More about Inception, Requirements Analysis and Use Cases

UP is iterative & incremental

- Development is organized in a series of short, fixed-length mini-projects called iterations
- Iterations are also incremental
- Successive enlargement and refinement of a system
- Feedback and adaptation evolve the specification, design and code
- How might iterative development be different from prototyping?
- Output of each iteration need not be experimental or a throw-away prototype
- Each iteration tries to be a production-grade subset of final system

A motto for requirements

- **Le mieux est l’ennemi du bien**
  – Voltaire
  (The best is the enemy of the good.)
- Why?
- Avoid *“Paralysis by Analysis”* – kills budget without significant benefit
- Classic mistake: Too much time and money wasted in the “fuzzy front end”

Early feedback is worth its weight in gold

- Each iteration involves choosing a small subset of requirements, and quickly designing, implementing and testing
- Early feedback (from users, developers and tests) drives development

Evolutionary requirements

- Requirements are capabilities and conditions to which the system and the project must conform

- A prime challenge of requirements analysis is to find, communicate, and remember what is really needed, in the form that clearly speaks to the client and development team members.

DEFINITION: Use Case

- A story of using a system fulfilling a goal
  - E.g., Deposit cash
  - A use case story consists of a set of alternative scenarios
- Actors are capable of active behavior
  - E.g., Person, computer system, organization
- *Primary actors* have goals that use case accomplish
  - E.g., Customer, Clerk
- *Supporting actors* help use case accomplish goal
  - E.g., Bank, Database

Fully-dressed use case

See alistar.cockburn.us

- Use case name
- Scope
- Level (user-goal or subfunction)
- Actors: Primary, Secondary
- Stakeholders and interests (who cares about this use case, and what do they want?)
- Preconditions (what must be true on start)
- Postconditions or Success guarantee (what must be true on successful completion)
- Main success scenario (typical path, happy path)
- Extensions (alternate scenarios of success and failure)
- Special requirements (related non-functional requirements)
- Technology and data variations list (varying I/O methods)
- Frequency of occurrence
- Miscellaneous

What behavior should we model with a use case?

- Cockburn: Elementary Business Process (EBP) guideline:
  - *"A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state."*
- Naively, can you apply the "boss test" for an EBP?
  - Boss: "What do you do all day?"
  - Me: "I logged in!"
  - Is Boss happy?
- Size: An EBP-level use case *usually* is composed of several steps, not just one or two.

Use Case Levels: Applying the Guidelines

- *Which of following meets EBP & size guidelines?*
    - Negotiate a Supplier Contract
    - [✓ Rent Videos]
    - Log In
    - Start Up
- The others *can* also be modeled as use cases
    - But focus first on essential cases (EBP level)

GUIDELINES: Use Case Modeling

- Keep use case names simple: Verb object
  - Deposit money.
  - Not: Deposit money into checking. Why not?
- Accomplish a user's goal
  - Invalid PIN is not a use case. Why not?
- Include Secondary Actors (e.g., Bank)
- Avoid ambiguity
  - E.g., in the ATM problem, System could be the machine or the Bank's back-end server
- *Start Up* and *Shut Down* are use cases
  - Why do we usually not include them at first?

More use case guidelines

- A use case diagram is not a flow chart
- Steps in the use case (such as enter PIN) are not necessarily use cases. Why not?

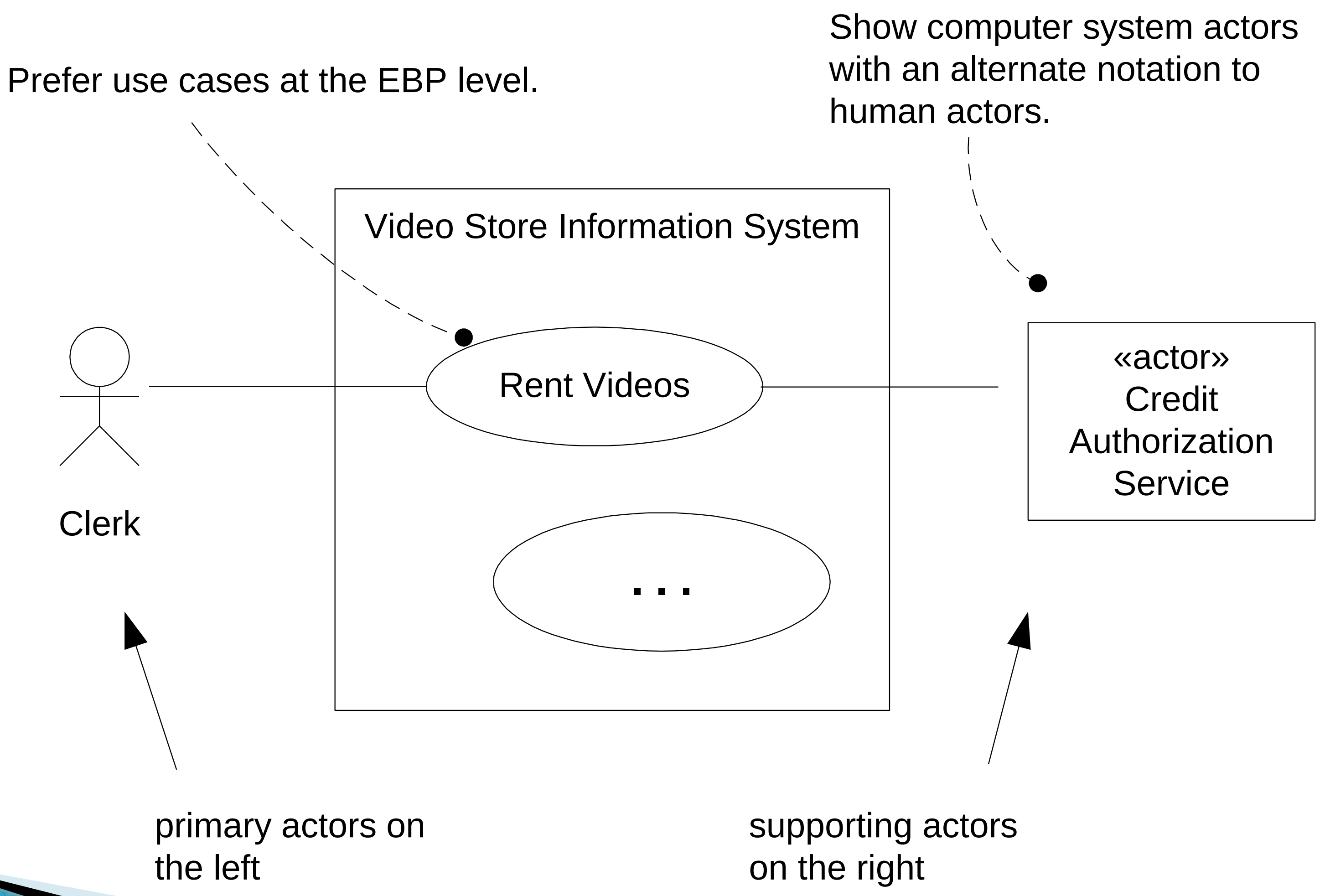
- Keep each step and alternative simple; e.g., don’t validate PIN and balance in same step (and same alternative scenario)

- Transactions (such as deposit money and withdraw cash) are candidate use cases. Why?

Use Case Diagrams

- UML has use case diagrams
- Use cases are *text*, not diagrams
- But a *short* time drawing a use case diagram provides a context for:
  - identifying use cases by name
  - creating a “context diagram”
- Again, a use case diagram is not a flow chart!

GUIDELINES: Use Case Diagrams


Prefer use cases at the EBP level.
Show computer system actors with an alternate notation to human actors.
Video Store Information System
Rent Videos
. . .
Clerk
«actor» Credit Authorization Service
primary actors on the left
supporting actors on the right

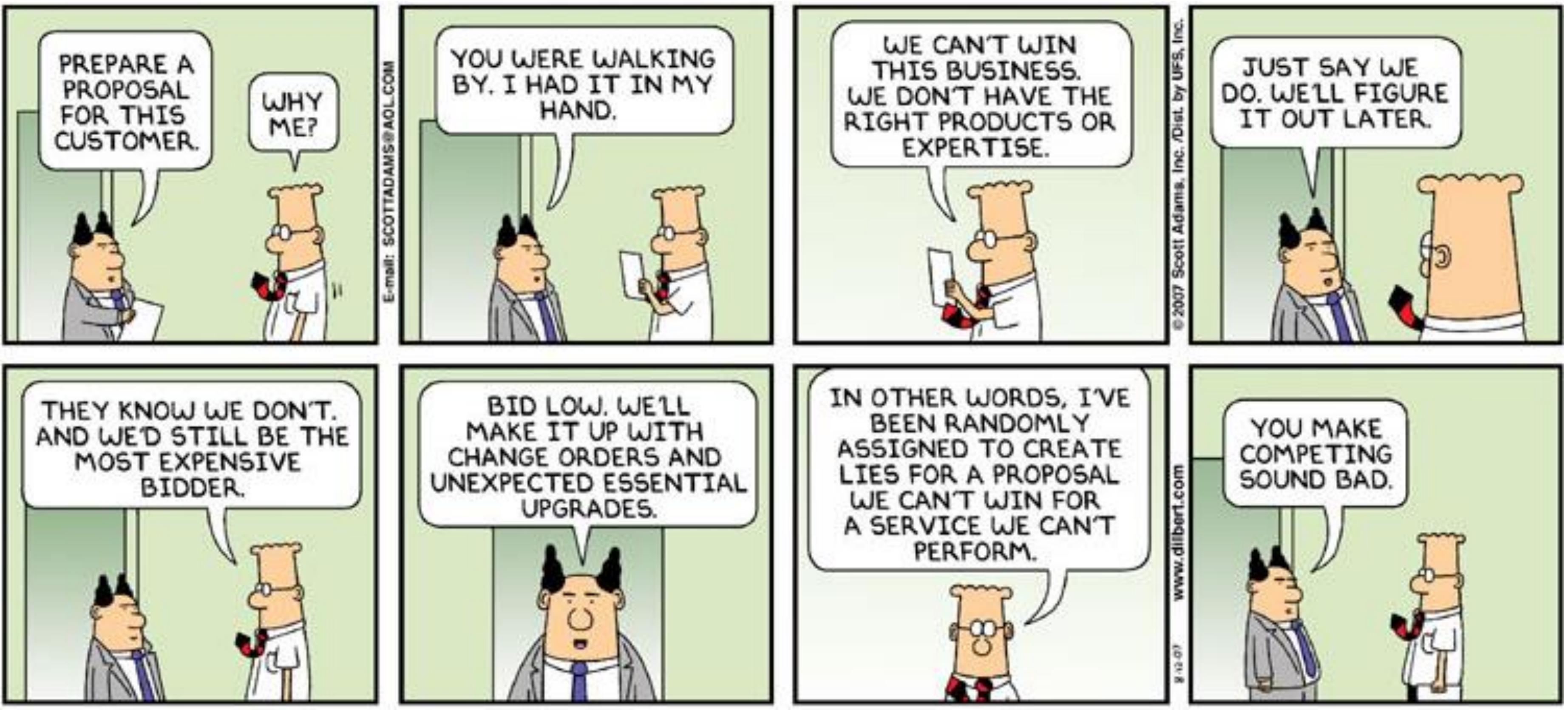
Supplementary Specification

- Use cases describe functional requirements
- Supplementary Specification (SS) captures non-functional reqs (URPS+):
- Vision and Scope
- Features list
- Glossary (Data Dictionary)
- Business Rules
- Risk plan
- Iteration Plan

Feature list

- Feature is a behavioral function a system can do
- A feature is an externally visible service
  - E.g., system does investment rate of return
  - System does credit risk analysis
- Why is a feature list useful when developing a Vision and Scope document?
- Prefer short (10-12) feature list of most valuable benefits

Is honesty the best policy?


PREPARE A PROPOSAL FOR THIS CUSTOMER.
WHY ME?
YOU WERE WALKING BY. I HAD IT IN MY HAND.
WE CAN'T WIN THIS BUSINESS. WE DON'T HAVE THE RIGHT PRODUCTS OR EXPERTISE.
JUST SAY WE DO. WE'LL FIGURE IT OUT LATER.
THEY KNOW WE DON'T. AND WE'D STILL BE THE MOST EXPENSIVE BIDDER.
BID LOW. WE'LL MAKE IT UP WITH CHANGE ORDERS AND UNEXPECTED ESSENTIAL UPGRADES.
IN OTHER WORDS, I'VE BEEN RANDOMLY ASSIGNED TO CREATE LIES FOR A PROPOSAL WE CAN'T WIN FOR A SERVICE WE CAN'T PERFORM.
YOU MAKE COMPETING SOUND BAD.
E-mail: SCOTTADAMS@AOL.COM
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

Risk Plan

- Contains a list of known and expected risks
- Business, technical, resource, and schedule risks identified by probability and severity
- All significant risks should have a response or mitigation plan

Ranking requirements

Rank requirements as:

- High (score high on all rankings; hard to add late)
- Medium (affects security domain)
- Low

by:

- Risk (includes both technical complexity and other factors, such as uncertainty of effort and usability)
- Coverage (all major parts of the system are tackled in early iterations)
- Criticality (refers to functions the client considers of high business value)

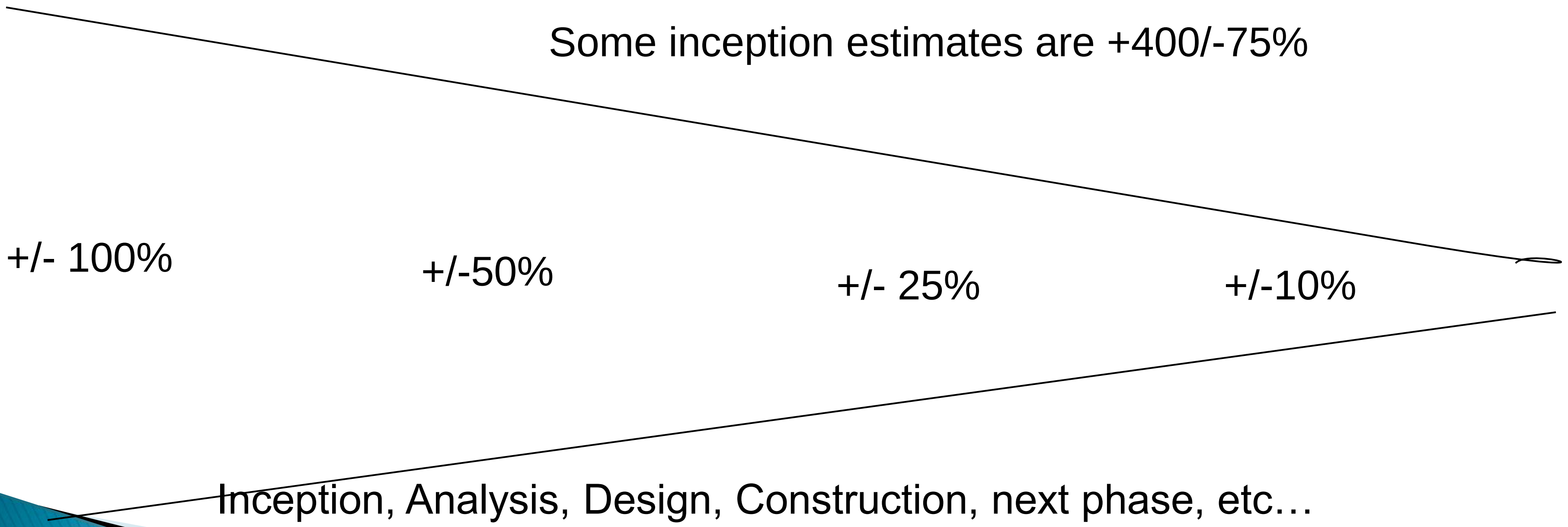
Ranking is done before each iteration

Iteration Plan

- Describes what to do in each iteration of product
- Usually first iteration implements core functionality
- Need to consider risks and make estimates
  - Eliminate biggest risk first
  - Worst risk is usually that the final product will not meet the most important requirement
  - Estimate what can be accomplished in 2-3 weeks

Accuracy of estimates

- There is a funnel of cost estimates
  - The earlier the estimate, the less accurate it is.


Some inception estimates are +400/-75%
+/- 100%
+/-50%
+/- 25%
+/-10%
Inception, Analysis, Design, Construction, next phase, etc…