System Requirements

CSE 403, Spring 2004

Software Engineering

http://www.cs.washington.edu/education/courses/403/04sp/

References

- References
  - » *The Mythical Man-Month*, Brooks
  - » Chapter 7, Before the Project, *The Pragmatic Programmer*, Hunt & Thomas
  - » *Structuring Use Cases with Goals* , A. Cockburn
    - http://alistair.cockburn.us/crystal/articles/alistairsarticles.htm
  - » *Use cases in theory and practice*, A. Cockburn
    - http://alistair.cockburn.us/crystal/articles/alistairsarticles.htm

System Requirements

- Essential features of the system
  - » defined at a level appropriate to the spin cycle
  - » capabilities, interfaces, reliability levels, appearance
  - » Easy to change early on, grows increasingly more difficult
- Customer's involvement very important
  - » they know the domain of interest far better than you do
  - » what fits with their daily work and life patterns
  - » what might the future bring
- Neither you nor the customer know everything
  - » try to build joint ownership of the process
  - » open communication can make change more acceptable

What does the customer want?

- Better products for free
  - » Scott Adams
- Many customers exist for any single product
  - » purchaser, user, user's management, support, etc
- Write down attributes of expected user set
  - » Who they are
  - » What they need
  - » What they think they need
  - » What they want

Attributes have a distribution

- Attributes of the user set are distributions
  - » many possible values
  - » each value with its own frequency
- The design will not meet all requirements of all members of the user set all the time
  - » Postulate a complete set of attributes and frequencies
  - » Develop complete, explicit, shared description of users
  - » It is better to be explicit and wrong than to be vague

FPBrooks, MMM

"Complete" Requirements

- You want to write down every requirement for every user of every aspect of the program
  - » It's not possible, there isn't enough time or money
- You have to find a balance
  - » comprehendible vs. detailed correctness
  - » graphics vs. explicit wording and tables
  - » short and timely vs. complete and late
- Different approaches for different parts are okay

Modularity, not a "pile of paragraphs"

- Split the information by point of view and adapt the documentation style as appropriate
  - » Business functions
    - top level mission of application (text, graphics, Flash?)
    - specific functions that must be implemented (use case)
  - » Context
    - drawings, text, references to interface standards
  - » User Interface
    - text goals, sample layouts, some prototypes
  - » Performance and Reliability
    - text goals, specific metrics for space, time, CPUs, ...

Concise is nice

- *All* the details are necessary at some point
  - » but only *some* of the details are relevant right now
- Arrange the requirements so that the reader can drill down in areas of interest without having to pick out the details from chaos
  - » Data flow graphics for top-level orientation
  - » Tabular presentation of specific metrics
- The lower the level, the more structured
  - » eg, Scenarios vs. Use Cases

Use Cases

- Use cases address "how to make functional requirements readable, reviewable"
  - » As an expression "use case" is immediately attractive because the term implies "the ways in which a user uses a system"
- "I have personally encountered over 18 different definitions of use case", A. Cockburn
- "True use cases are textual descriptions, with a hierarchy and cross-links.", Hunt & Thomas

Use case dimensions

- Purpose
  - » To gather user stories, or build requirements?
    - values are *stories*, or *requirements*
- Contents
  - » Consistent, or can they be self-contradicting?
    - *contradicting*, *consistent prose*, *formal content*
- Plurality
  - » Does a use case contain more than scenario?
    - *1* or *multiple*
- Structure
  - » Informal structure or formal structure?
    - *unstructured, semi-formal, formal structure*

A. Cockburn

One choice

- Consistent, semi-formal documentation of requirements
  - » Purpose = requirements
  - » Contents = consistent prose
  - » Plurality = multiple scenarios per use case
  - » Structure = semi-formal

A. Cockburn

What is a use case?

- Sequence of interactions between the system under discussion and its external actors, related to a particular goal
  - » An action connects one actor's goal with another's responsibility
  - » An interaction is simple or compound
  - » Scenarios and use cases go until goal success or abandonment

A. Cockburn

Figure 7.1. Cockburn's use case template

A. CHARACTERISTIC INFORMATION
- Goal in context
- Scope
- Level
- Preconditions
- Success end condition
- Failed end condition
- Primary actor
- Trigger

B. MAIN SUCCESS SCENARIO
C. EXTENSIONS
D. VARIATIONS
E. RELATED INFORMATION
- Priority
- Performance target
- Frequency
- Superordinate use case
- Subordinate use cases
- Channel to primary actor
- Secondary actors
- Channel to secondary actors

F. SCHEDULE
G. OPEN ISSUES

Sample use case

Figure 7.2. A sample use case

USE CASE 5: BUY GOODS

A. **CHARACTERISTIC INFORMATION**
- **Goal in context:** Buyer issues request directly to our company, expects goods shipped and to be billed.
- **Scope:** Company
- **Level:** Summary
- **Preconditions:** We know buyer, their address, etc.
- **Success end condition:** Buyer has goods, we have money for the goods.
- **Failed end condition:** We have not sent the goods, buyer has not sent the money.
- **Primary actor:** Buyer, any agent (or computer) acting for the customer
- **Trigger:** Purchase request comes in.

B. **MAIN SUCCESS SCENARIO**
1. Buyer calls in with a purchase request.
2. Company captures buyer's name, address, requested goods, etc.
3. Company gives buyer information on goods, prices, delivery dates, etc.
4. Buyer signs for order.
5. Company creates order, ships order to buyer.
6. Company ships invoice to buyer.
7. Buyer pays invoice.

Pragmatic Programmer

C. **EXTENSIONS**
- 3a. Company is out of one of the ordered items: Renegotiate order.
- 4a. Buyer pays directly with credit card: Take payment by credit card (use case 44).
- 7a. Buyer returns goods: Handle returned goods (use case 105).

D. **VARIATIONS**
1. Buyer may use phone in, fax in, Web order form, electronic interchange.
7. Buyer may pay by cash, money order, check, or credit card.

E. **RELATED INFORMATION**
- **Priority:** Top
- **Performance target:** 5 minutes for order, 45 days until paid
- **Frequency:** 200/day
- **Superordinate use case:** Manage customer relationship (use case 2).
- **Subordinate use cases:** Create order (15). Take payment by credit card (44). Handle returned goods (105).
- **Channel to primary actor:** May be phone, file, or interactive
- **Secondary actors:** Credit card company, bank, shipping service

F. **SCHEDULE**
- **Due date:** Release 1.0

G. **OPEN ISSUES**
- What happens if we have part of the order?
- What happens if credit card is stolen?

Pragmatic Programmer

Overspecifying

- The simplest statement that accurately reflects the business need is best
  - » Requirements are not architecture or design
  - » Requirements are *need*
- Overspecified requirements are dangerous
  - » they cannot be understood
  - » they will not be read
  - » they will rot
  - » and they will be wrong

Requirements are fun

- This is the time when you have the most leverage to create a successful project
  - » you can change direction with the stroke of a pen
  - » you can re-architect the moment you gain a deeper understanding of the true need
  - » you can apply all the design tools and experience in your tool chest to finding ways to enable what is now only a dream for the customer
- Plus, you learn about a new knowledge domain!