廣東工業大學
Guangdong University of Technology

软件需求工程

刘文印，liuwy@gdut.edu.cn

广东工业大学网络身份安全粤港联合实验室 WIS Lab

个人微信：csliuwy；微信公众号：wislab；denglu-1或“登录易”

教学资料网站：http://wislab.cn/blockchain

“用户需求”的演化过程

- 用户需要的>
- 用户表达出来的>
- 软件团队能理解的 + 团队的商业目标>
- 软件团队成员具体表达出来的（PM写**Spec**）>
- 在各种约束条件下，具体执行表达出来的（Dev写代码）>
- 验证通过的（Test）>
- 通过各种渠道告诉目标用户（发布/推广）>
- 用户终于能用上了

软件需求工程的步骤

- 获得（Acquisition）
- 描述（**Spec**ification -软件需求规格说明书）
- 审计（Review）
  - 可行性分析等

软件需求获取的一些常用方法

- 联合应用开发方法Joint Application Development (JAD)
- 需求洗涤Requirements Scrubbing
- 敏捷快速原型方法Rapid Prototyping

软件需求获取方法：

Joint Application Development (JAD)

for Requirements Definition & UI Design

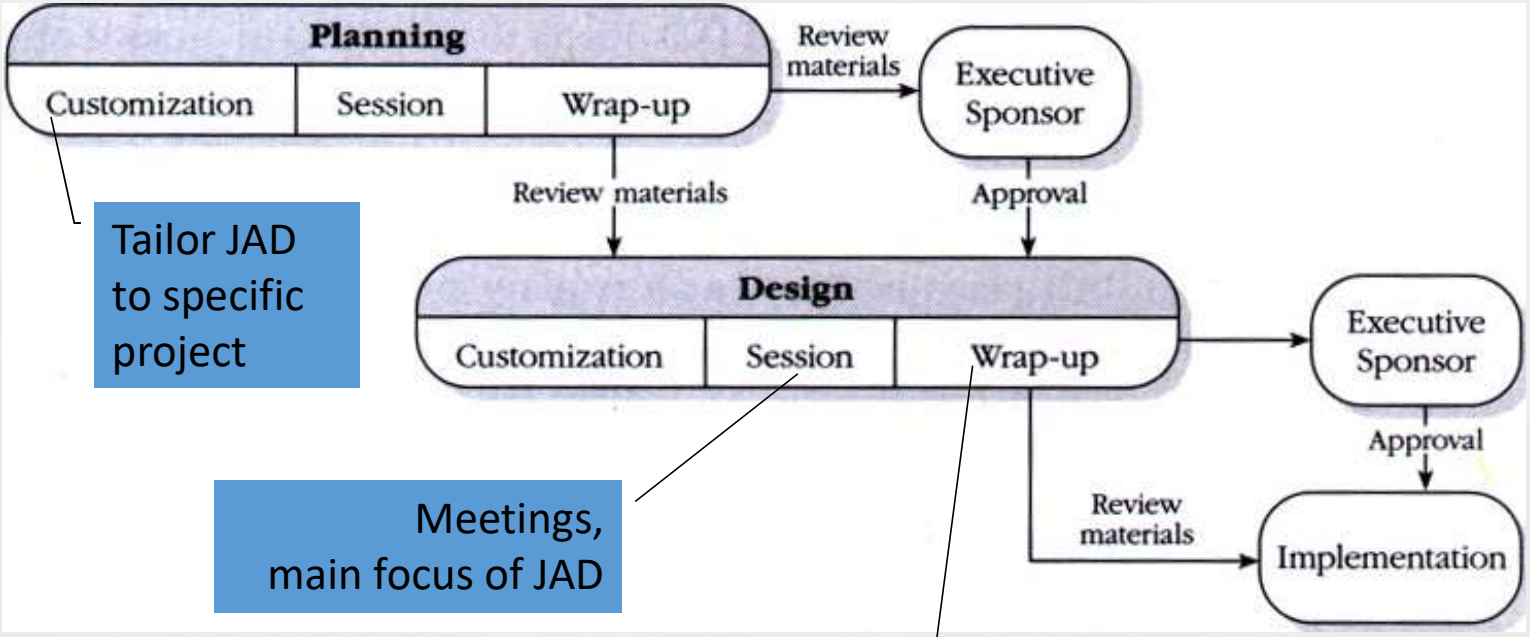
JAD—Definition

- A Variation to FAST, for requirements elicitation
  - Facilitated Application Specification Techniques (FAST)
  - A team-oriented approach to requirement gathering at early stage
- Approach to Requirement Definition & UI Design
  - A joint team of customers/users, executives, & dev
  - An intensive off-site meeting, to work out
  - A system’ s details (biz problem, not technical details)
    - Identify the problem
    - Propose elements of the solution,
    - Negotiate different approaches
    - Specify a preliminary set of solution requirements

JAD Process: Two Main Phases

- JAD-Planning
  - Map out broad capabilities of the software system
  - More business concerns than detailed tech concerns
  - Outcome: system’ s goal, preliminary effort & schedule estimates, and **a decision about whether to continue**
  - Set up the JAD-Design phase
- JAD-Design
  - Elicit more detailed requirements
  - Create user-level (not functional) design of the software
  - Use **prototyping**
  - Outcome: **detailed UI design, DB schema (if any), refined budget and schedule estimates**, for **approval**

JAD Activities



JAD Planning: Customization

- Organize the JAD team
  - Who will need to attend the meeting?
  - Usually not the same people for design
- Orient participants to the JAD Process
- Tailor JAD tasks and outputs to the specific project
- Prepare the materials for the JAD-planning sessions

JAD Planning: Session

- Timeline: 1~10 days, all people attend full-time
- Facilities: off-site (avoid distraction!!), visual-aid
- Roles
  - Leader: key person, trained/experienced, JAD-enthusiastic
  - Executive sponsor (financial), decide go/no-go
  - Key end-user representatives (good communicator)
  - Developer (assist end-user by providing info, not judge)
  - Scribe (from dev team, record things, & clarify problems)
- Note: involve key people, limit team size, no observers

JAD Planning-Session Activities

- Conduct orientation (purpose/timetable/agenda)
- Define high-level requirements (business need, benefits, rough functions and priorities, strategies…)
- Limit system scope
- Identify & estimate follow-on JAD-design phases
- Identify JAD-design participants
- Schedule JAD-design phases
- Document issues and considerations
- Conclude the session

JAD Planning: Wrap-up

- JAD Document of session activities
- List of system objectives (strategic considerations)
- Details of system functions, business needs, benefits/returns, & prioritization…
- Limitations on system scope (sth not included)
- List of interfaces to other systems
- List of issues unresolved during JAD session
- Plan for what happens next, follow-on…

JAD Design: Customization

- Similar as JAD Planning
  - Preparation
    - Room, visual aids and forms,
  - Team organization
    - Executives not required but may drop in sometimes
    - Developers are key to develop quick prototypes
    - A group of key end-users can devote time on details
    - Project manager can attend but don’ t lead
  - Orientation

JAD Design: Session Activities

- Conduct orientation
- Review & refine JAD-planning requirements & scope
- Develop a workflow diagram (scenario)
- Develop a workflow description (textual)
- Design the screens and reports (prototypes)
- Specify the data processing requirements (data volume, rates, security…)
- Define the interface requirements
- Identify the system data groups and functions
- Document issues and considerations

JAD Design: Wrap-up

- Complete JAD design document
- Complete the prototype
- Have all participants review them
- **Present results to executives for approval**
  - Summary of design session
  - JAD design doc
  - Preliminary target implementation dates
  - Project’ s current status

JAD Advantages

- One of the most powerful req spec practices
- Commit top executives to the software planning process
  - **Shorten the product-approval cycle**
- Shorten the requirements-specification phase
- Eliminate features of questionable value
- Improve communication, get req right after 1st time
- Get quick feedback & get the UI right after 1st time
- Reduce organizational infighting (political conflicts)

Keys to Success in Using JAD

- Experienced JAD leader (e.g., PMs, 要准备好!!!鼓励大家集思广益)
- Be sure an executive sponsor committed
- Full time participations of all sessions by key people
- Off-site intensive workshops (avoid distractions)
- Prepare participants thoroughly (process & objectives)
- Set realistic end-user expectations for work to be done after JAD design
- Follow up JAD design with an incremental model
- Move quickly from JAD design to functional design, implementation, otherwise, out of date, no support

JAD Efficacy

- 15~35% time off for requirement specification
  - IBM (1985)
- Almost 70% efforts off
  - CNA Insurance Company
- 20~60% time and efforts off
  - Judy August (1991)
- So, 5~15% off the total dev process

Readings

- R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 5 ed, McGraw-Hill, 2001
  - Chapter 11.2.2, pp. 275~279
- S. McConnell, *Rapid Development*, MS Press, 1996
  - Chapter 24
- Joint Application Design—Business Requirements Analysis for Successful Re-engineering
  - *by Bill Jennerich*
  - http://www.bee.net/bluebird/jaddoc.htm

软件需求描述Spec方法：需求清洗及最小化

Requirements Scrubbing and Minimal Specification

Req Scrubbing—Definition

- After JAD
  - Many requirements
- A **Practice for Requirement Specification**:
  - Examine a product specification carefully,
  - Find out and REMOVE entirely those requirements
    - Unnecessary
    - Overly complex (expensive, risky, …)
  - 最小化可行产品 minimum viable product (MVP)

Req Scrubbing—How to Do

- After creating a product specification
- Go over the specification with a fine-tooth comb and with the following aims:
  - Eliminate all requirements that are not absolutely necessary
  - Simplify all requirements that are more complicated than necessary
  - Substitute cheaper options for all requirements that have cheaper options

Req Scrubbing —Benefits & Risks

- A most powerful way to shorten a SW schedule
  - Because you remove all effort associated with that feature
    - Specification , design, testing, documentation, …
- Reduce the size and complexity of the product
- So, reduce the risk level: less risky than minimal specification (which is shown in the next slide)
- Easy to practice for the first time
- Risk: Elimination of requirements later reinstated
  - Cost more than it would have if you had retained it

Minimal Specification

- Consists of the **minimal amount of information** needed to meaningfully specify the product:
  - A short paper of spec (<10 pages)
  - User manual (or on-line help) as specification
  - UI prototypes
  - Paper storyboards (sometimes low-tech works best)
  - Vision statement or product theme (what to do & not)
- Benefits:
  - Opportunistic efficiency (free to design and implement, less constrained)
  - Less wasted effort and shorter requirements phase

Minimal Spec: Major Risks

- Omission of key requirements
  - more cost to recover
- Unclear or impossible goals
  - dev speed or max usability?
- Gold-plating
  - an individual may want perfection
- Use it for the wrong reason
  - for overall time, not for quick spec itself
- Lack of support for parallel activities
  - User manuals and test cases should wait until “visual freeze”

Minimal Spec: Keys to Success

- Use it only when requirements are flexible
- Keep the spec to a minimum
  - not too many details!!!
- Capture the important requirements
  - users really care about!!!
- Involve key users
  - who know business and software needs
- Focus on graphically oriented documentation (better than text)

快速原型方法

User-Interface Prototyping

What is Prototyping

- Prototype is an approximation of a system
- Prototyping:
  - Using a prototype system to simulate the system behaviors
  - Pretending it is the real system
  - An approach to developing complex software
    - Evolutionary Prototyping
    - User-Interface Prototyping
    - Rapid Prototyping

What is Rapid Prototyping?

recreating a

particular aspect

of a software experience

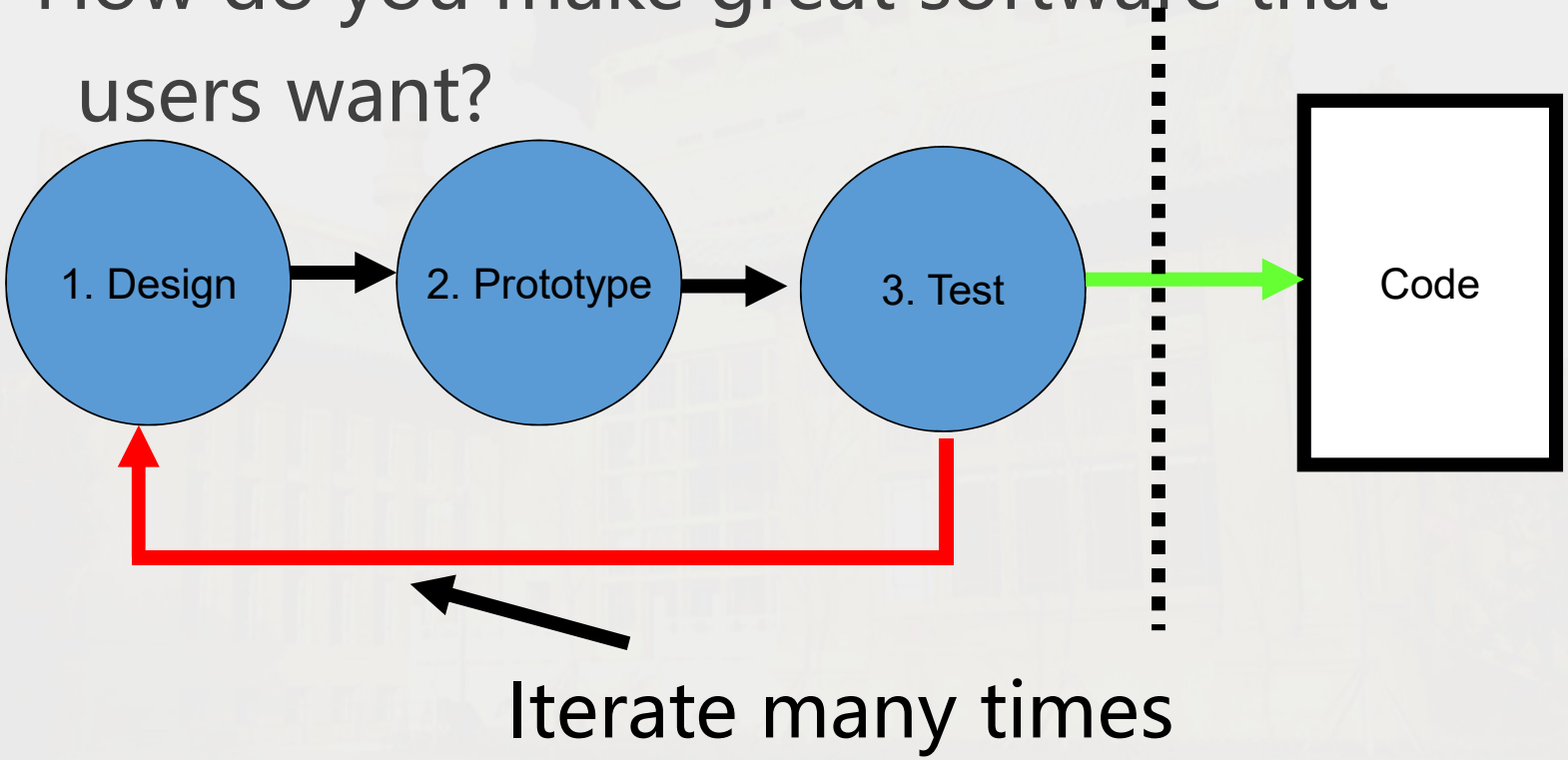
as quickly as possible

Why Rapid Prototyping

- Requirements are known gradually
  - Rapid Prototyping is a way of progressively develop an app hand in hand with an understanding of its requirements
  - SW lifecycle based on prototyping is rather different from the typical waterfall model, which requires a complete requirements analysis and spec before dev
  - Prototyping is based on a more flexible and iterative development model, which affects not only technical issues but also organizational and managerial issues.
- Rapid Applications Development (RAD)

The Iterative Design Cycle

How do you make great software that users want?



Start Prototyping with Brainstorming and Scenarios

- You cannot design in a vacuum
- Anything you make is eventually for some user
- Users have:
  - Things that they have to do
  - Things that they would like to do
  - Real-world constraints: time, money, patience
  - Lots of complaints, few solutions
- The secret:
  - Get users involved early
  - Keep them involved throughout the process

Scenario Development

- A “scenario” is a little story
  - about a user,
  - that could happen,
  - at certain time and place
  - where the user is doing something with the software,
  - and how his/her wants and needs are satisfied by some technology in the software

Making Good Scenarios

- Specify everything:
  - Who, What, When, Why, How Long, How Much
- Be as specific as possible
  - The user is “Andy, a new student to GDUT”
  - Or, “Ben, forearms-cut, wants to play a CD w/ PC”
  - “Tom, a tech writer, wants to draw a flowchart”
  - Not “anyone who likes music”
- *If you can’t be specific, it is usually means your scenario is flawed*

User’s Desires and Constraints

- First, think in terms of user’s desires
  - Does Andy want to know how to use Library, or is his boss asking him borrow books?
  - Focusing on desires ensures you create something that the user actually wants
- Next, think about user’s constraints
  - Is Andy’s office located in the same or different building with the Library?
  - Is Anna’s PC equipped with an camera and mic?
  - Is Tom’s PC equipped with a pen/tablet?

Propose Solutions

- Now think about how they would actually like to satisfy their needs
  - Come up with several possibilities
  - It is OK to be unrealistic

- At this stage, identify potential problems and solutions

Make Good Scenario: Full Story

- Finally, come up with a story that takes everything into account:
  - "Anna, a 40 yr-old woman whose forearms are cut, has just brought some DVD movies from store. Now that she has some spare minutes before supper, she wants to quickly browse them on her PC. She sits down in front of the PC which is equipped with a camera and a microphone. She says "turn on the PC" and the PC is booting. After that she says "Open CDROM", then it is opened. She puts one CD in it. The PC then closes the CDROM and begins to display the list of scenes inside the DVD. Anna stares at one of the scene for 5 seconds and the PC starts to play that segment …"

What is a scenario good for?

- Find the user in the real world
  - "Reality check" scenario aspects
  - Feedback & user testing of prototypes
- Determine technical requirements/limitations
  - Do we need to recognize many voice commands, or just a limited set of commands, like "open", …
  - Is the user willing to help, or must it be automatic?

Brainstorming for Scenarios

- Goal: **rapidly** generate a large number of **ideas**
- Focused on a particular topic, but
  - Fast
  - Spontaneous
  - Unedited
  - Uncensored
  - Uncritical
- Generate ideas first, *then* do the editing

How to Brainstorm

- Find a good location
  - You need lots of space to write
- Allocate enough time
  - It takes a while to get going
  - But, brainstorming can be very exhausting
  - Several short meetings are better than one long one

How to brainstorm (cont.)

- Invite the right people
  - A diverse mix works better than a homogenous crowd (artist, admins, researchers, devs, boss)
  - Everyone should be equal – no authority
- Select one member to be mediator
  - writes stuff on the board
  - Encourages the flow of ideas
  - Prevents wandering from the topic
  - Knows when to move on

How to Brainstorm (cont.)

- Start with a theme, concept, idea, or question
  - "What kinds of disable people?"
- Then run with it:

Blind	One arm
Arms	Both arms
Hands	One hand
Legs	Both hands
…	…

DO NOT EDIT OR CRITICIZE !!!

How to Brainstorm (cont.)

- When the ideas slow down, stop the process
  - Trick is to keep things happening fast
- After you stop, spend a little time organizing or editing
- Then, move on to the next topic
- End result:
  - Fairly complete map of some problem area
  - Your own interests identified

Brainstorming

- Brainstorming is just a tool
- Use it whenever it feels appropriate
  - Coming up with scenarios
  - When you are stuck in the middle of a scenario
  - You have a new idea
  - Brainstorming works well as a regular event
  - You could have “Friday morning brainstorming”
  - People can sign up & bring a list of topics
  - Don’t worry! Over time, this gets easier

Rapid Prototyping

- Create a scenario or part of a scenario
  - **for the user**

- DOES NOT HAVE TO BE “REAL”

Design a Prototype

- Figure out what is essential to the prototype
  - Everything else is a *waste of effort*
- Need to be extremely specific and minimal
  - If your prototype takes two weeks to code, it isn't a prototype
  - This is where scenarios can really help
- The hardest part:
  - *Overcoming your desire to "do it right"*
  - The product will do it right, not the prototype

Prototyping Techniques

- "Pencil Test"
  - Fastest way to code is to not code at all
  - Walk someone through the steps on paper
  - Can be done in programs like Director or Photoshop
- "Wizard of Oz"
  - Replace a complex technology with a person
    - Speech recognition
    - face recognition / eye tracking / attention detection
- "Pre-baking"
  - Good for annotations or complex unknown equations
    - Pre-computing every possible search result by hand

Prototyping Tools

- Simple Class libraries
  - C++ & MFC, Java
- Rapid Application Development Platforms
  - C#, Visual Basic
- Authoring Tools
  - Director, Flash, FrontPage
- Paint Programs
- Cardboard
- *Whatever does the job*

Points to Remember

- Focus on what matters, ignore what doesn’ t
  - No one cares if your prototype needs 256 MB RAM, and a Pentium 4 CPU
- Code fast, not well
  - Memory leaks, lost data, crashes, etc. are fine
  - It only has to last for the specific scenario
- Prototype code will NOT be the basis of the product
  - Just like a clay car prototype will not be the basis for the actual car

User Testing

- GOAL: Observe the user moving through your scenario via your prototype
- Step 1: Find out who the subject is
  - Lots of computer-based survey tools exist
  - Age & Sex are important
  - Computer experience
  - Experience with similar technology and products
  - Existing expectations / preconceived notions

User Testing (cont.)

- Give them appropriate guidance
  - Let them know what they need
  - Don’ t lead them by telling them too much
  - Warn them of things to ignore
- Record everything the subject says and does
  - Video & Audio
  - Computer keystrokes and screen
  - Encourage subjects to “think out loud” during the process
  - Be sure to watch them yourselves

User Testing (cont.)

- Follow up after the test
  - Typically another questionnaire
  - What was good and bad
  - Likes and dislikes
  - How it compares to other ways
  - What the user wanted
  - Would they buy this product

Iteration

- Incorporate the results of the user test into the next prototype
- Sometimes, you find flaws in your scenario
- Often, more brainstorming is required
- A product might have dozens of prototypes
  - But ten prototypes should be cheaper than even one re-write of actual code
  - Prototyping does NOT require key developers

Making it Real

- The lessons learned from the prototype should influence the final code
  - Identify good solid design (code)
  - Start coding on what is solid
- Often, the task is made simpler
  - “We don’t need to recognize all voices, just three or four words in a sentence at a time”
- Other times, they change your direction entirely

Wrap Up

- Scenarios are the key to usable results
- Iterative prototyping is what makes scenarios into working products
- Rapid prototyping saves dev work and increases dev speed (for RAD)
- But remember, these things take practice
  - New techniques
  - New way of thinking

Next Steps

- Start brainstorming on existing projects
  - What could someone do if speech recognition worked perfectly?
- Form a prototyping team
  - Mix designers with developers
  - Makes a good shared resource


THANKS
感谢您的关注！个人微信：csliuwy
http://wislab.cn/blockchain