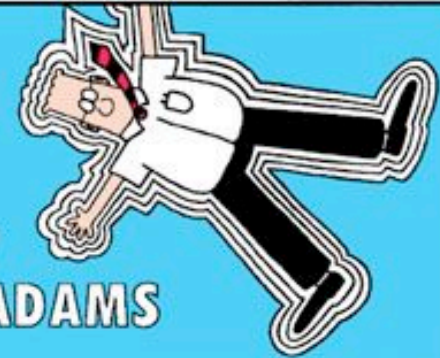


Requirements Elicitation

CSE 308: Software Engineering



DILBERT[®]



BY
SCOTT ADAMS

I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.



E-mail: SCOTTADAMS@AOL.COM

FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?



I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.



© 2006 Scott Adams, Inc./Dist. by UFS, Inc.

I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?



I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.



UFS

TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!



www.dilbert.com

CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?



© Scott Adams, Inc./Dist. by UFS, Inc.

Requirements Engineering

- [Requirement:

- a feature that a system must have

- a constraint that a system must satisfy

- [Requirements engineering: the task of defining the requirements of a system under construction

Requirements Activities

- [Requirements elicitation

- results in a system specification for the client

- harder of the two activities

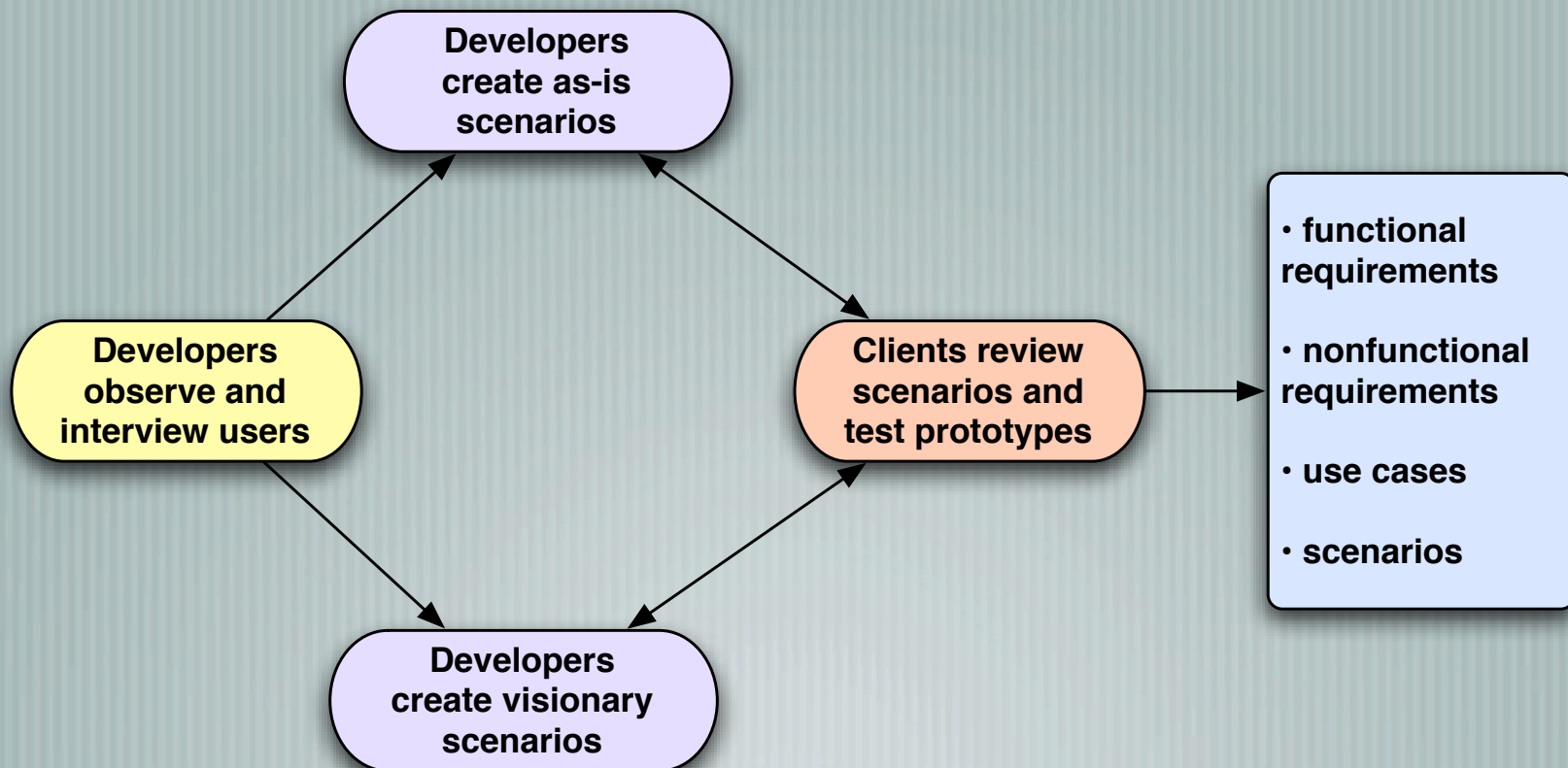
- [Analysis

- results in an analysis model for developers

Bridging the Gap

- [Scenario: describes an example of system use
 - lists a series of interactions between the user and the system
 - “As-is” scenario: describes existing functionality
 - “Visionary” scenario: describes new/desired functionality
- [Use case: describes a class of scenarios
- [Both are written in natural language

Scenario-Based Requirements Elicitation



Requirements Elicitation

- [RE focuses on describing the purpose of a system
- [Requirements specification: a definition of a system that addresses a specific problem area
 - this is formalized to produce an analysis model
- [RE focuses only on the user's view of the system

RE Activities

- [Actor identification
- [Scenario identification
- [Use case identification
- [Use case refinement
- [Identification of use case relationships
- [Identification of nonfunctional requirements

Major RE Concepts

- [Functional requirements
- [Nonfunctional requirements
- [Completeness, consistency, clarity, and correctness
- [Realism, verifiability, and traceability
- [Greenfield engineering, reengineering, and interface engineering

Functional Requirements

- [Describe interactions between a system and its environment
- [Functional requirements are implementation-independent
- [The environment includes user and other (external) systems

Nonfunctional Requirements

- [Describe aspects of the system that are not directly related to its functional behavior
- [Include many aspects of a system, from usability to performance

FURPS+ Categories

- [Usability

- [Dependability (originally Reliability)

 - includes reliability, robustness, and safety

- [Performance

 - includes response time, throughput, availability, accuracy

- [Supportability

 - adaptability, maintainability, and portability

FURPS+ Pseudo-Requirements

- [Implementation requirements
- [Interface requirements
- [Operations requirements
- [Packaging requirements
- [Legal requirements

The Four Cs

- [Completeness: all possible scenarios are described
- [Consistency: no two requirements contradict one another
- [Clarity: the specification is unambiguous
- [Correctness: accurate representation of client's wishes

Realism, Verifiability, Traceability

- [Realism: the system can be implemented within constraints
- [Verifiability: repeatable tests can demonstrate that the system fulfills its requirements
- [Traceability: each requirement can be traced through SW development to its corresponding system functions, and vice versa
 - this is critical for testing and change evaluation

*Engineering

- [Greenfield engineering: development starts from scratch
 - triggered by user need or creation of a new market
- [Reengineering: redesign/implementation of existing system
 - may extend functionality, but purpose remains the same
- [Interface engineering
 - Redesign of the UI of an existing project

RE Activities

- [Actor identification
- [Scenario identification
- [Use case identification
- [Use case refinement
- [Identification of use case relationships
- [Identification of nonfunctional requirements

Actor Identification

- [Actor: an external entity that interacts with a system
 - Actors are role abstractions (not necessarily people)
- [This activity defines system boundaries and finds perspectives from which the system must be considered
- [Initially, actors may be confused with system objects

Scenario Identification

- [Scenario: concrete, focused description of a single system feature from a single actor's point of view
 - scenarios cannot contain branches
- [Scenario types: as-is, visionary, evaluation, training
- [Developers and users write and refine scenarios as part of RE

Use Case Identification

- [A scenario is an instance of a use case
 - a use case represents a complete flow of events
- [Use cases contain entry and exit conditions, event flow descriptions, and quality requirements
- [Use cases and scenarios are critical for early validation of system requirements and functionality

Actor-Use Case Relationships

- [Communication relationships: represent information flow
 - also serve to clarify access control
- [Extend relationships: under certain conditions, one use case includes the behavior of another
 - allows for streamlining and distinction of exceptions
- [Include relationships: factor out redundant/shared behavior

Extend vs. Include

- [Extend: extending use case lists extended source use case as a precondition
 - ex. driving to 3VI extends driving to SBU
- [Include: source use case lists triggering event (for included use case) as part of its flow of events
 - ex. driving to SBU includes stopping at red lights

Extend vs. Include

- [Use extend relationships for exceptional, optional, or occasional behavior
 - ex. resource failure
- [Use include relationships for behavior that is shared across two or more use cases

Identifying Analysis Objects

- [Problem: clients and developers use different terminology
- [Solution: developers should create a glossary that unambiguously names and describes participating objects
- [Participating objects are identified for each use case

Object Identification Heuristics

- [terms that users must clarify to understand the use case
- [recurring nouns in the use cases
- [real-world entities that must be tracked
- [real-world processes that must be tracked
- [use cases
- [data sources/sinks
- [application domain terms

Nonfunctional Requirements

- [The set of nonfunctional requirements usually includes conflicting requirements
 - ex. physical robustness vs. low cost
 - we resolve this by prioritizing requirements
- [In practice, analysts use taxonomies (e.g. FURPS+) to create checklists of nonfunctional requirements

RE Management

- [Joint Application Design (JAD)
- [Maintaining traceability
- [Documenting RE

Joint Application Design

- [All requirements elicitation work is done in a single workshop session
- [Users, clients, developers, and session leader come together
- [Outcome is a complete requirements specification document

JAD Activities

- [Project definition
- [Research
- [Preparation (creation of working document)
- [Session (lasts 3-5 days)
- [Final document (production of revised specification)

Traceability

- [Ability to follow the life of a requirement
 - where it came from
 - which parts of the system it affects
- [Simplest approach: document cross-references
- [Specialized database tools also exist to solve this problem

Documenting RE

- [Requirements Analysis Document (RAD)

- completely describes a system in terms of functional/
nonfunctional requirements

- [Sections: introduction, current system, proposed system
(overview, functional requirements, nonfunctional
requirements, system models)

- written after the use case model is stable

Case Study: "Alpha Quadrant"

- [Goal: to develop a *Massively Multiplayer Online Role-Playing Game (MMORPG)* set in the "Star Trek" universe
- [We need to analyze this idea and generate a set of requirements (including scenarios/use cases)

Game Features

- [Cooperative and Player vs. Player modes

- PvP should be optional

- [Multiple payment options

- Flat monthly rate plus per-session pricing structure

- [“Marketplace” based on real money

- Players can use game money and real-world money to buy equipment, weapons, etc.

Game Features

- [Multiple platforms

- can play on PC/Mac, XBox/PS2/GC, cell phone, Web

- [Players can create new content for the game

Assignment

- [Think about the requirements for “Alpha Quadrant”
- [Create a few (1-3) scenarios for user interaction
 - We’ll use these scenarios to develop use cases

Next Time

- [We'll work through some use cases for "Alpha Quadrant"
- [The Analysis phase of development