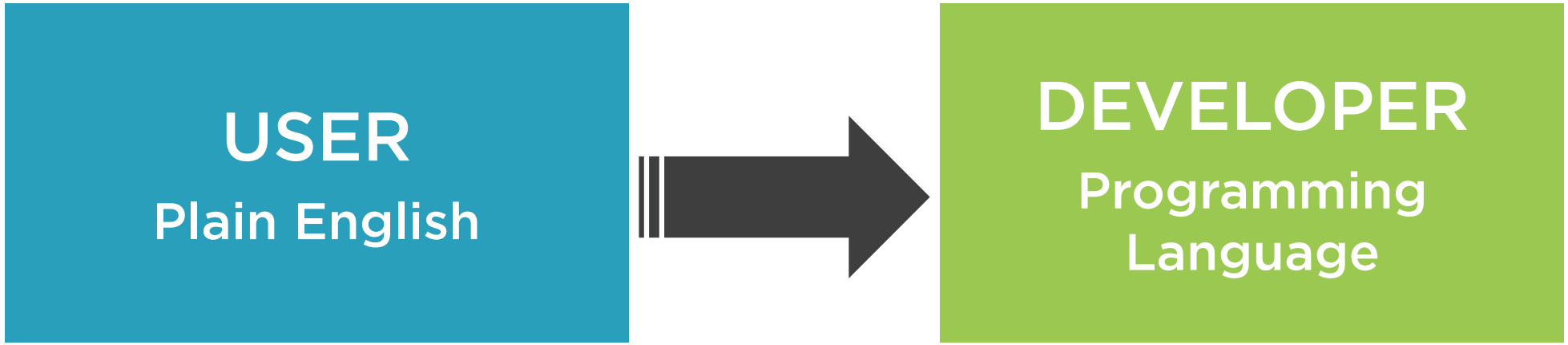
Why Software Projects Fail

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

Source: http://spectrum.ieee.org/computing/software/why-software-fails

Good Requirements = Much Better Chance!

Requirements = Translation



Unitary (Cohesive)	Addresses one and only one thing
Complete	Fully stated in one place, with no missing information
Consistent	Does not contradict any other requirement; fully consistent with all authoritative external documentation
Atomic	Does not contain conjunctions "Validate fields A and B" becomes 1) "Validate field A" 2) "Validate field B"
Traceable	Meets all or part of a business need as stated by stakeholders and is authoritatively documented
Current	Has not been made obsolete by the passage of time
Unambiguous	Concisely stated without use of jargon, acronyms or esoteric verbiage; expresses objective facts; subject to one interpretation
Specify Importance	Specifies a level of importance (defined by stakeholders, time or budget)
Verifiable	Implementation can be determined through inspection, demonstration, test or analysis

The User and System Perspectives

User's Perspective

User type: "The [user class or actor name]..."

Result type: "...shall be able to [do something]..."

Object: "...[to something]."

Qualifier: [response time goal or quality objective]

Example:

The pharmacist shall be able to send an opt-in notification to the patient [within 2 clicks]

System's Perspective

Conditions: "When [some conditions are true]..."

Result: "...the system shall [do something]."

Qualifier: [response time goal or quality objective]

Example:

When a new notification is saved to the database, the system shall send an SMS message [within 30 seconds]

Source: *More About Software Requirements* by Karl Wiegers


Now let’s talk about styles of requirements.


We have three options.


LEVEL OF DETAIL
High
Low
Use Cases
Traditional (Text-Based)
User Stories
LEVEL OF SPECIFICATION
High
Low

Requirement Writing Styles

- User Stories
- Traditional (Text-Based)
- Use Cases

Requirement Types

- Business
- User (Stakeholder)
- Functional
- Non-Functional
- Interface

Regardless of style and type, the goal is to communicate clearly and effectively

Tips for Writing Clear Requirements

Do

- Use terms consistently
- Define terms in a glossary
- Use active voice
- Be careful of boundary values (e.g., "*less than or equal to*")
- Avoid negation

  Before: *Users without an account cannot log into the system*

  After: *Only users with valid accounts can log into the system*

Don't

- Design the system (component names, types of controls, database fields)
- Use vague terms (user-friendly, efficient, high-performance, approximately, several)
- Speculate (usually, often, typically)
- Express possibilities (could, ought to, probably)
- Ramble

Requirement Writing Styles: User Stories

Requirement Writing Styles

- **User Stories**
- **Traditional (Text-Based)**
- **Use Cases**

Requirement Types

- **Business**
- **User (Stakeholder)**
- **Functional**
- **Non-Functional**
- **Interface**

User Story

One or more sentences in everyday or business language that capture what a user needs to do

Format	As a <**type of user**> I want <**some goal**> so that <**some reason**>
Examples	As a <**patient**>, I want <**to receive an SMS message when my prescription is ready to pick up**> so that <**I can avoid unnecessary waiting at the pharmacy**> As a <**pharmacist**>, I want to <**enroll a patient in the SMS notification service**> so that <**they can receive notifications when their prescriptions are ready to pick up**>
Benefits	Brief, bite-size, understandable by users and developers, require little maintenance, iterative, easy to estimate effort
Limitations	Can be vague, open to interpretation, incomplete, lack performance or non-functional details

Used with Acceptance Criteria

User Story

As a <**pharmacist**>, I want to <**enroll a patient in the SMS notification service**> so that <**they can receive notifications when their prescriptions are ready to pick up**>

Acceptance Criteria

- A pharmacist must complete all required fields before submitting the enrollment form
- Information from the form is stored in the enrollment database
- A confirmation SMS message is sent to the patient upon successful enrollment
- A pharmacist can view the enrollment status of a patient

Requirement Writing Styles: Traditional (Text-Based)

Requirement Writing Styles

- User Stories
- **Traditional (Text-Based)**
- Use Cases

Requirement Types

- Business
- User (Stakeholder)
- Functional
- Non-Functional
- Interface

Traditional (Text-Based) Requirements

One or more sentences used to specify high-level functionality for the business or stakeholders

Format	<**Subject doing the action**> <**auxiliary verb**> <**capability or functionality to be provided**> <**criterion that limits or further explains requirement** (optional component)>
Examples	<**The Company**> <**shall**> <**develop an SMS notification system**> enabling patients to <**receive alerts when their prescriptions are available to pick up**> <**The Pharmalantalert system**> <**shall**> provide the ability to <**enroll patients in a notification service**>
Benefits	Can be used to capture complete requirements early in the project
Limitations	May lack enough detail for implementation

Auxiliary Verbs According to IEEE

Word	Indicates
Shall	Mandatory requirements; implies "*is required to*"
Should	Preferred possibility among several; implies "*is recommended that*"
May	A permissible course of action; implies "*is permitted to*"
Can	Used for statements of possibility and capability; implies "*is able to*"
Must	Only used to describe unavoidable situations (not mandatory requirements); implies *"is a natural consequence of"*
Will	Only used in statements of fact (not mandatory requirements); implies *"it is true that"*

Source: https://development.standards.ieee.org/myproject/Public/mytools/draft/styleman.pdf

Requirement Writing Styles: Use Cases

Requirement Writing Styles

- User Stories
- Traditional (Text-Based)
- Use Cases

Requirement Types

- Business
- User (Stakeholder)
- Functional
- Non-Functional
- Interface

Use Case

A list of actions or event steps, typically defining interactions between an actor and a system, to achieve a goal

Format

Use Case Number:	A unique identifier for this use case
Title:	An active-verb goal phrase that names the goal of the primary actor
Description:	Brief description and purpose of use case
Actors:	All actors involved in the use case, both primary and secondary
Scope:	Name of system or subsystem defined by the use case
Priority:	How important is this requirement?
Assumptions:	Any conditions presumed to be true
Preconditions:	State the system must be in for the use case to proceed
Postconditions:	Changes in the environment as a result of the use case
Trigger:	What causes this use case to run
Main Success Scenario:	Step-by-step walk through the use case
Extensions:	Alternative flows and means of achieving the stated goal, including error conditions

Example

Use Case Number:	UC-2.1.5
Title:	Receive and acknowledge notification to opt in to the program
Description:	The patient receives an SMS on their cell phone, indicating they have been successfully enrolled. Upon receipt, the patient replies with a message indicating they accept.
Actors:	Patient
Scope:	Pharmalantalert patient SMS
Priority:	Essential
Assumptions:	Patient is able to receive SMS messages
Preconditions:	A valid phone number for the patient is stored in the system
Postconditions:	Patient has fully enrolled in the program with a double opt-in
Trigger:	Pharmacist enrolls the patient in the notification program

Example

Main Success Scenario:

1. When the pharmacist enrolls the patient in the program, the system sends an opt-in SMS message to the patient's cell phone
2. The patient opens the notification on their phone
3. The patient acknowledges receipt of the message and opts in by responding with "Accept"

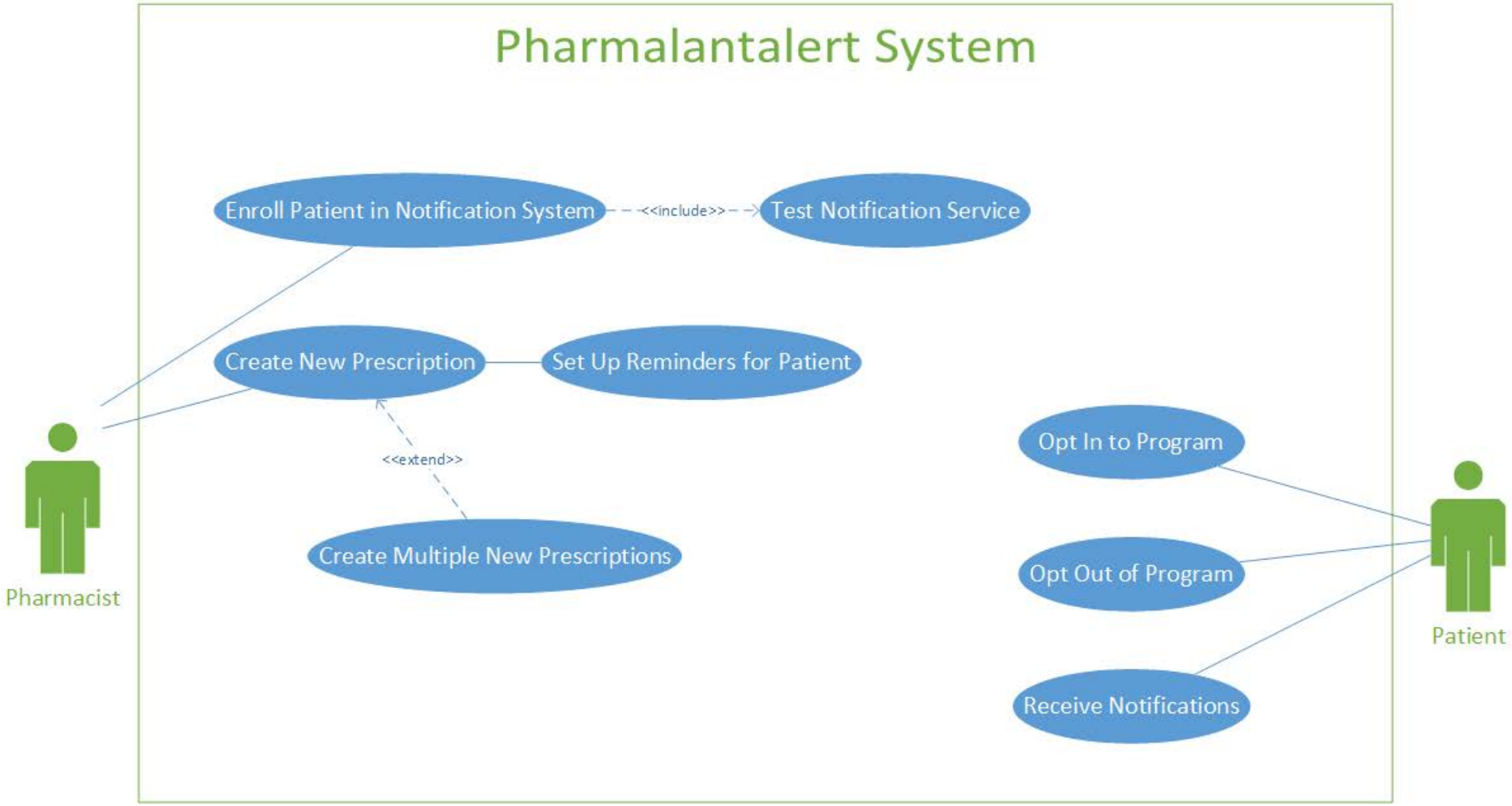
Alternative Scenario:

If the SMS message is not received after one attempt on the patient's device (due to system outage, wrong phone number, etc.), the system will:

1. Log a failure message
2. Discontinue delivery attempts for the current message

Benefits	Robust and comprehensive, up-front research required can be beneficial long term, requires identification of alternative scenarios and error cases
Limitations	Extensive maintenance required, not always suitable for agile development

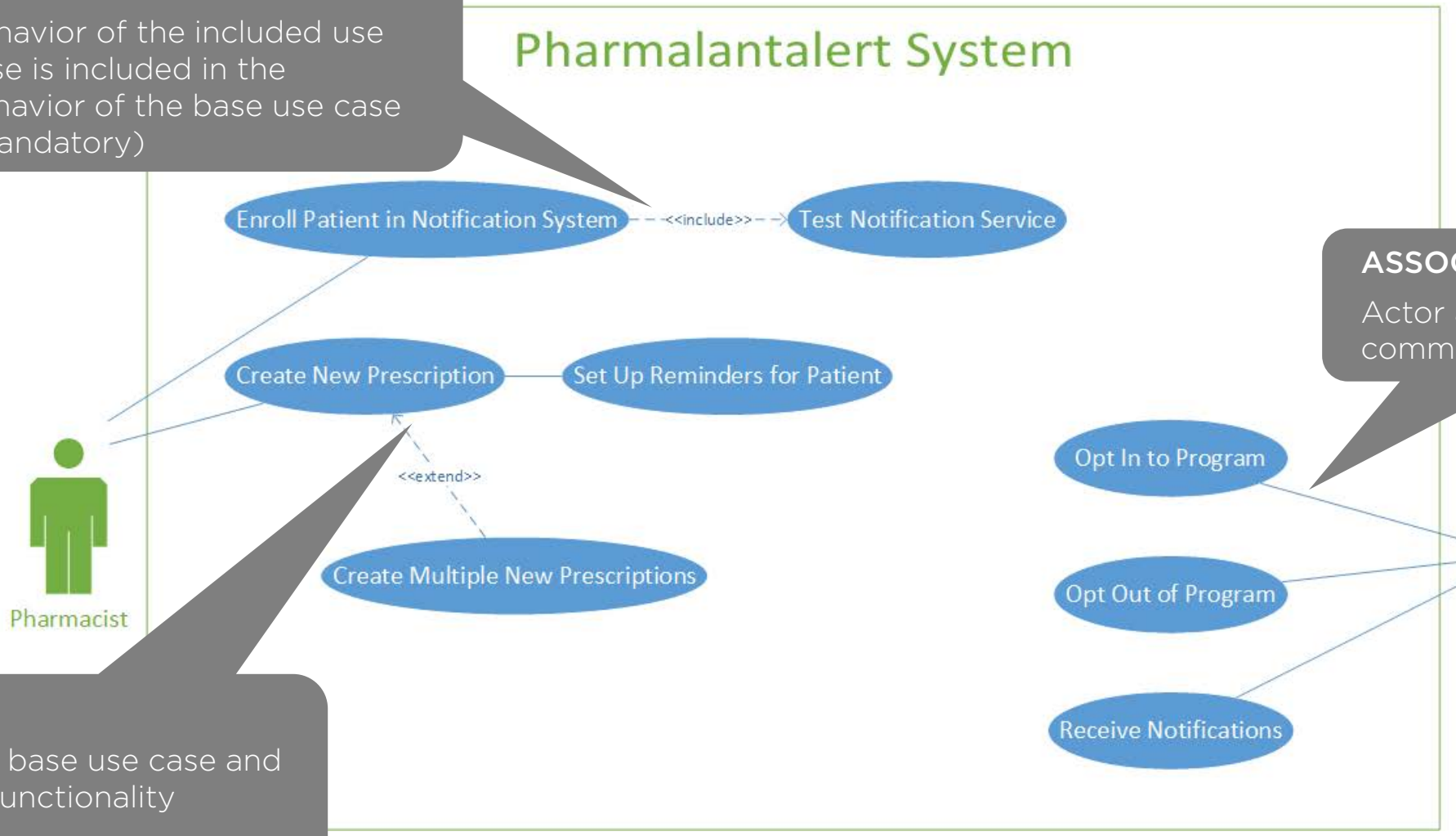
Visual Modeling of Use Cases



Visual Modeling of Use Cases

INCLUDE

Behavior of the included use case is included in the behavior of the base use case (mandatory)



ASSOCIATION

Actor and use case communicate or interact

EXTEND

Extends the base use case and adds more functionality

(optional)

Introduction to UML

By Mike Erickson

This course introduces the Unified Modeling Language (UML) and several of the diagrams that are most often used in software development.

Start free trial now

Table of contents Description Exercise files Transcript Discussion Learning check

Expand all

History, Need and Tools 16m 9s

UML Basics 24m 17s

Structural Diagrams 44m 42s

Behavioral Diagrams 48m 18s

Course info

Level	Intermediate
Rating	★★★★½ (820)
Duration	2h 13m
Released	9 Sep 2013

Course authors

Mike Erickson

Mike is a developer, architect and trainer and has worked with many different tools and technologies for over 20 years. When not working on, learning or sharing something to do with technology he enjoys spending time with his family, especially camping and traveling.

Start free trial now

Share course


Is this all making sense so far?


Yes, it is. But how do I know which style to use and when?


A lot of that depends on the type of requirement.


We’ll look at that next.

Types of Requirements

Requirement types will vary by project

Requirement Writing Styles

- User Stories
- Traditional (Text-Based)
- Use Cases

Requirement Types

- Business
- User (Stakeholder)
- Functional
- Non-Functional
- Interface

Business Requirements

- High-level features of the system
- Describe what will be accomplished for the business (not how)
- Usually requires authority from stakeholders

Traditional (Text-Based)

Provide an SMS notification system for Pharmalantis' customers

User/Stakeholder Requirements

- Describe what will be accomplished for the users/stakeholders

Traditional (Text-Based)

A patient shall be able to receive an SMS notification when their prescription is ready to pick up

User Stories

As a patient, I want to receive an SMS message when my prescription is ready so that I know when to visit the pharmacy

Functional Requirements

- Define the function of the system or its components
  - Behaviors, inputs and outputs

User Story

As a pharmacist, I want to update the patient's phone number so that they can receive SMS messages

Use Cases

UC Number:	UC-2.2.7
Title:	Update patient's phone number
Description:	Pharmacist updates the patient's phone number in the system
Actors:	Pharmacist
Preconditions:	Patient is enrolled in the system with an existing phone number
Main Scenario:	1. Pharmacist navigates to the patient's account
	2. ...

Non-Functional Requirements

- Criteria that define the operating environment in which the functional requirements exist
  - Quality, constraints, non-behavioral

User Stories

As a pharmacist, I want the system to display pages within 3 seconds of navigating to them so that I can be quick and efficient in my job

Traditional (Text-Based)

The system shall display all system pages within 3 seconds of user navigation to them

Use Cases

UC Number:	UC-2.3.3
Title:	Update patient's phone number
	...
Non-Functional:	The system shall display all system pages within 3 seconds of user navigation

Interface

- Describes how one system will interact with another system
  - Hardware, software, communication and user interfaces

Traditional (Text-Based)

The system shall send SMS messages by utilizing the SMS APIs

Use Cases

UC Number:	UC-2.1.3
Title:	Enroll a patient in the notification program
	...
Interface:	The system shall send SMS messages using the SMS APIs

Linking Requirement Type to Writing Style

Type of Requirement	Style
Business	Traditional (Text-Based)
User (Stakeholder)	User Story Traditional (Text-Based)
Functional	User Story Use Case
Non-Functional	User Story Traditional (Text-Based) Use Case User Story
Interface	Traditional (Text-Based)

Software Requirements Specification (SRS)

Software Requirements Specification

Description of a software system to be developed

This is where we get it all down on paper.

Goals of the SRS

- Facilitate reviews
- Describe the scope of work
- Provide a reference to software designers (i.e. navigation aids, document structure)
- Provide a framework for testing primary and secondary use cases
- Links features to customer requirements
- Provide a platform for ongoing refinement (via incomplete specs or questions)

Table of Contents

Revision History

Introduction

- Purpose
- Document Conventions
- Intended Audience
- Product Scope
- References

Overall Description

- Product Perspective
- Product Functions
- User Classes and Characteristics
- Operating Environment
- Design and Implementation Constraints
- User Documentation
- Assumptions and Dependencies

External Interface Requirements

- User Interfaces
- Hardware Interfaces
- Software Interfaces
- Communications Interfaces

Functional Requirements

- System Feature 1
- System Feature 2
- ...

Non-Functional Requirements

- Performance
- Safety
- Security
- Software Quality
- Business Rules

Other Requirements

Glossary


Well, we've made it through our first deliverable of Requirements.


Right on!

Summary and Additional Resources

Additional Resources

Software Requirements (3rd Edition) (Developer Best Practices)

- by Karl Wiegers

Summary

Well-written requirements are key to a successful project

The writing style depends on the project and requirement

- User story
- Traditional (text-based)
- Use case

Regardless of style, clear communication is the goal

Types of requirements include Business, User, Functional, Non-Functional and Interface (among others)

The Software Requirements Specification (SRS) captures it all

Up next >>