Building Use Cases

Craig D. Wilson

MATINCOR, INC.

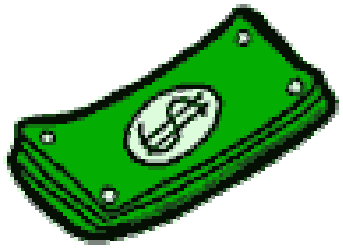
Outline

- Use Case overview
- Integration with development process
- Constructing scenarios
- Other components
- Use Case process
- Benefits & Limitations

What are Use Cases?

- Text documents - Defined content, flexible context
  - 3x5 index cards ("stories" in eXtreme Programming)
  - Formal word processing templates with fill-in-the-blanks sections
- Each describes 1 system function which is of value to user
- Present various "user experience" scenarios
- Detail evolves thru iterative "drill downs"

Function of Value



- Interaction which provides visible function of value to user
  - ATM: withdraw cash
  - Video Store: rent DVD
- A project's initial scope is defined by list of high-level use cases
  - ATM Example
    - Included: withdraw cash, deposit cash, transfer
    - Not Included: open account, write check, get loan

Detail Evolves

- System's initial UC's are high-level:
  - ATM Use Cases:

Withdraw cash	Deposit cash
Transfer funds	Mini-statement

- During drill-downs, UC's may break down into sub-use cases:
  - Withdraw cash Use Case:

Authentication	Account update
Print statement	Eject card

Usage Scenarios

- Present different ways that UC can be executed
  - Successful:
    - Withdraw cash
    - Withdraw cash after re-entering PIN number
  - Unsuccessful:
    - Wrong PIN number
    - Closed account
    - Insufficient funds

Happy Day

Gloomy Day

Use Cases & Software Engineering Process

- Established early in project lifecycle to define project scope
- Evolve through conceptual design phase
- Vehicle for capturing & expressing system functionality
  - as experienced by user
  - as experienced by system

Standard Project Approach

- Scope the project
  - Identify initial Use Cases
  - Present relationships as Use Case Diagram
- Capture Requirements
  - List Use Cases w/ requirements checklists
- Prepare Conceptual Design
  - Prepare Use Case scenarios w/ evolved UC diagram
- Drill-down
  - Identify and detail sub-Use Cases

Use Cases – input for what?

- ✓User interface design
- ✓System architecture
- ✓Test cases
- ✓User manuals
- ✓Training
- ✓Infrastructure design
- ✓Business process re-engineering

Scenarios

- Detail the "experience" that the ***actor*** will have performing the use case
- Provide complete scenario for most commonly expected execution
  - Provide *alternate* scenarios for other experiences which still result in successful conclusion
  - Provide *exception* scenarios for unsuccessful conclusions

ATM "Withdraw Cash" Scenario

	Actor	System
	Input ATM card	Request PIN
	Enter PIN	*Authenticate user* Display Transaction menu
	Select "withdraw cash"	Display "which account?" then list accounts
	Select appropriate account	Display "how much?"
	Enter $300	Display "working" *Verify sufficient funds* *Debit account*
		Dispense cash

ATM "Withdraw Cash" Alternate

	Actor	System
	Enter PIN	PIN not recognized
	Re-enter PIN	Display transaction menu

ATM "Withdraw Cash" Exception

	Actor	System
	Enter PIN	PIN not recognized. Request re-entry
	Re-enter PIN	PIN not recognized. Request re-entry
	Re-enter PIN	PIN not recognized. Display message "call 555-1234". Eject ATM card

Scenario Guidelines

- Be concise
- Describe the interaction but not the details of what is going on "behind the scenes"
- Use terms understandable to the actor
  - For high-level UC's, use business terms
  - For detailed UC's, use terms associated with the problem domain (business, architectural, code)

Alternates and Exceptions

- No need to re-enter entire scenario
  - Only describe difference from main scenario unless *very different*
- Alternates and Exceptions are one of the most frequently missed "requirements" of a system
  - If not captured during analysis, may not be discovered until well into coding
  - Required for effective QA testing effort

Use Annotation for Clarification

- Add notes which impact entire use case or individual steps. For example:
  - Allow "cancel" at any point prior to commit
  - System response in less than 2 seconds
  - Transaction "times out" after 10 minutes

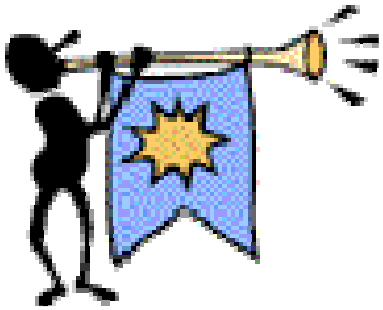
Additional UC Information

- Actor
- Trigger
- Performance
- Frequency
- Pre-condition
- Post-condition
- Business rules
- Data elements
- Screen mock-ups

Actor

- Defines role of person who will perform the UC
  - Can also be a system or clock
- Identifies the recipient of the UC's "function of value"
- Is not usually a person's name or title
  - Individuals and systems can play multiple roles
  - e.g., a video store's employee can be a "clerk" or "manager". The manager can also act as a clerk.
- There can be "sub-actors" who interact with UC but who are not the recipient of the UC's "function of value"

Trigger

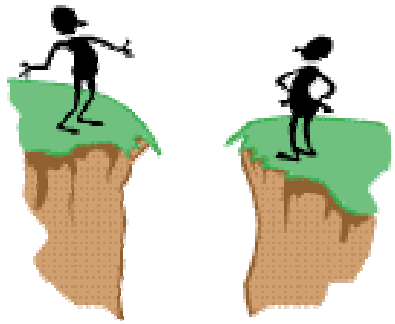


- The action which initiates the UC
  - ATM customer enters ATM card in slot
  - Video clerk scans DVD for return
  - Clock initiates system backup
  - Finance system begins check printing

Performance & Frequency

- Performance
  - How long the UC should take to execute
    - e.g., how long should the average "withdraw cash" transaction take?
- Frequency
  - How often will the UC be executed?
    - e.g., How many "withdraw cash" transactions will occur in an average day? What is the most transactions that the system should be able to handle in a day? How many concurrent withdrawals can occur?

Pre & Post Conditions



- Pre-conditions
  - System's state to execute the UC?
    - Before a "withdraw cash" transaction can occur:
      - The person must have a valid ATM card
      - The person must have been validated by the system
- Post-conditions
  - System's state at end of UC?
    - Receipt printed
    - ATM card ejected
    - Screen displays "Welcome" page

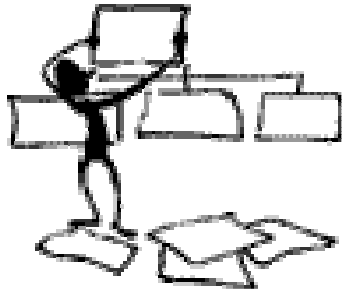
Business Rules

- Provide rules, limitations, algorithms, or any other “system logic”
- Provide Actor requirements such as access rights, authority, system interface limitations, etc.
- Security requirements

Data Elements

- Define data elements including:
  - Type
  - Length
  - Valid domain sets
- Differentiate between new, existing, and modified data elements

Screen / Report Mock-ups



- Provide mock-ups of all user interfaces
  - Mock-ups – not detailed designs!
  - Can be done prior to, during, or after writing the scenarios
    - This is contrary to the text books which state this should be done ***after*** the use case scenarios
    - Depends upon numerous factors including
      - Is this a new system or maintenance?
      - Is this heavily dependent upon UI experience?
      - Are users more comfortable working with text or pictures?
    - Will probably be an iterative approach; UC, GUI, UC, GUI..

UC Creation

- Meeting with project team
  - Extremely interactive – use brainstorming techniques
- Straw man may help get discussion going
- Iterative process
  - First meeting may be just to identify UC's
  - Next meetings may focus on "happy day" scenarios
  - UI's may be addressed in separate meeting
- Have fun! This is the time to explore alternatives!

Participants

- Analysts (lead and team)
- Subject matter experts (users during business level)
- Technical lead
- GUI designer (when system deals with human interaction)
- Quality Assurance lead
- Optional
  - Project manager
  - Configuration Management lead
  - Infrastructure lead

UC Capture

- Most frequently in a word-processing document
- May be captured in requirements database
- Requirements may be cross-indexed with UC

Benefits

- Provide a way for entire team to communicate their "vision" of the solution
- Keeps team focused on functions of value rather than code structures
- Supports Object Oriented Analysis and Design techniques
- Provides usable input for analysts, architects, testers, technical writers, trainers, etc.

Limitations

- Not a system architecture
  - System still requires modeling
- Beware of "analysis paralysis"
  - Get just enough to identify architecturally significant use cases then document those
  - Get system modeling and architecture going while secondary and supplemental UC's are being prepared

Discussion & Questions