BTC-210

AGILE USE CASE WORKSHOP FOR BUSINESS ANALYSTS

Course outline

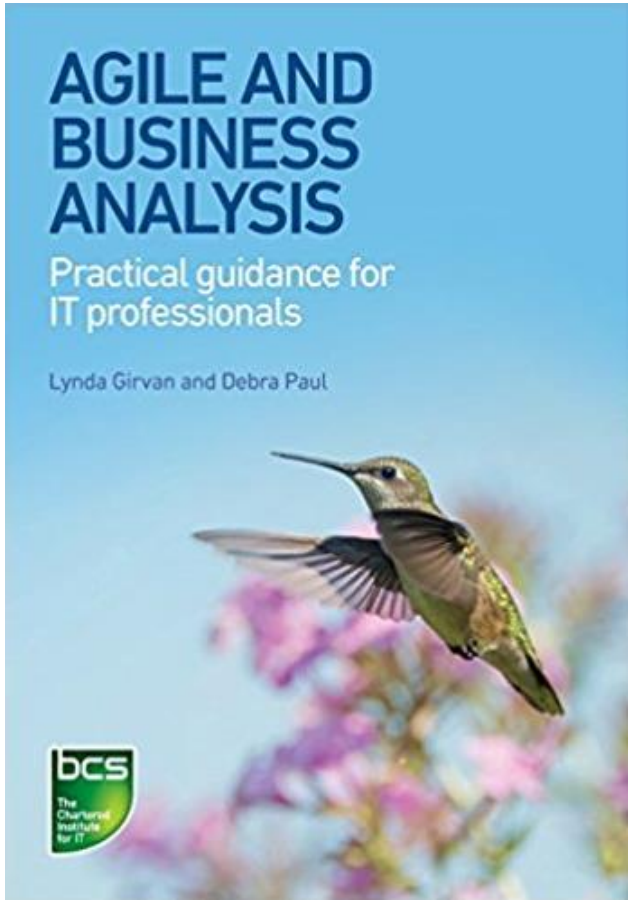
- Two days
- Exercise packet

Related courses

- Business Analysis
- Software Testing

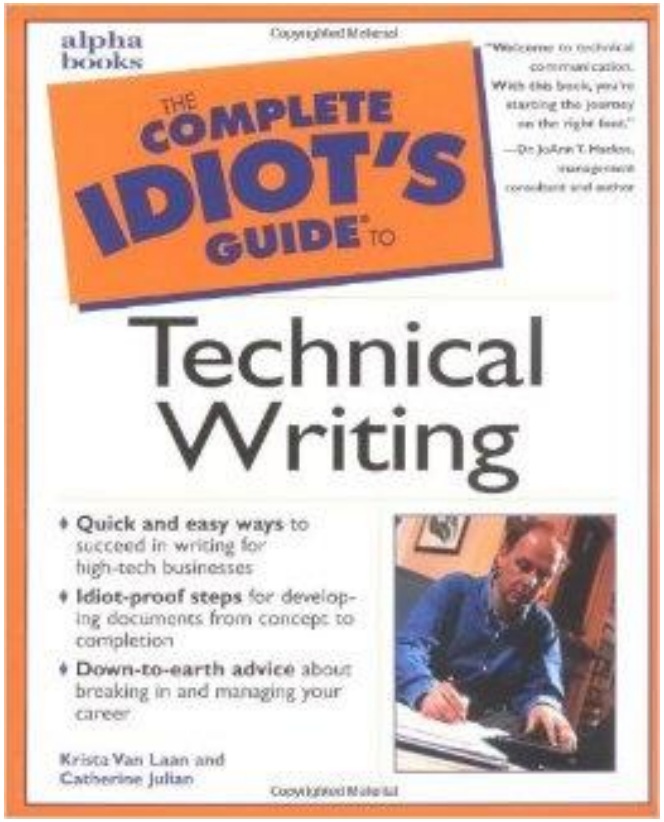
Books, recommended

- **Agile and Business Analysis: Practical guidance for IT professionals**, Lynda Girvan and Debra Paul, BCS Learning and Development Ltd., 2017



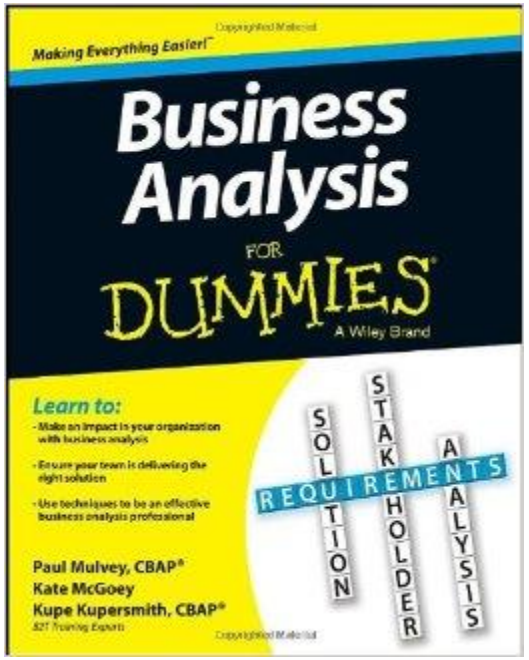
Books, recommended

- **Complete Idiot's Guide to Technical Writing**, Krista Van Laan and Catherine Julian, Alpha Books, 2001



Books, recommended

- **Business Analysis For Dummies** by Kupe Kupersmith, Paul Mulvey, Kate McGoey. July 2013 – recommended
  - $13 used Amazon



Resources

- **https://github.com/doughoff/BTC-210**
- **Modern Analyst**
  - http://modernanalyst.com/
- **Agile Modeling web site**
  - http://agilemodeling.com/
  - Scott Ambler

Agile


LOOK THIS NEW AGILE THING: TO DEAL WITH UNPREDICTABLE EVENTS AND THINGS WE CANNOT CONTROL IN OUR PROJECTS
Dilbert characters Scott Adams Inc.
WE CAN PRIORITIZE, REDUCE THE SCOPE, CHANGE REQUIREMENTS AT ANY TIME AND INCREASE THE CHANCES OF SUCCESS OF THE PROJECT
Punch your own at http://dilbert.com
LOOK, THIS IS YOUR NEW PROJECT, WITH FIXED DEADLINE, FIXED SCOPE AND FIXED QUALITY: YOU CAN BE "AGILE" INSIDE THIS TRIANGLE !!!

Agile values

- http://www.agilemanifesto.org/
- We value
  - **individuals and interactions** over processes and tools
  - **customer collaboration** over contract negotiation
  - **working software** over comprehensive documentation
  - **responding to change** over following a plan
- Principles not a process
- **Quality,** simplicity (not simple)
- 12 agile principles

Agile values for business

- The Agile Manifesto written for business improvement
  - Flexibility of approach – over methods and processes
  - Holistic solutions – over working software
  - Relevant artifacts – over comprehensive documentation
  - Team collaboration – over directive governance

Agile key elements

- Common to all Agile methods
  - A list of work to be done
  - High levels of customer involvement
  - Transparency and sharing progress
  - Regular reviews of progress
  - A whole team mindset
  - Iterative development

Agile values for BAs

- Derived from the 12 agile principles
  - Collaborative working
  - Self-organizing teams
  - Continuous improvement
  - Iterative development and incremental delivery
  - Planning for and building in change
  - Doing the right thing and the thing right

Agile requirements process

- In three steps
  - **Elicit** – acquire and document needs, verify
  - **Specify** – write statements, create models, validate, verify
  - **Structure** – improve, simplify, realign, verify
- Testing types
  - Validation – checklists
  - Verification – get feedback
- Iterate as necessary

Agile requirements process

- Business context (strategic)
  - Visioning – applying vision/mission
  - Planning – estimating asset availability and capital
- System context (systems analysis)
  - Data - dictionary
  - Process – user stories, use cases
- System delivery/iteration context (project analysis)
  - Breakdown – rules, scenarios, UX, prototypes
  - Prioritizing
  - Non-functional brainstorming
- Testing

Agile requirements practices

- **Backlog**: a prioritized list of requirements or work items that is frequently updated
- **Definition of done/definition of ready**: setting acceptance criteria for a requirement
- **Personas**: a way of identifying and describing users of the system
- **User stories**: a way of capturing requirements
- **Story mapping**
- **Story splitting**: breaking down stories that are too big
- **3Cs**: a way of structuring user stories: Card, Conversation, Confirmations

Project Management - Scrum

Agile / Scrum

- Agile (principles), Scrum (implementation of Agile)
- Three pillars of Scrum
  - **Transparency:** those responsible for the outcome must have visibility of all the aspects of the process that can affect the outcome.
  - **Inspection:** the work in progress should be inspected in order that it can be improved.
  - **Adaptation:** when inspection uncovers issues that could lead to the goals not being met, changes must be made to prevent failure. Adjustments should be done as soon as possible.

Scrum common tasks

- Create user stories and put into product backlog
- Prioritize on a wall
- Roll up stories into features into themes
- Work out details continuously with stakeholders
- Re-prioritize as necessary
- Complete requirement just before development
- Use change control

Scrum methods

- deliver software in time-boxed iterations
- focus on highest business-value software features in each iteration
- interact directly with business users to confirm ongoing software usability, relevance and business value throughout the process.

Scrum roles

- Product Owner
  - represents the needs of the business, documents and prioritizes solution requirements for backlogs
- Scrum Team
  - a cross-disciplinary team charged with undertaking the agreed work in each sprint
- Scrum Master
  - facilitates the team's work, removing project impediments and ensuring that appropriate Scrum practices are being followed by the team.

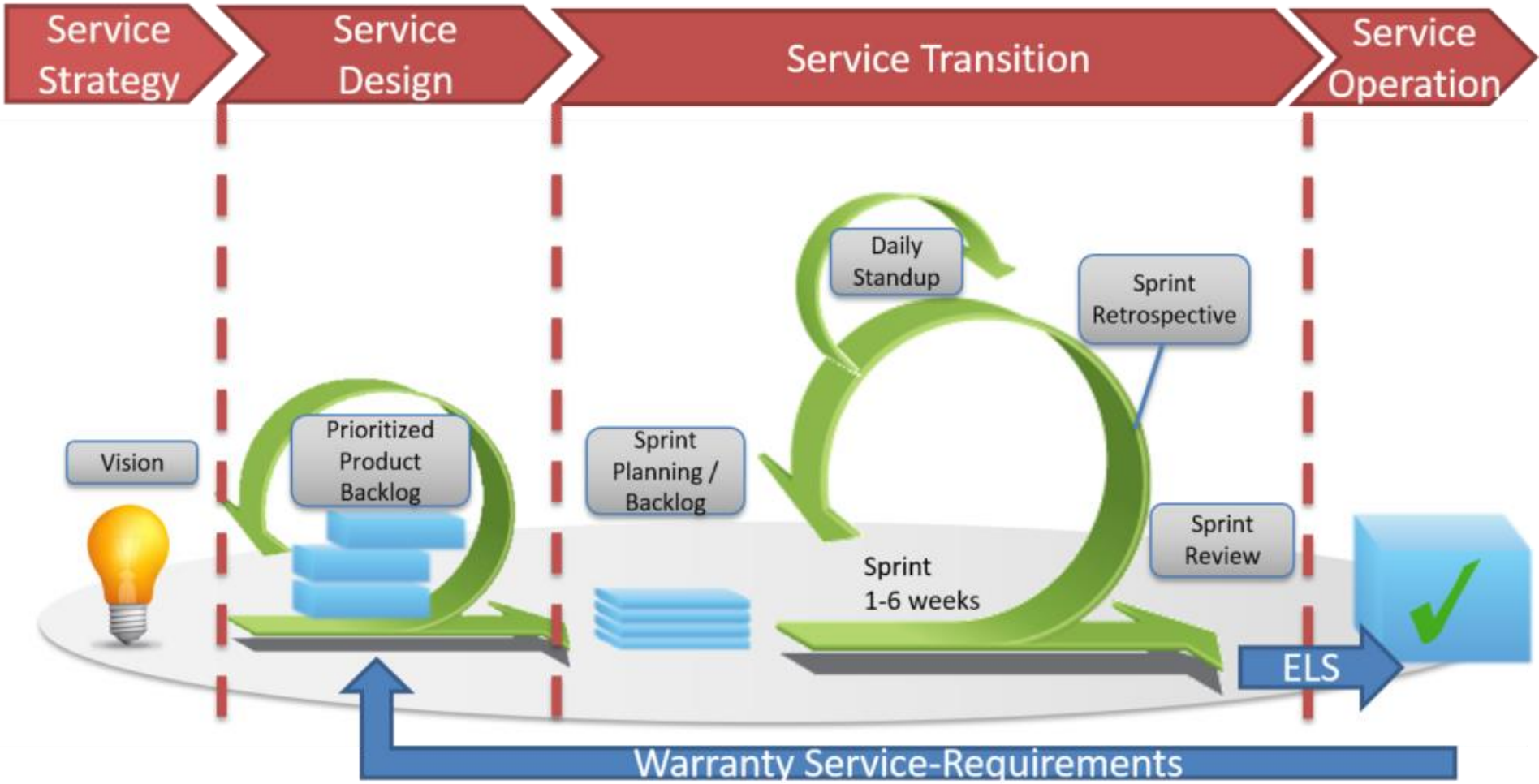
Scrum activities

- Sprint Planning Meeting
  - at the beginning of each sprint, everyone reviews the highest-priority items and agrees on the subset of priority items for the next sprint
- Daily Scrum stand-up meeting
  - encourages teams to hold short update sessions each morning to quickly review completed and planned work and address any hurdles
- Sprint Review & Retrospective
  - at the end of each sprint, demonstrate work completed in that sprint and a retrospective review of the work undertaken to enable continuous improvement for subsequent iterations.

Scrum artifacts

- *Executive dashboard*
  - summary monitoring report of work within (and across) Agile teams and the organization
- *Product backlog*
  - a monitoring report of work against the agreed business requirements for stakeholders and project teams, Kanban
- *Sprint backlog*
  - a monitoring report of actual day-to-day work
- Increment
  - The items completed during this and previous sprints. Must be usable.

Scrum – ITIL view



User story estimation

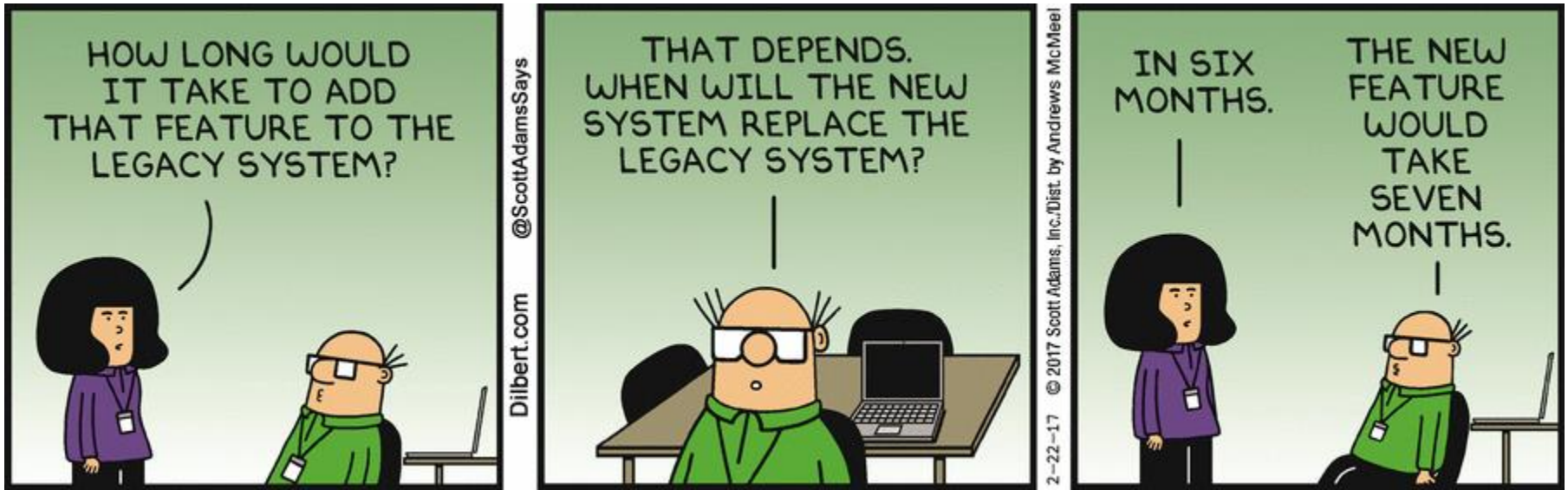
- Story points - a subjective number representing a combination of things:
  - Volume – How much is there?
  - Knowledge – What's known?
  - Uncertainty – What's not known?
  - Complexity – How hard is it?
- Ideal days
  - Days without interruptions

User story estimation - poker

- All team members can estimate but the Product Owner does not estimate. The Scrum Master does not estimate unless they are doing development
- Each team member is given a deck of cards with 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞, and ?
- For each backlog item to be estimated, the Product Owner reads the description of the story
- Questions are asked and answered
- Each estimator privately selects an estimating card
- All cards are simultaneously publicly turned over
- High and low estimators explain their estimates
- After discussion, each estimator re-estimates by selecting a card
- Repeat the process for consensus if the estimates don't converge.

Estimation - other

- T-shirt sizing (XS, S, M, L, XL, XXL, XXXL)
  - 1 point for extra small features, 2 points for small features, 3 points for medium features, 4 for large, and 5 for extra large…



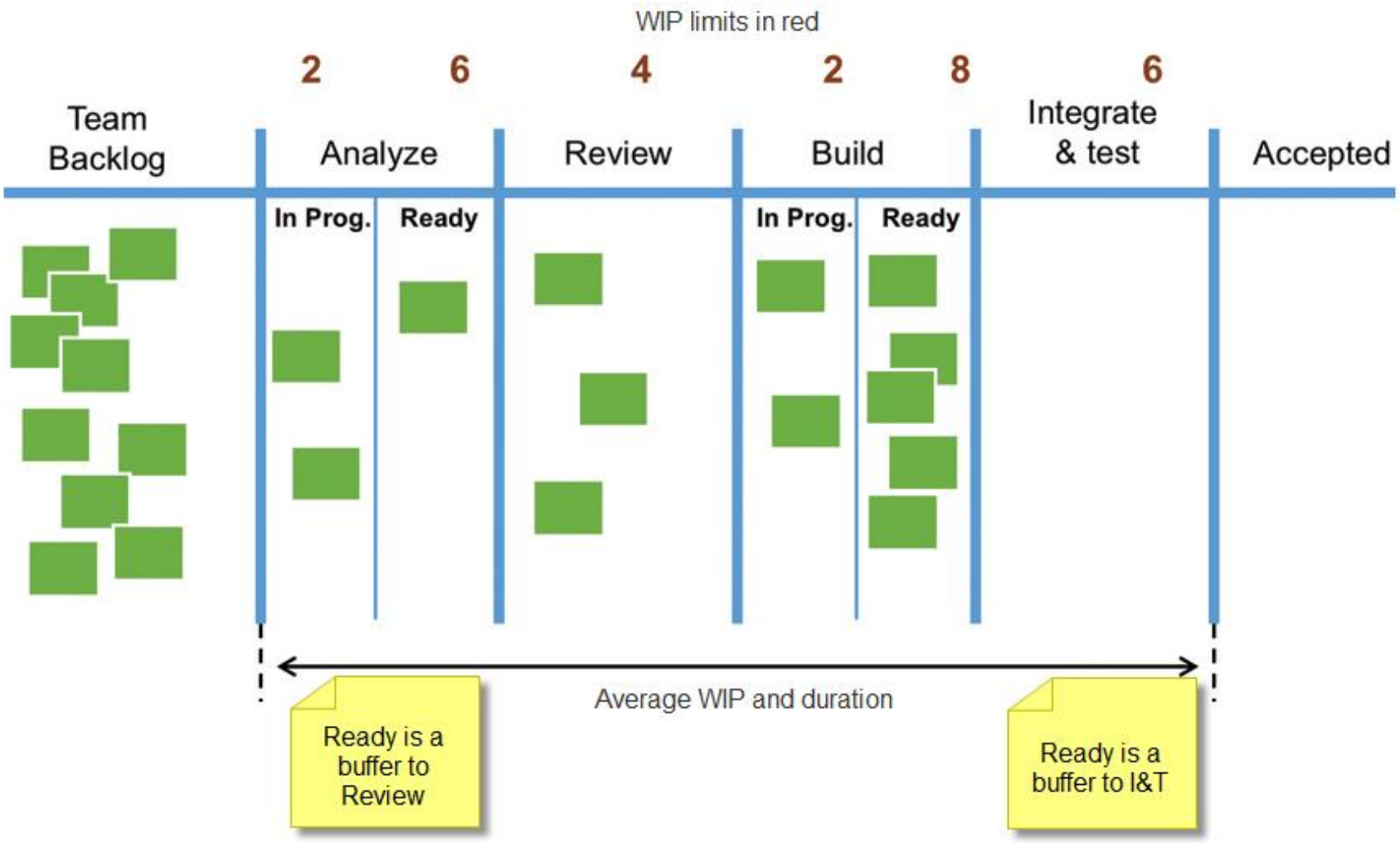
User story velocity

- SAFe estimates team velocity:
  - For each developer-tester,
    - add eight points (adjust for part-timers)
    - Subtract one point for each vacation day or holiday
  - Find a small story that would take about a half-day to code and a half-day to test and validate. Call it a one.
  - Estimate every other story relative to that one.

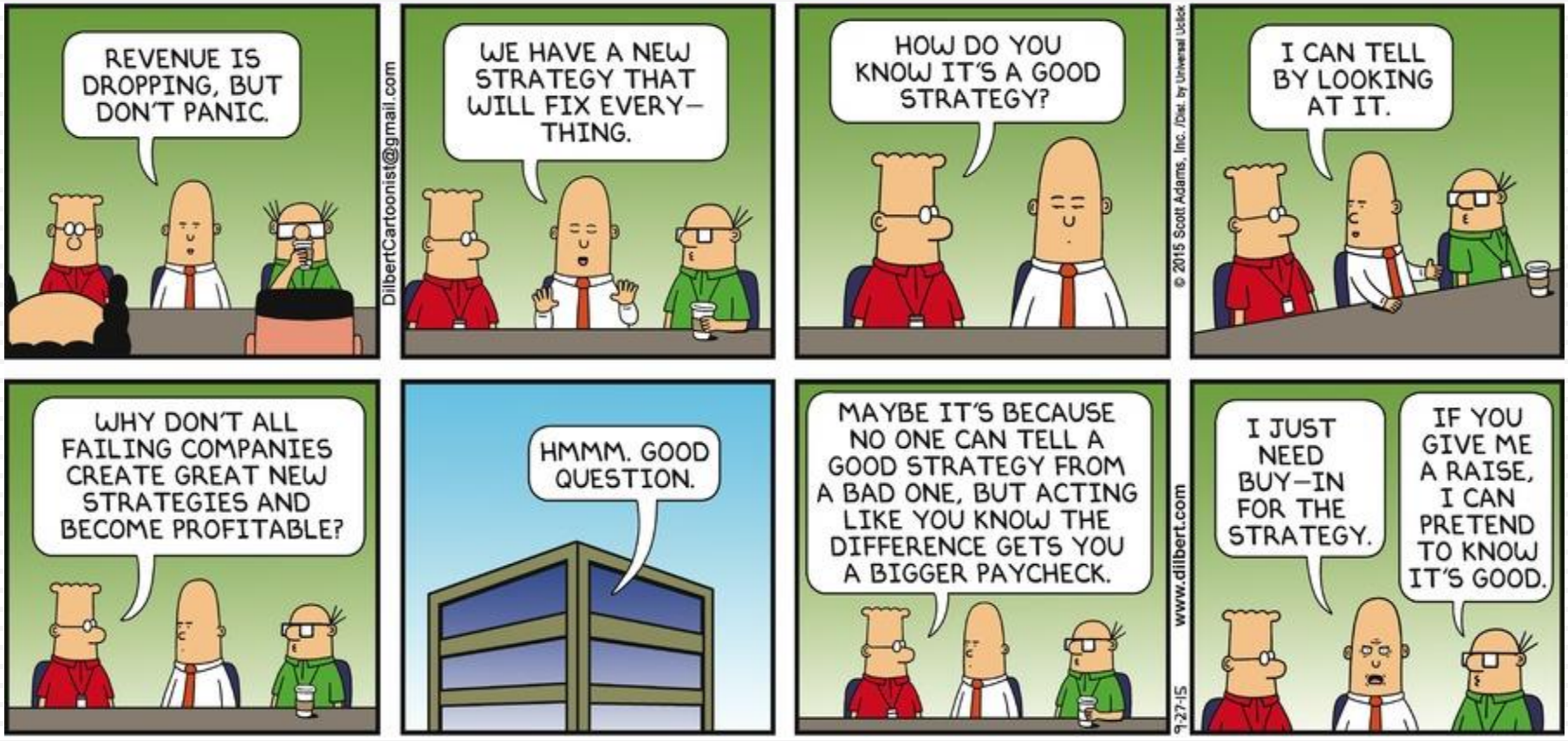
Kanban boards

- team workload management and change management methods
- team to portfolio levels
- limited work in progress (WIP)
- visualize work flow in
  - planned, current, and completed work status
  - availability for work
  - blocks to work

Kanban board - team



Visioning and planning



Organizational strategy

- Baseline
  - Key areas
  - SWOT
  - Constraints
- Governance
  - Beliefs, vision, principles, policies, rules
- Functional
  - Mission, goals, processes, objectives, milestones, tasks
- Non-functional
  - CSFs, KPIs, metrics, measurements

Project strategy

- Mission, goals
- Problem statement, other problems
- Constraints
- Stakeholders
  - People
  - Systems

Business case

- Necessary
  - Market for service, resources available, use of resources for service, value of service to company, tie to vision, constraints.
- Optional
  - feasibility study
  - top-level architecture
  - business requirements (goals)
  - project strategy plan
  - operations concept document

Business case

- High-level “requirement”
  - Sell packaged products without a cashier.
  - Inventory control for pharmaceuticals
- Constraints
  - Must keep digital logs
  - Must use credit
  - Must use employee ID card

Agile requirements planning

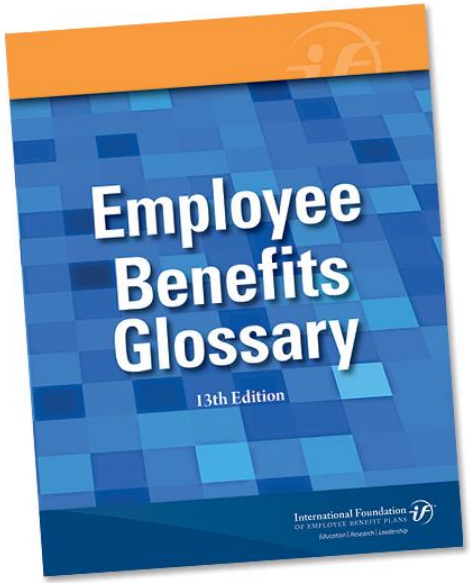
- Prepare yourself (skills)
- Identify stakeholders
- Understand problem domain
  - GO-SEE – Toyota Way 2001, don't just look at numbers, not micro-management
- Design the approach
  - Scrum?
- Schedule sessions

Identify stakeholders and users

- Document them
- Questions to drive out details
  - Who are the users of the system?
  - Who is the customer (buyer) of the system?
  - Who else will be affected by the outputs of the system?
  - Who will evaluate and bless the system at delivery and deployment?
  - Who will maintain the system?
  - Are there other internal / external users with needs?

Glossary

- Dictionary of common terms relevant to project
- Can be enterprise wide but should be extracted for each project
- Business terms
- Assign a responsible analyst



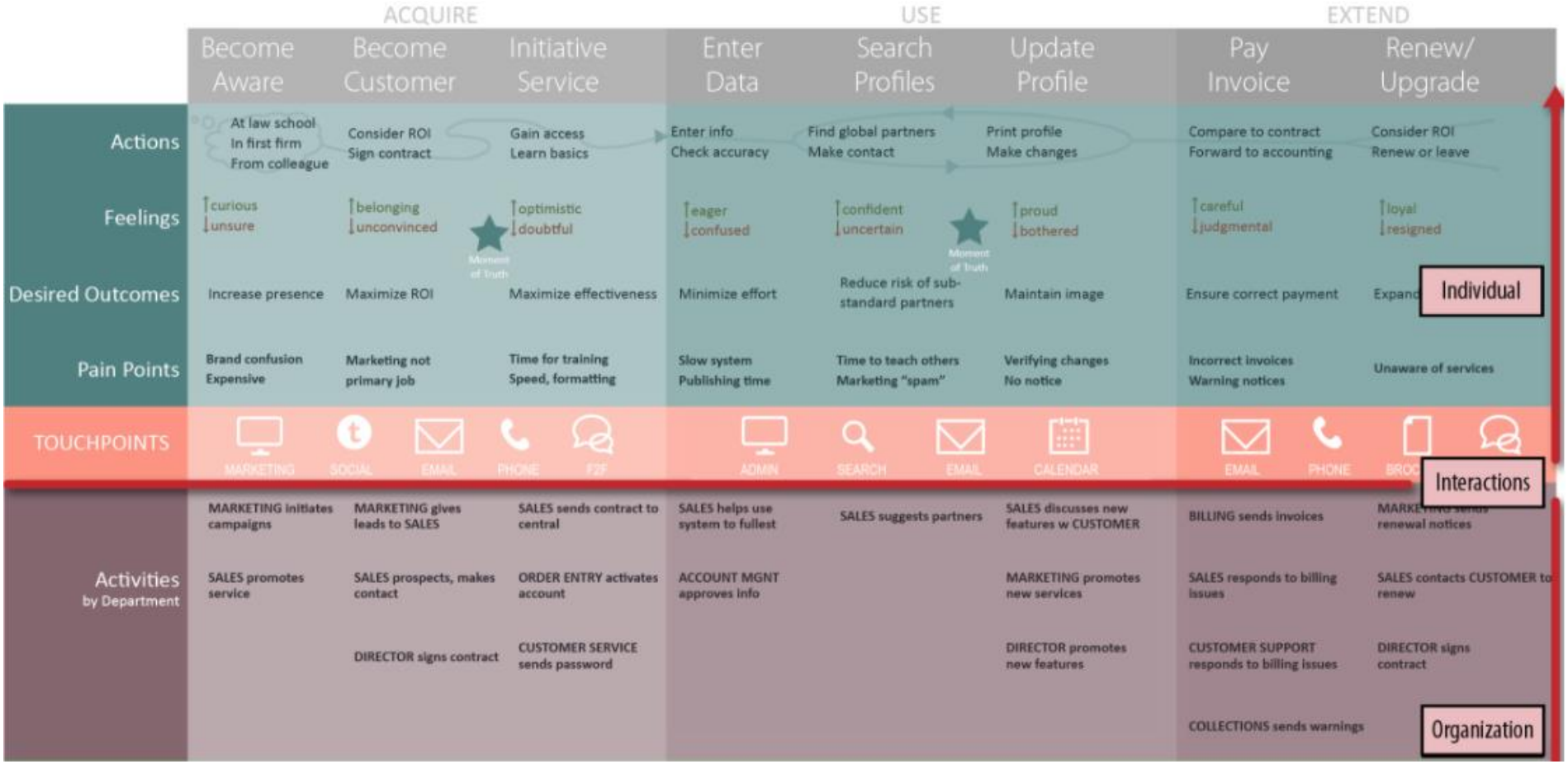
Four aspects of design

- Project
- Context
  - Other systems, “side effects”, infrastructure
- Management systems
  - How to be aware
- Metrics
  - System non-functional requirements

Value-centered design

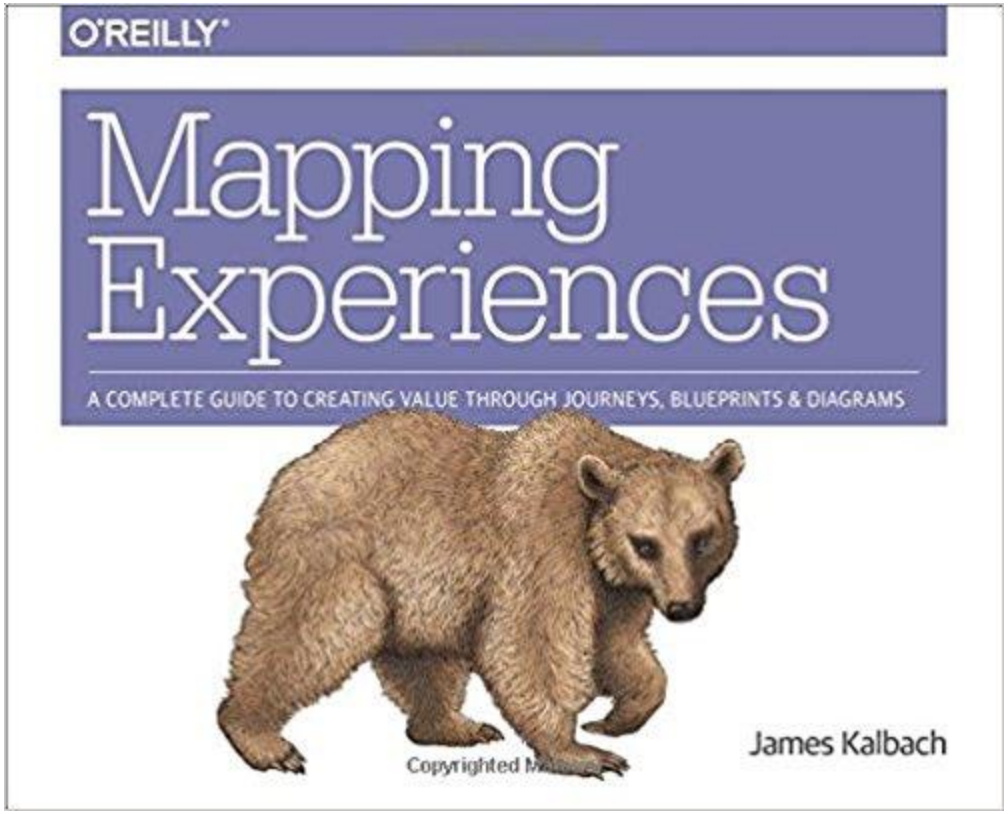
DIAGRAM TYPE	STORY	INTERACTION	INDIVIDUAL	ORGANIZATION
Customer journey map	Chronological	Touchpoints	Actions, thoughts, feelings, pain points, etc.	Roles and departments involved in creating an experience
Experience map	Chronological	Touchpoints	Actions, thoughts, feelings, pain points	Physical and social artifacts in a system; opportunities
Service blueprints	Chronological	Line of interaction	Actions, physical evidence	Backstage actors and processes
Mental model diagrams	Hierarchical	Center line	Tasks, feelings, philosophies	Support—products and services available
Spatial maps	Spatial	Midpoint with arrows	Actions, needs, information flow	Data systems, departments

Customer journey map



Book

- **Mapping Experiences** by James Kalbach
  - O'Reilly Media, Inc., April 2016



Problem statement style - simple

- The sales pitch to sell the project
- The elevator pitch / marketing style:
  - for <customers> who have <reason> our <idea> so that <benefits> unlike <currently / competition>
- Pixar pitch
  - once upon a time… every day… one day… because of that… and … until finally …
- Focusing question
  - How can <we> do <idea> for <customer> so that <benefit>
- Twitter pitch
  - <idea> #<benefit> e.g. convenient music player #1000PocketSongs

Problem statement style - generic

- Use at beginning of requirements gathering
- the **standard format of problem writing** is
  1. “this <problem>…
  2. affects <all these people>…
  3. with <unhappy specific symptoms, actual effects, not causes> for the business...
  4. but our solution would benefit us by<better business results, overall improvements>…”
     - Don’t just negate 3. to get 4.

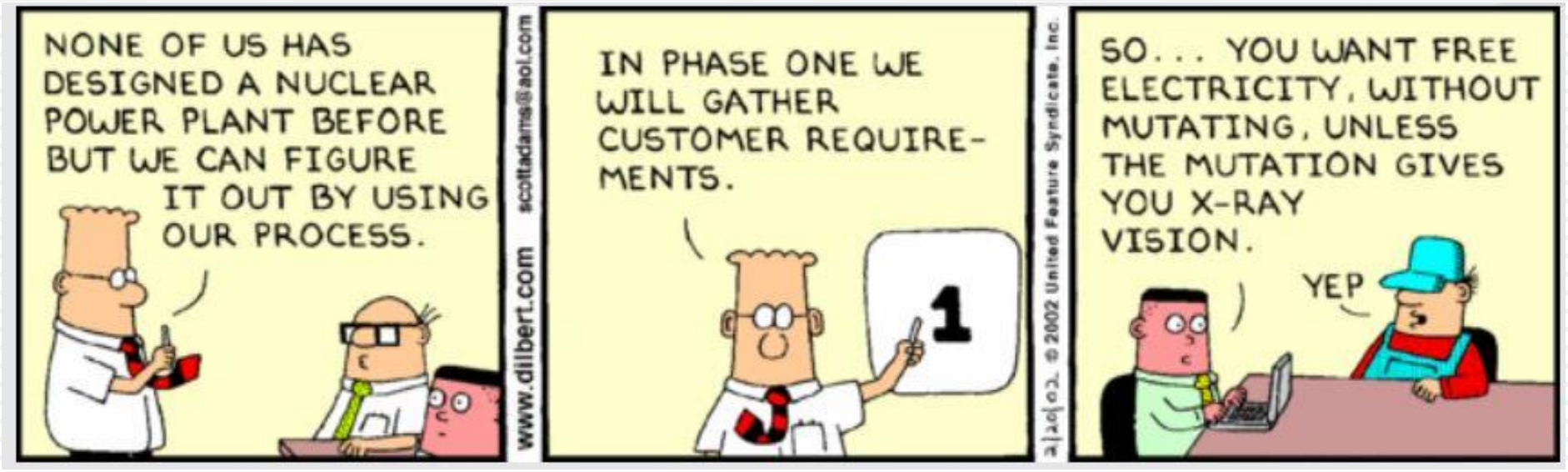
Problem statement style - detailed

- Vision statement for products
  - For <target customers> who <statement of the need or opportunity> the <product name> is a <product category> that <key benefit or compelling reason to buy>.
  - Unlike <primary competitive alternative, current system, or current manual process>, our product <statement of primary product differentiation>
- **Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers** by Geoffrey Moore 1999

Exercise: Problem statement

- Create a problem statement for the vending machine using two techniques
  - 1. the standard format of problem writing
  - 2. Moore's vision statement

System - data


NONE OF US HAS DESIGNED A NUCLEAR POWER PLANT BEFORE BUT WE CAN FIGURE IT OUT BY USING OUR PROCESS.
www.dilbert.com scottadams@aol.com
IN PHASE ONE WE WILL GATHER CUSTOMER REQUIRE-MENTS.
1
©2002 United Feature Syndicate, Inc.
SO... YOU WANT FREE ELECTRICITY, WITHOUT MUTATING, UNLESS THE MUTATION GIVES YOU X-RAY VISION.
YEP

Data elicitation – document review

- Getting copies of files and reports
  - gather data more than process
  - require little time for stakeholder
  - allow analyst to review whenever they like

Data recognition

- Business level language terms
  - known as entities
- Data is required by the system / role for it to
  - know about
  - track for changes
  - remember to use later
  - report on

Affinity diagram

- created in the 1960s by Japanese anthropologist Jiro Kawakita
- Process
  - record ideas/observations and spread on surface
  - place ideas side by side **silently**
  - discuss patterns, shape, motives to move, promote/create a heading idea to group
    - group groups if possible

Exercise

- Brainstorm data entities for the vending machine system
  - Discover the entities in affinity diagramming

Exercise

- Create the data entity diagram for the vending machine system
  - Discover the entities in affinity diagramming
  - Analyze the entities by
    - Placing simple data types under complex data types
    - Associating complex data types to other complex data types
    - Using cardinality on the associations.

Data diagram process

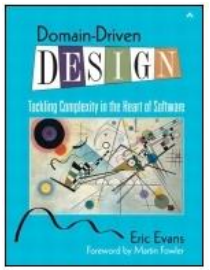
- Entity diagram - A high-level UML class diagram
- Discover all of the nouns in a use case.
- Place simple data under complex data
  - **Simple** (fields): Numbers, dates, text, flags
  - **Complex** (entities): People, places, things, roles, events
- Show a line from complex to complex data based on a report where that first complex datum will
  - Need to report on the other datum
  - Need to know about the other datum
  - Needs to be tracked with this other datum

Class diagram

- Show cardinality on the line
  - The first datum will need many (*) of the other datum
  - The * datum will go back and relate to one or more of the first datum.
- Show role on the line
  - A name is common to describe the role
    - Person ---------- has-a ----------- Car
    - Person ---------- sells ------------ Car
- Show direction if needed
  - Dog ---------- <- owns ------------ Person
  - Dog -------- eats from bowl at ->---------- Person

Resources

- **Domain-Driven Design: Tackling Complexity in the Heart of Software** by Eric Evans. Addison-Wesley Professional, Aug 2003



- **Analysis Patterns** by Martin Fowler
- For programmers
  - Larman, Craig 1998. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design.* Prentice Hall PTR. (get the 2nd version not the 3rd)

System - process

Styles of elicitation

- Analysts traditionally interact with stakeholders through
  - interviews
  - prototyping sessions
  - document review
  - game style sessions – popular with Scrum
  - surveys

Grouping scope by system

- High level scope units allocated to a component (subsystem) help establish
  - Project management resource goals for
    - Hardware
    - Software
    - People
  - Interfaces between components

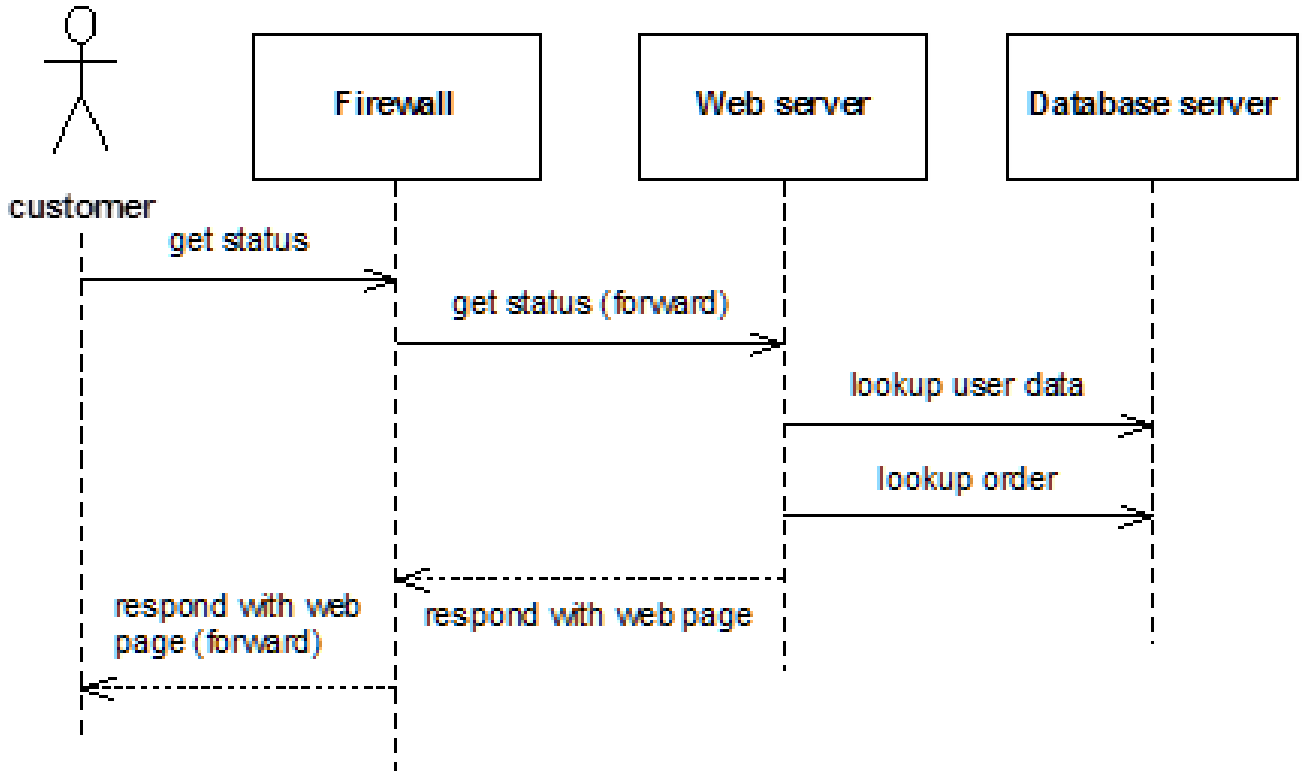
Systems, messages, data

- Systems/components = rooms in house – kitchen, TV room, kid's room
- Messages = come get dinner, come get my dirty plate, come get dessert, come get my dirty bowl.
- Data = food, beverage, china, silverware, napkins

System - context

System-level sequence diagram

- Sequence diagrams show
  - participating systems and roles
  - how often we communicate
  - events in time order



Sequence diagrams - tools

- Text driven – best!
  - * http://sequencediagram.org
  - https://www.websequencediagrams.com/
- Drag and drop objects
  - Visio
  - https://www.gliffy.com
  - https://creately.com/

Exercise - System level sequence diagram

- Vending machine
  - Walk through a scenario of purchasing a product without cash and record the systems and the messages
    - Cell phone app
    - Vending machine
    - Credit card authorization system
    - Back end business system

System process - user stories

User stories

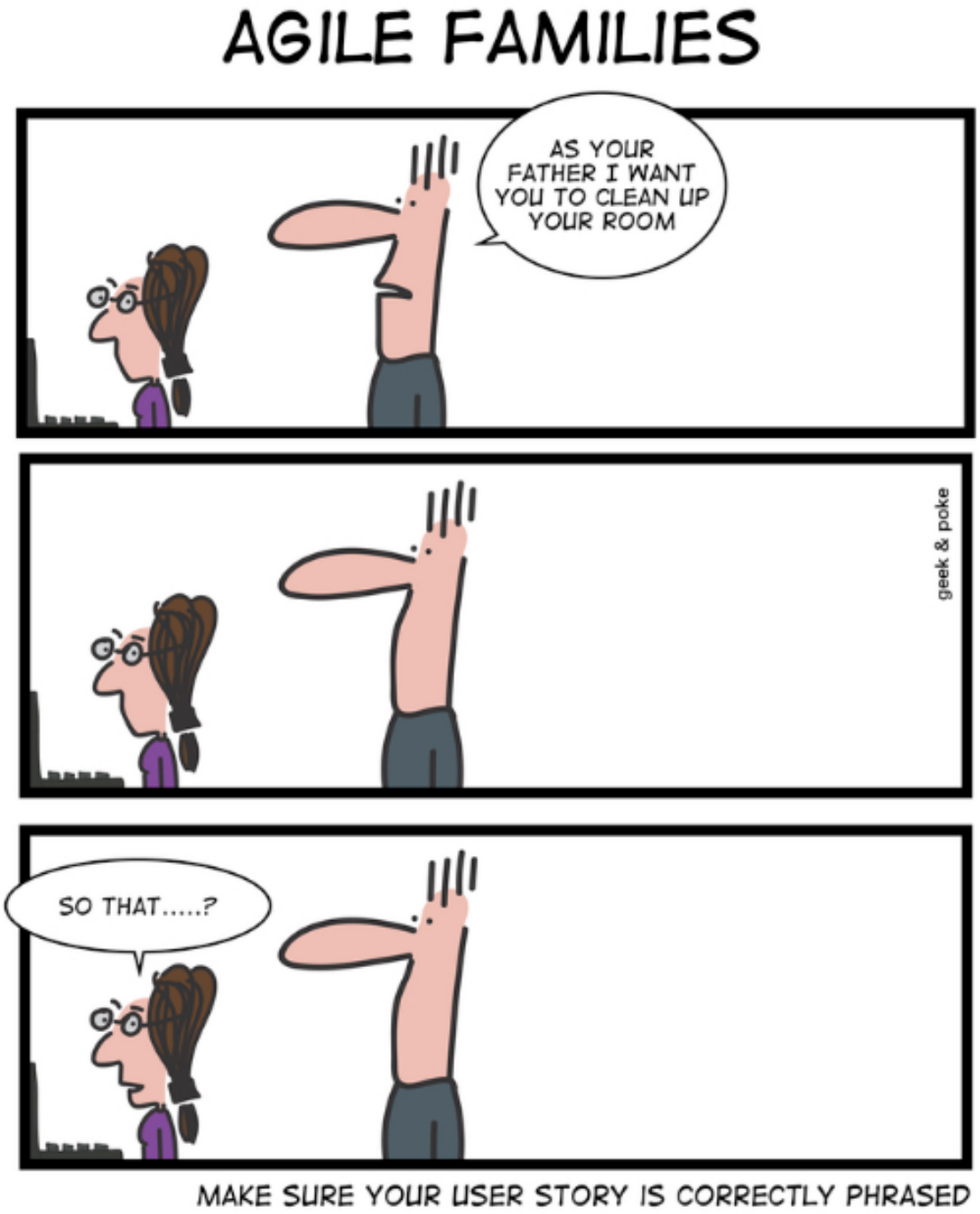
- A requirement elicitation technique
  - not a final product
  - conflicts are OK
  - uses note cards
  - Any granularity is OK
- As a <some user role>, I need the system to <high-level functional requirement> so that I can <perform next task / get value for the business>.
  - detail with use cases, prototypes later
- Kent Beck's idea in XP story telling to simplify requirements.

User stories are for:

- Always
  - Product description
  - Planning items
  - Exploring ideas that generate features
- Users' needs most often
  - Tokens for a conversation – **main intent**
  - Way to defer a conversation
  - Opportunities for value, evaluated by team

User story - value

- User stories can see value as either
  - User centered
  - Business centered



User story – value

- User value
  - Same as basic requirements collection from user
  - Focus of the story is on the user / actor
  - Value is most often expressed for the user
- Business value
  - For use in programming, focus must be on system
  - To prioritize stories, the value must be in context of the business

User story granularity

- User activities (backbone)
  - Group of user stories by role, features
  - Only a heading, title, name of group, in time order
- User tasks – 2nd row (skeleton)
  - Only big enough to be delivered in one iteration
  - Things people do
- Epic
  - Hard to estimate, not a clear goal
  - Sometimes related to a portfolio of projects
  - Jeff Patton hates term
  - A large user story, can be decomposed into many
- Feature

User story terms

- Theme
  - A way of marking stories in a category. Maybe a release, a feature type, or user related stories.

Slicing

- Top row of story map is Minimum Viable Product
- Lower indicates lower priority
  - Build left to right
  - And the top to bottom
- Masking tape to segment off future releases

User stories – 3Cs

- Ron Jeffries, one of the inventors of XP, is credited with describing the "3Cs" of a story
- ***Card***
  - the *statement of intent* on an index card, sticky note, or tool.
- ***Conversation***
  - "promise for a conversation" between the team, Customer/user, the Product Owner, and other stakeholders.
- ***Confirmation***
  - of the *acceptance criteria* provides the precision necessary to ensure that the story is implemented correctly and covers the relevant functional and nonfunctional requirements.
  - Given <role> is <doing use case / activity>
    when <role> <does task>
    then I want <response from system>

User story quality

- Bill Wake's INVEST for a good story
  - **I** – Independent (of all other stories)
  - **N** – Negotiable (a flexible statement of intent, not a contract)
  - **V** – Valuable (providing a valuable vertical slice to the Customer)
  - **E** – Estimable (small and negotiable)
  - **S** – Small (fits within an iteration)
  - **T** – Testable (understood enough to know how to test it)

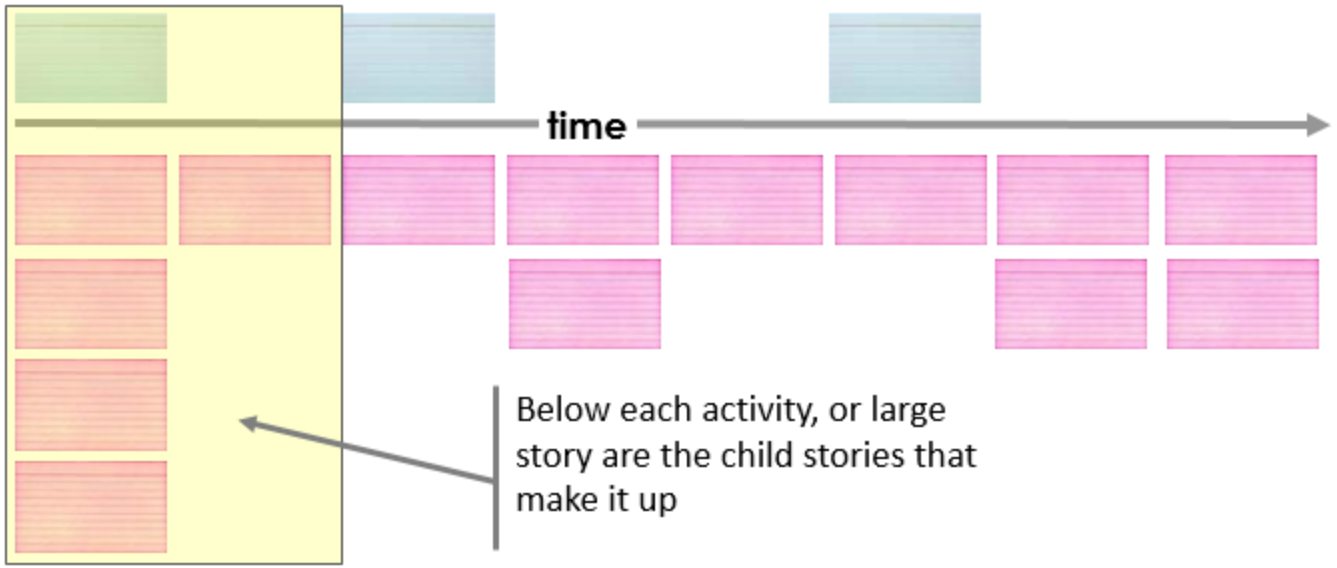
User stories - issues

- Vague and incomplete for implementation
  - especially when everyone isn't sitting in the same room
- Result?
  - feedback at the end of your sprint that rejects or significantly rewrites your user stories
- Solution?
  - analyst writer, sufficient detail

- *Mike Hughes, senior director of innovation solutions at iRise (requirements prototyping)*

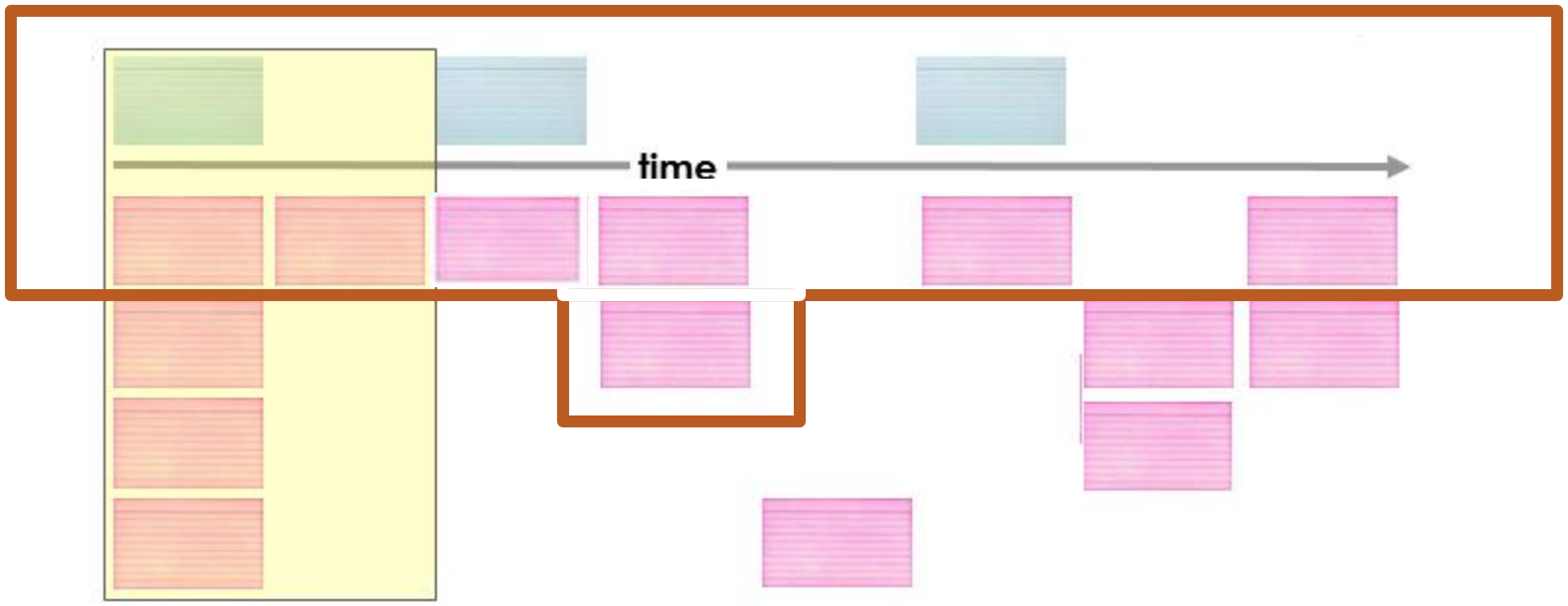
User story map

- Tells a (big) story of the product in time order
- Starts at the top with major activities of groups
- 2nd line breaks down activities into achievable goals
- 3rd, 4th etc. lines are for concurrency



User story map organization

- Change row levels to show necessity (value)
- Backbone : one color
- Skeleton: another color, row 1, always in iteration 1



User story mapping steps

- identify user roles and user stories
- discover logical groupings
- place in a narrative flow (backbone)
- analyze for breaks in the workflow
- decompose the stories
- create a first release
- Mark in progress or complete

Exercise – User story map

- Backbone (functional goals)
- Body (tasks)


Organize Email
Manage Email
Manage Calendar
Manage Contacts
Search Email
File Emails
Compose Email
Read Email
Delete Email
View Calendar
Create Appt
Update Appt
View Appt
Create Contact
Update Contact
Delete Contact
Search by Keyword
WIP
Move Emails
Create and send basic email
Done
Open basic email
Done
Delete email
View list of appts
Done
Create basic appt
Done
Update contents /location
View Appt
Done
Create basic contact
Done
Update contact info
WIP
Create sub folders
Done
Send RTF e-mail
Open RTF e-mail
View Monthly formats
WIP
Create RTF appt
Accept/ Reject/T entative
Release 1
Limit Search to one field
Send HTML e-mail
Open HTML e-mail
Empty Deleted Items
View Daily Format
Create HTML appt
Propose new time
Add address data
Update Address Info
Delete Contact
Limit Search to 1+ fields
Set email priority
Open Attachments
Mandatory/Optional
Example story map created by Steve Rogalsky
http://winnipegagilist.blogspot.com
Release 2
Search attachments
Get address from contacts
View Weekly Formats
Get address from contacts
View Attachments
Import Contacts
Search sub folders
Send Attachments
Search Calendar
Add Attachments
Export Contacts
Release 3

User story map organization

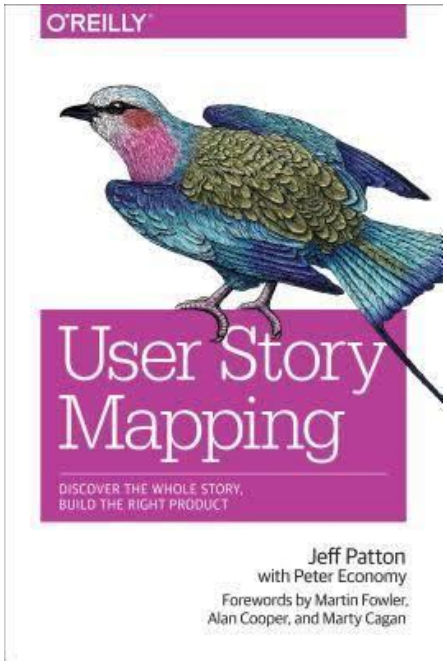
- Work as a team to improve map
  - similar tasks
  - similar people following similar goals
  - break off sub-systems
  - fill in missing pieces
- Work with candidate users to improve map
- Details can be recorded as use cases or other cards under task cards

Enhancements and tips

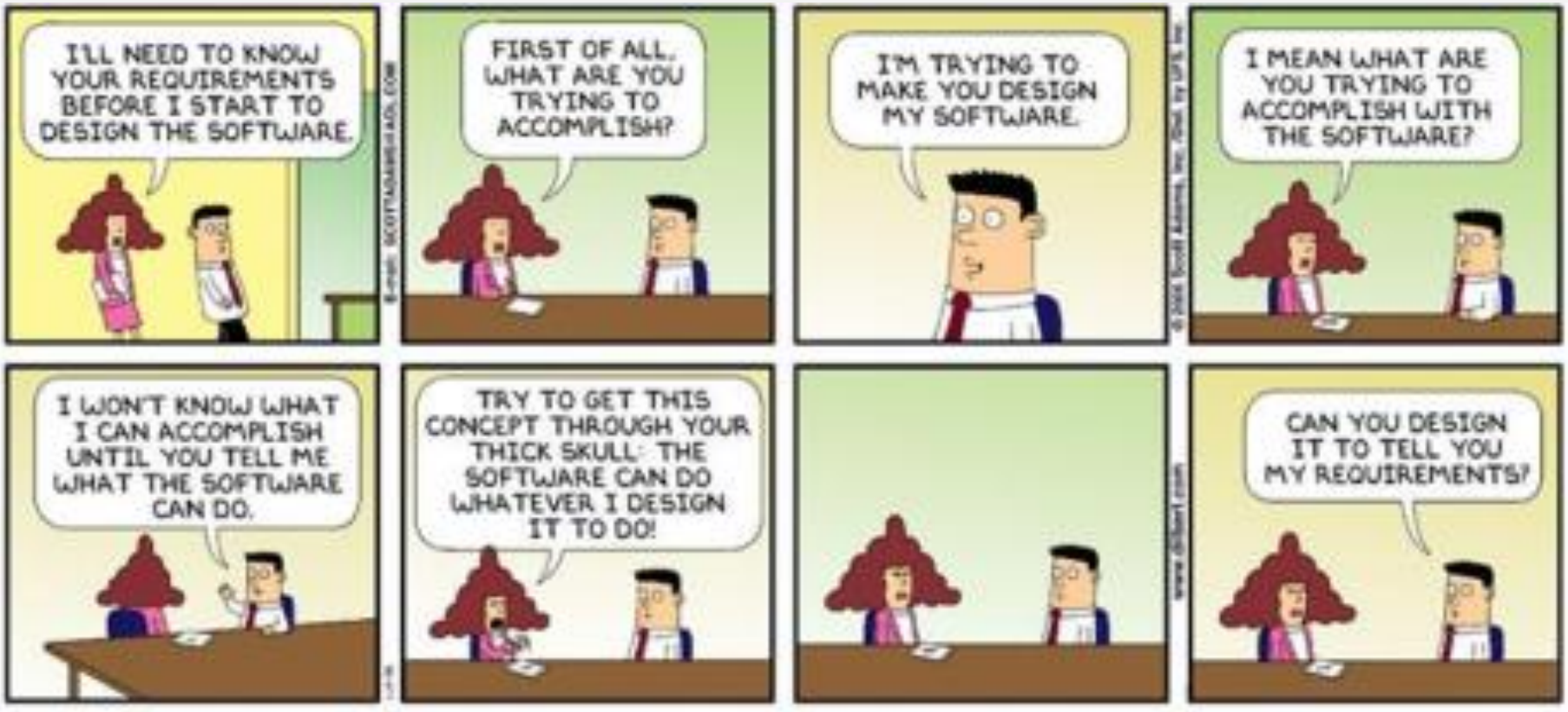
- Duplicate cards from map to kanban board
- Use bright yellow instead of dull yellow to show changes
- Use bright red to show blockers / impediments (Scrum)

Book

- Jeff Patton
  - http://www.agileproductdesign.com/
- The process uses stories, prototypes, and lots of conversations.
  - The story has the structure, the conversation has the meaning.
- "Handing off all the details about the story to someone else to build doesn't work. Don't do that."

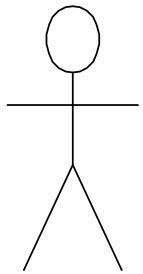


Brainstorming with use cases



Actors

- Should have been called roles.
- Actors initiate a use case / user story.
- Actor roles enforce the ability to do processes
  - Actors describe security group names that have **permission** sets.
  - For any two actors, one will have a unique use case that the other doesn't do.



Actor name

Actor table

- Identifies and classifies system users by roles and responsibilities
- Includes
  - Names of actual stakeholders (people/systems)
  - Description
  - Related job titles
  - Location
  - Level of expertise
  - Domain expertise
  - Frequency of use

Use case definition

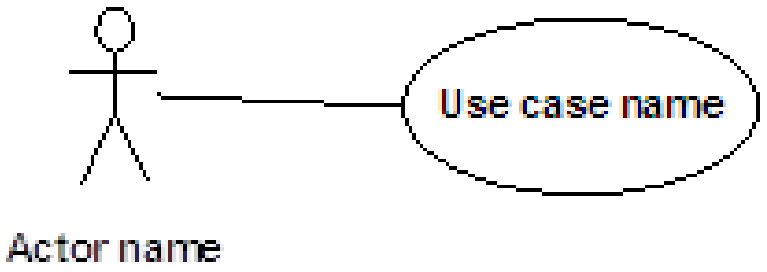
- A repeatable
- ordered sequence of tasks
- by an initiating party
- to support a business goal (provides value)

Use case project types

- Two types
  - Business - Goals for employee roles
  - System - Goals for a system under development or maintenance
- No good distinction in a diagram except to put domain box around system use cases.
  - name the box on top inside

Use case diagrams

- Only to show scope, granularity and triggers!
  - No sequences so no arrows!
- Split diagrams into readable sections.
- Only show actors who initiate use cases on diagrams.
- Keep lines from crossing when possible



Use case styles

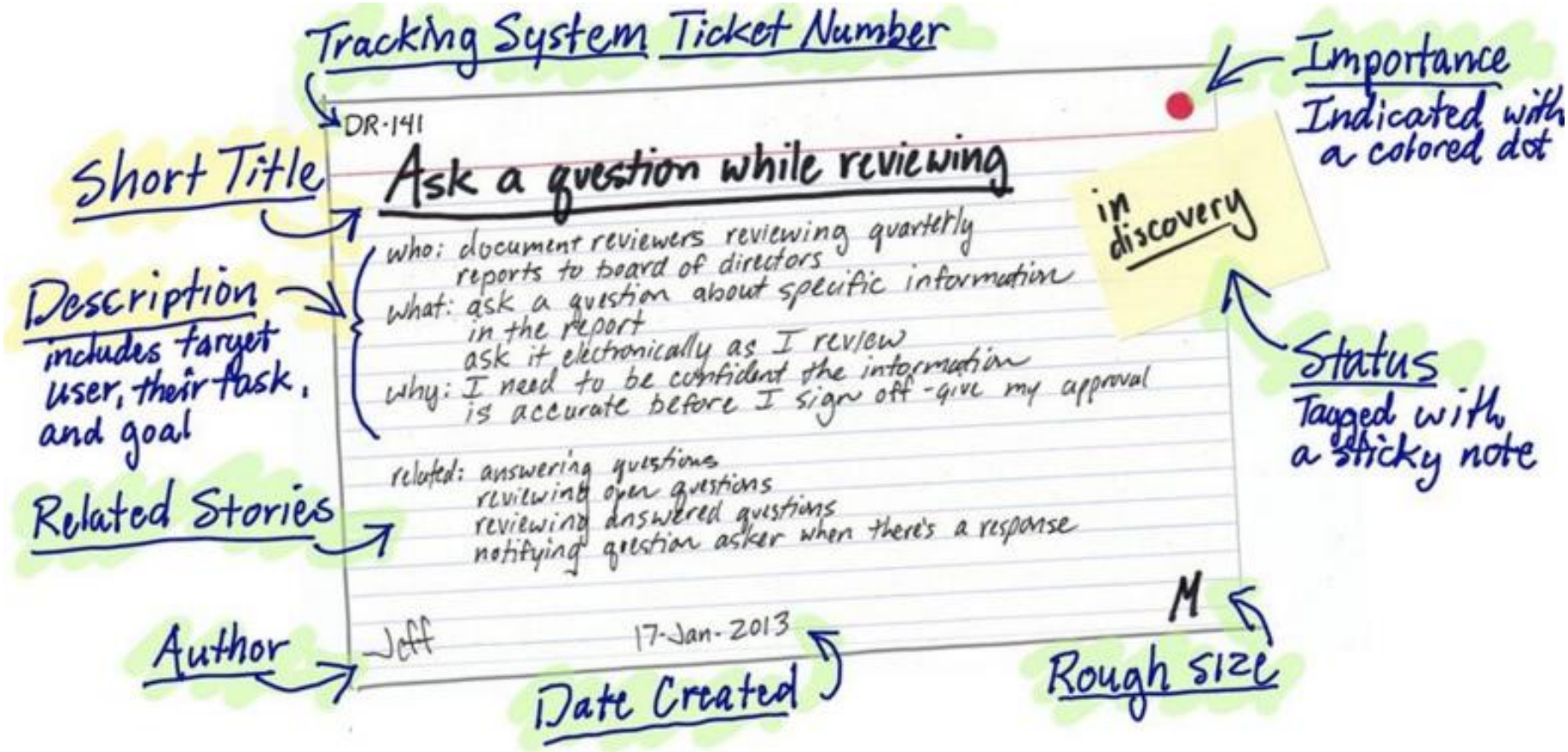
- Mix ‘n’ match
- Informal – the story
  - An elevator speech
  - Use for a table of contents
  - A descriptive sentence or paragraph
- Formal – all the facts
  - When it’s important to be clear
  - Up to several pages

Exercise: Elicit requirements

- Create a list of processes
- Create a list of roles

System delivery – user story

The detailed story card



User story granularity

- Objective
  - Use-case scenario type (**goal**, partial, group)
    - Operations (example: CRUD)
  - Work flow steps (number)
  - Business rule variations (number)
  - Data variations (number)
    - Data entry methods (prototype screens)

User story granularity

- Subjective
  - Major effort
  - Simple/complex
  - Deferred system qualities
- Spike (SAFe)
  - an enabler story to explore needed info or increase reliability

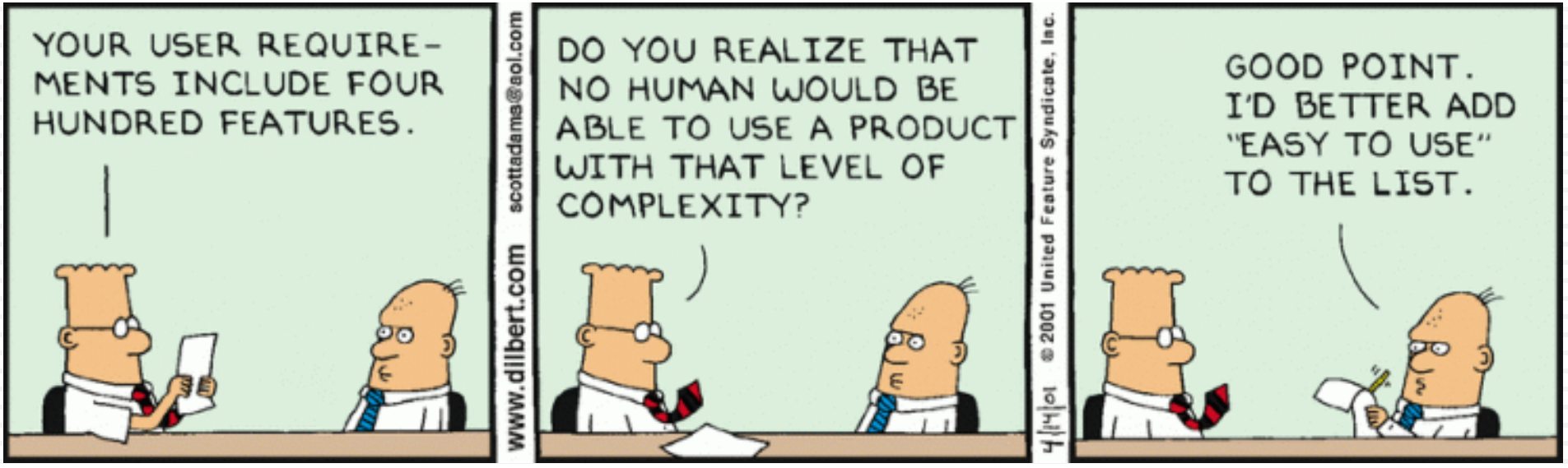
User story acceptance

- Acceptance criteria must be written with the customer.
- Detail is uncovered here and turns the informal requirement into a formal one.
- Written as confirmations, often on the back of the user story card.
- This is what the use case post-condition section is about

Process & data elicitation - prototyping

- **Sketching / wireframing / prototyping**
  - Analysis to design information will be elicited
  - Prototyping for eliciting needs, not
    - Prototypes for analysis
    - Prototypes for design
  - Types
    - Screens
    - Reports

System delivery – use case

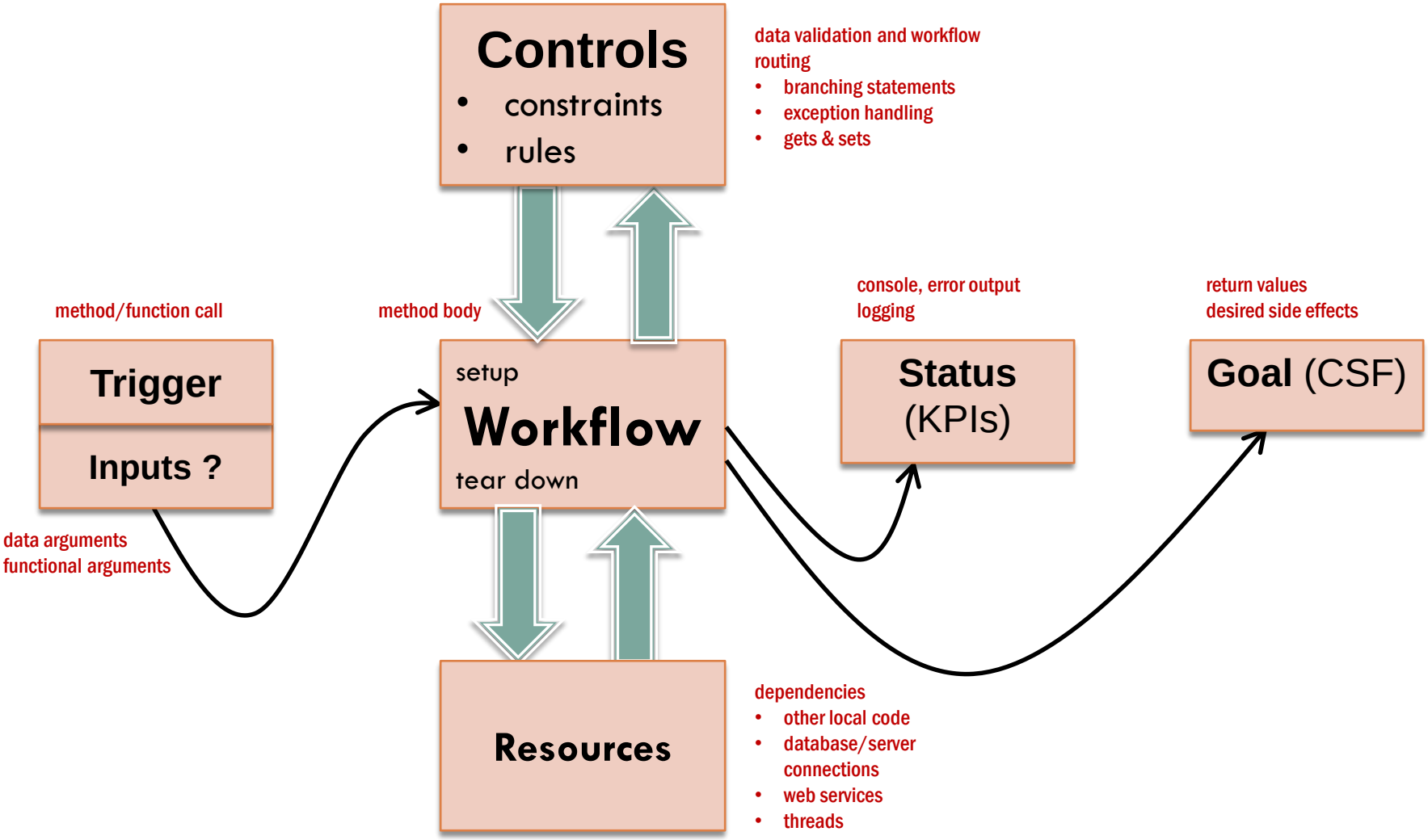


Process analysis breakdown

- role of initiator
- trigger
- steps
- rules
- relationships (cardinality, dependencies)
- status checks, goal

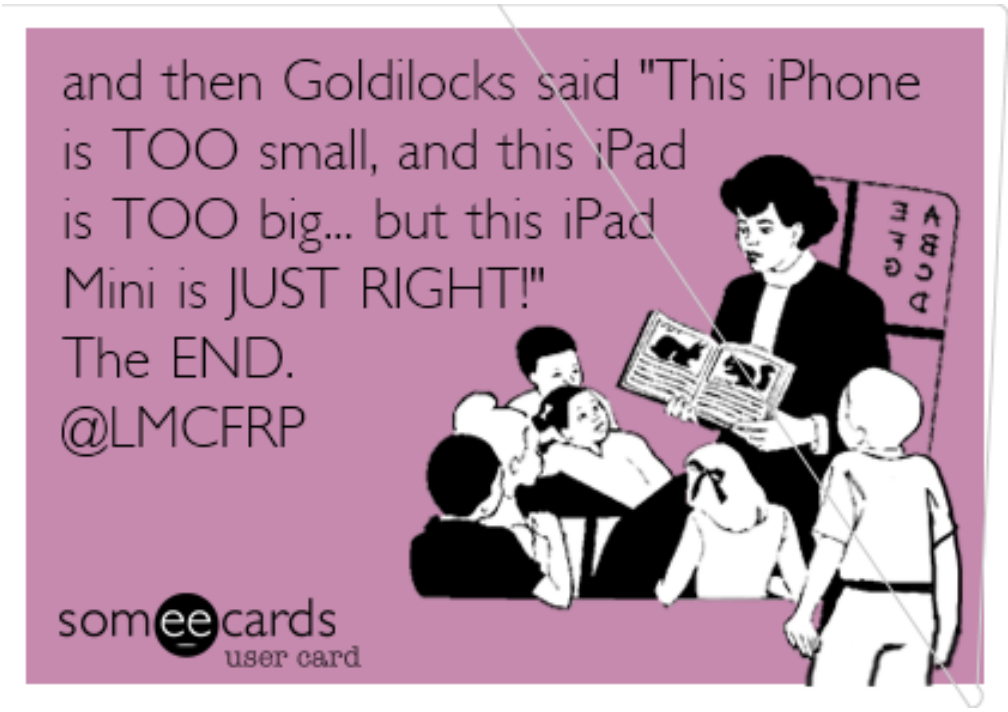
A generic process/service model

the process parts in computer language



Granularity

- Too small
  - Log in
  - Log out
  - Search
- Too big
  - Manage accounts
- Just right
  - Deliver package
  - Adjust account
  - Edit personal data



Granularity and traceability

- Strategic – business requirements
  - high level - mission, overall business model
  - low level - business case, project goals, epics, use case groups
- Analysis – user requirements
  - high level – achieves value, use cases, user stories
  - low level – performs a task, use cases
- Design - technical requirements
  - uses constraints to design a solution, risk
  - models that turn into code, databases, web pages…

Traceability

- Use words “roll-up” and “drill-down” to talk about relationships between levels with business language.
- All higher level requirements have lower levels
- All lower level requirements have higher level
- Traceability matrix
  - Assigns codes
  - Tracks relationships

ID	USER REQUIREMENTS	FORWARD TRACEABILITY
U2	Users shall process retirement claims.	S10, S11, S12
U3	Users shall process survivor claims.	S13

Granularity - traceability levels

- **Goal driven scenario**
  - What sequence of steps leading to a goal will give value to the business?
  - Scope like PM's WBS: 3 - 10 days of work
  - Lower level manager
  - Target for initial requirements document
- **Group of goals**
  - What broad grouping of goals do you want the system or <role> to do?
  - manage, handle, control, do, work with, take care of
  - Higher level manager

Granularity – breaking up groups

- Group of goals (compound goals) can be broken up into goals by
  - difference in final results
  - use of different business rules
  - separating simple and complex tasks
  - using different data sets
  - difference in middle tasks
  - seeing a CRUD combination

Granularity - traceability levels

- **Partial scenario** / group of tasks
  - What are the individual or named processes in the scenario ?
  - no or little business value by itself
  - Staff
- **Task**
  - What are the specific actions that need to happen that are the basic steps of the scenarios?
  - Staff SME
- **Design "requirement"**
  - an idea about how it should be built
  - Record as a design recommendation

Use case sections

- Metadata
- Flow of events / task sequence
- Optional
  - Preconditions
  - Post-conditions
    - Guarantees (minimal & maximum success)
  - Alternative flows – options
  - Alternative flows - errors

Use case metadata

- Required
  - Name – verb-noun syntax
  - ID and date
  - Actor(s)
  - Stakeholder originator
  - Priority (goal level and above, business value not personal)

Use case metadata

- Optional
  - Project
  - System / subsystem
  - Date updated
  - Cross-references
    - Business rules, data, prompts & menu text, designs
  - Level
  - Tracing
  - Index

Use case metadata

- Optional
  - Purpose
  - Explanations
  - Examples of ways to meet
  - Stability
  - Complexity
  - Stakeholders' interests

Use case detail

- Always a “happy path”
  - A success scenario
  - Problems will be captured later
- No conditionals
  - No if-then-else statements
  - Multiple partial sequences (loops) should be expressed as optional parts.
- Detail level
  - as much detail as possible without design
- No design (without constraints)
  - e.g. button click, submit buttons or anything that connects system to hardware, software, tools, or materials

Use case detail

- The course of events
  - The use case starts when the actor …
  - Response: The system …
- Possibly multiple actors could initiate the use case
- Numbering
  - Group one or more statements/tasks together
  - Smaller increments are better when you need to start in the middle at a specific spot due to an error.
  - Start with system does… usually.

Tasks (functional requirements)

- Tasks are sequenced low-level activities that can't be broken down any further
- The task statement contains
  - A responsible party/noun
  - The action/verb to be done
  - A description of the things/direct object which the verb acts on.
- A system functional requirement starts with "the <system> shall …"
- A business functional requirement starts with "the <role> shall …"

Alternative flows

- Done after structuring so numbering is done once.
- Two types
  - **Extension** points return back where you came from after an optional set of steps. <<extends>>
  - **Failure** points stop the use case, return you to a different point, or fix the problem and let you continue.
- Write in your choice of styles (informal, formal)
  - **Bad thing happens** (13, 15) – try to fix and return to 12.
- Include a return point or end the use case.

Pre- and post-conditions

- Pre-conditions
  - block the use case from doing the first step.
  - validate the state of the software before anything happens specific to that use case.
  - A log on is not a pre-condition in a system use case
- Post-conditions restate the important points connected to the goal.
  - Optional usually

Special requirements / notes

- Put things like SLAs and location or time needs in a special category.
- Non-functional requirements that are specific to this use case should be documented with the use case.
- Admin people can understand why a requirement should be met.

Using references

- Move out the details that are not functional
  - Small detailed parts are OK for clarity.
    - Sub points, mark the type
  - Use the specific document to capture reusable or complex rules, designs, etc.
    - Rules
    - Data dictionary
    - Designs - menus, screens
    - Externalized text – prompts, error messages
  - Use character style to show rules & data dictionary items

Exercise – create use case

- Do the use case for Make Transaction with credit
  - Metadata
  - Use case data
  - Flow of events
  - Pre-conditions
  - Post-conditions

System delivery – non-functional

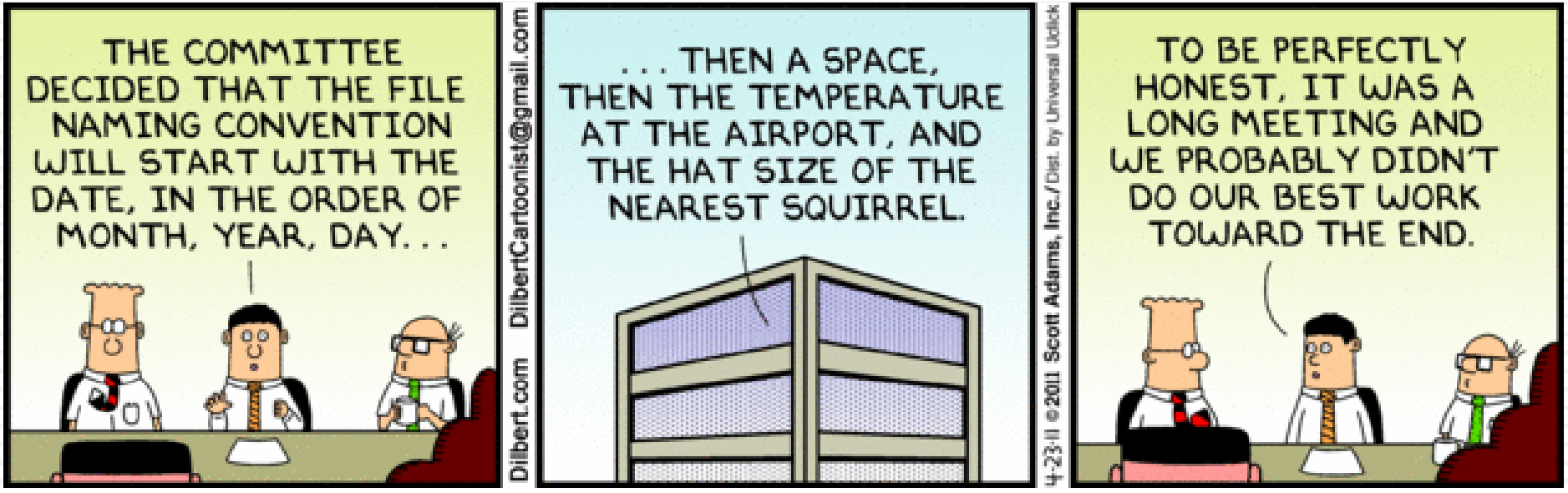
Non-functional requirements

- Placed in use case notes, another category, or other documents
- Use cases can have specific NF requirements
  - security issues
  - capacity needs
  - maintenance needs
- Tests
  - What adjective or adverb describes how all/a subset of the functional parts should behave?
  - What details do not affect what the functionality of the system or <role> does?

Non-functional requirement types

- Notes
- Security
- SLAs (expectations of performance)
- Quality or integration
- Data dictionary – data rules
- Process rules
- Interfaces
- Design recommendations
- Prompts, menus, and messages

System delivery - data



Data analysis

- description
- type
  - simple, complex
- rules
  - inputs
  - outputs
  - bounds, members
  - relationships (cardinality, dependencies)
- amount collected over time
- use
  - read/write
  - aggregated/processed report

Process and data models

	Business (analysis)	Technical (design)
Roles	User story, use case diagram	RACI chart, actor table, stakeholder list, security roles
Triggers	System level sequence diagram	API
Workflow steps	Use case, flow chart, prototype	Flow chart, prototype, ADM, SOP manual
Workflow rules	Use case, rule lists	Rule tables
Workflow relationships	Use case, flow chart	Flow chart, ADM
Tests	Use case	Testing docs
Data interfaces	Data flow diagram	API
Data definition	Data dictionary	Schemas
Data rules		
Data relationships		

Verify data rules

- Testability check -play out differently
  - Use data range boundary values
  - Substitute more extreme values
  - Repeat more times
  - Make environment worse

Data dictionary

- Formal document recording data entities
- Describes
  - Names of individual types of data
    - Name, address, city, state, zip, phone, …
  - Constraints / rules
    - Validation
    - Dependency
  - Related entities
    - Customer, internal rep, …
  - Examples

System delivery - process rules

Constraints

- **Constraints** on processes are known **before** analysis
  - Must follow HIPAA privacy
  - Must follow WAI-ARIA on web site
  - Must comply with PCI to process credit cards
- Also called
  - General requirements – legal…
  - Technical requirements – infrastructure…

Business rules

- **Business rules** are used **during** process decisions or data choices driven by best practices.
  - Workflow based on variables that produce an output.
  - No mention of any automation.
  - Typically discovered in elicitation with staff or SME

Process rules

- More often called business rules. Data rules are included in data dictionary.
- The **functional** part of a requirement…
  - The system shall print a report …
- …May be modified with a **rule** part
  - …**If**/**when** sales are > 100,000 then using rate chart DF3
  - …Use rate chart DF3 **when** sales are > 100,000
  - …On Thursday
  - …When I say so

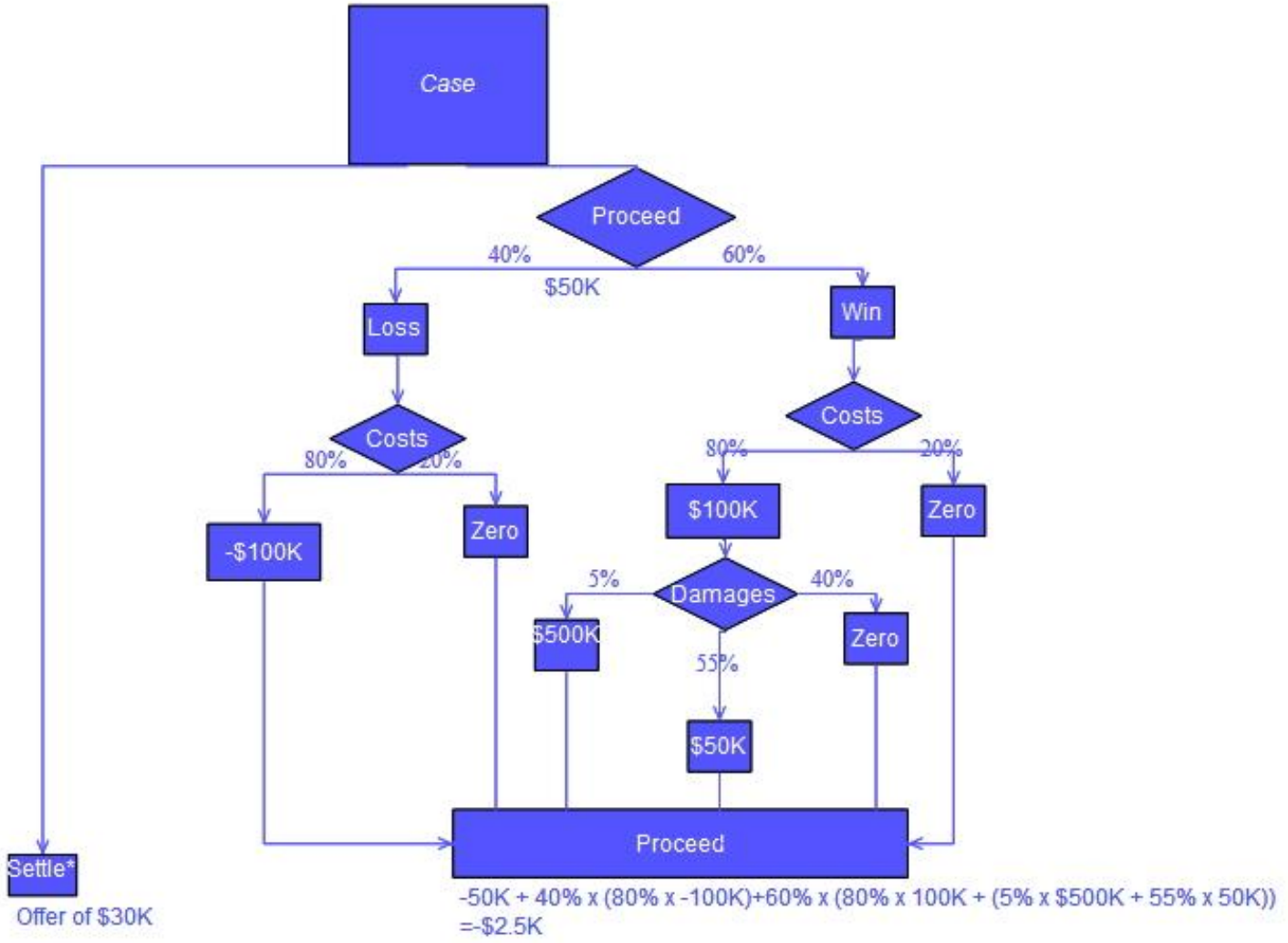
Business rules

- Made up of facets/variables and result/output
  - Facet – customer type
    - Loyal = purchased items twice in the last 50 weeks
  - Facet – additional product purchase
    - Product = web hosting
  - Result – discount on additional product
    - Discount = 15%
- Multiple facet rules are modeled with
  - rule tables
  - decision tree – often associated with probabilities

Rule tables

Facet	Facet	Facet	Facet	Result
Client	**Amount due**	**Average purchase**	**Last purchase**	**Type of e-mail**
Region 1 client	<10	100	<50	Follow-up
Region 1 client	<10	100	>50	Follow-up and discount coupon
Region 1 client	>10	100	<50	Follow-up
Region 1 client	>10	100	>50	Follow-up and small gift coupon
Region 2 client	Etc.			

Decision tree



Specifying rules

- Short one-time use rules are better included in use case documentation below the functional statement
  - The system validates the amount.
    - RULE: **Available funds** - Account balance is larger or equal to than amount requested.
    - RULE: **Daily total withdrawal**: Amount requested is less or equal to **R24 MAXIMUM WITHDRAWAL AMOUNT**
    - RULE: **Increments** – Must be in increments of $20.
- Larger rule tables, decision trees, or reused rules are better in a separate document and referred to.
  - **R24 MAXIMUM WITHDRAWAL AMOUNT**: $500 per day starting at midnight.

Verify business rules

- Describes one decision
- Is not a task in a workflow
- Described by measurable facts or tables of facts
  - Facets (units, variables) and quantities
  - reorder if shelf quantity =< 5 SKUs (when)
  - reorder par quantity of SKUs (how much)
  - reorder using Prime 2-day shipping (how sent)
  - reorder from approved vendor list (who)

System delivery – prioritization

Prioritization

- Value to project management
  - Selection of scope based on budget and schedule.
- When to prioritize
  - Early
  - Progressively
- How to prioritize
  - It's not how important to the stakeholder it is, it's about the business
  - Don't ask the stakeholder for "their" priority

Prioritization - popular

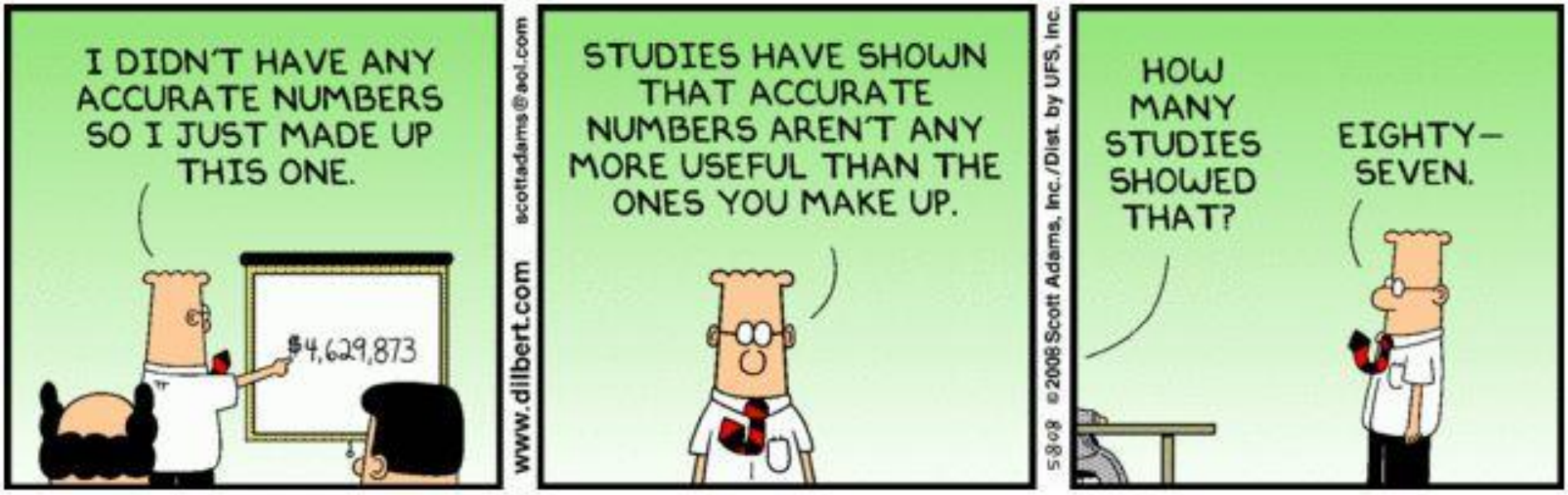
- Levels – 1,2,3 or mandatory, desirable, nice to have
- Kano – Dissatisfiers, Satisfiers, Delighters
- Analytical Hierarchy Prioritization – compares pairs
- WSJF - User business value, Time criticality, Risk reduction/opportunity enablement
- MoSCoW = must have, should have, could have, want to have but won't have this time

Prioritization – ITIL

- Essential
  - Scope of use (impact)
    - how much of the business will it improve?
    - how many of the staff will it help?
    - Externally equated to target market
  - Business value (urgency)
    - how much do you wish the business had it now?
    - how bad will the business look if it fails in the future?
    - what level of person is asking for it?
    - Externally equated to price willing to pay
- Optional
  - Anything else important to the business

Prioritization

- Using weighted averages
  - "I'll give it a 9.27"
  - No units = no metrics = no standard
  - Use for understanding but not communications



Prioritization

- Other categories to use
  - ROI
  - Satisfaction
- Negative risk = Impact * Urgency * probability of failure

Process analysis - structuring

Structuring

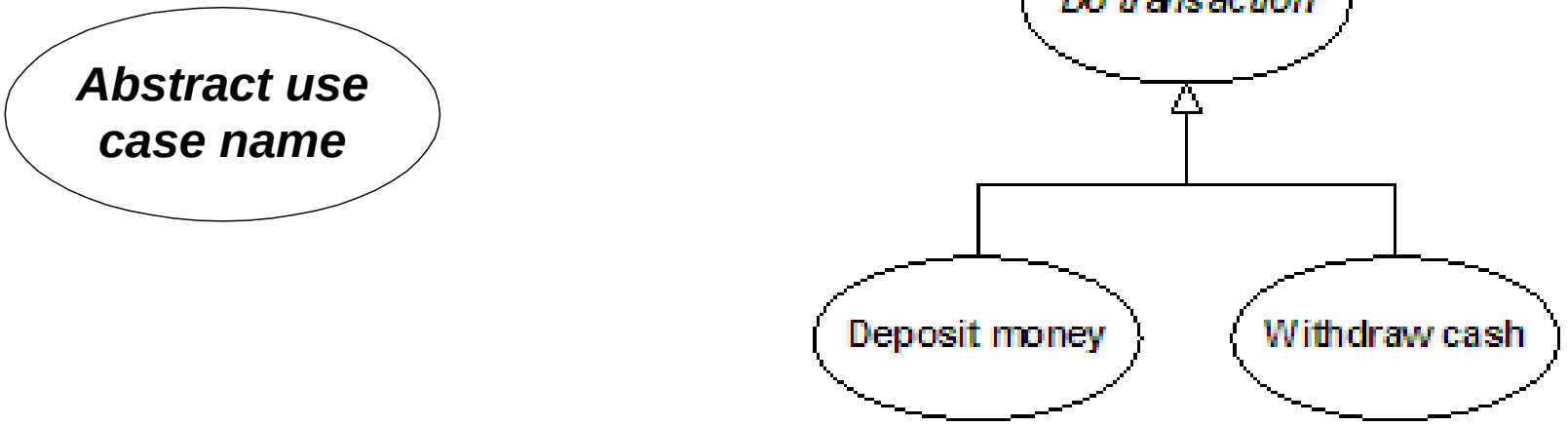
- Changing process models for better understanding or communication
- Improving documentation
- Follows similar design principles as in software design
- Slicing up organization for project management

Product release roadmap

- Targets benefits over time
- Derived from slicing story maps into rows
- Each release
  - give a name for its purpose
  - describe benefits to the business
  - describe benefits to users
- Commit to user needs / stories
  - not to features

Use case structure - groups

- Use grouping when it cleans up a use case diagram and makes it easier to understand.
- Produce several versions of the functionality



Use case structure – partials

- Use names when users expect that name.
- Use names when options are important to see.
- Task level names (includes) are shown as a part of a standard use case ellipse with a dependency arrow and
  stereotype (category).

Include and extends – partials

- **Include** is required (dashed arrow goes to what is required to be done)
- **Extends** is optional (dashed arrow goes from what can be done to what it must be a part of)
- Usually a scenario use case is broken into tasks and shown but not always.
- Do not use these until all scenario level use cases have been detailed!

Include vs. different use case

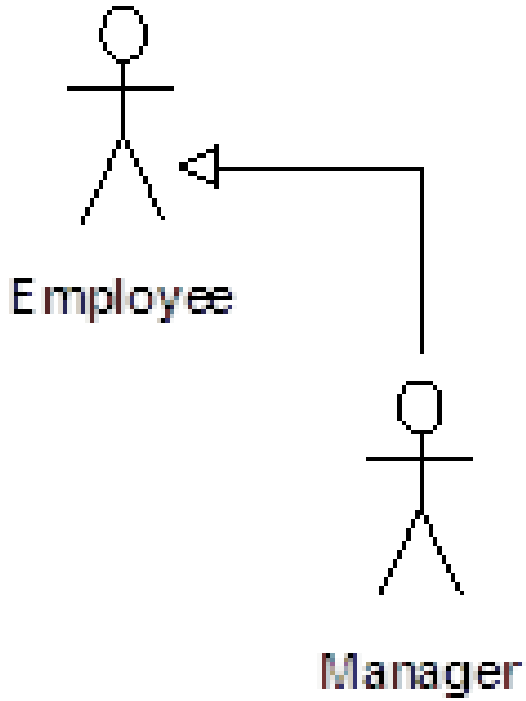
- Write includes into use cases as bolded names in course of events
  - **4. Do Log On (SF24)** workflow returns here
  - 4. <<includes>> **Do Log On (SF24)**
- Alternate flow (extends) – separate section
  - **(#3) Print Receipt (SF33)** workflow returns here
  - (#3) <<extends >> **Print Receipt (SF33)**

Structuring for security roles

- Security is a use case wrapper around other use cases
  - Start session (authenticate)
  - <<include>> Do secure process
  - End session (clean up and deaccess)
- This is allows for <<extends>> Do another secure process as an option
- Don't do
  - <<include>> Start secure session
  - Process tasks…
  - <<include>> End secure session

Role generalization

- Actors can share use case initiation
- Show with generalization arrow



Use case numbering / id symbols

- By system component (ATM1, ATM2, B1, B2, INV1)
- A group of use cases
  - *ATM2 – Do A Transaction (generalized partial goals)
  - ATM2.1 – Do Withdrawal (one partial goal
  - ATM2.1.5 Do Withdrawal step 5
  - ATM2.1.5b Do Withdrawal step 5 and 2nd system requirement in step.
- A partial use case with system security
  - SS+PUR2.1 Check shopping cart contents (include this)
  - +*PUR2 Check account property (include one of a group of partials)
  - SS#5+PUR13.1 – Purchase item but check shopping cart contents at step 5. (include which focuses on one step)

Design


WALLY, WE DON'T HAVE TIME TO GATHER THE PRODUCT REQUIRE-MENTS AHEAD OF TIME.
I WANT YOU TO START DESIGNING THE PRODUCT ANYWAY. OTHERWISE IT WILL LOOK LIKE WE AREN'T ACCOMPLISHING ANY-THING.
OF ALL MY PROJECTS, I LIKE THE DOOMED ONES BEST.
NEWS
S. Adams
www.unitedmedia.com
5/9/97 © 1997 United Feature Syndicate, Inc.

User scenarios

- A short real-life script of what a distinct role / system does.
- Several different personas can be used for each actor/role

Brad

Student

Brad is a student with a full-time job who has not brought any lunch for his one-week class and expects to eat at a restaurant for lunch. He also gets free snacks.

Doug

Employee

Doug is a employee near the vending area who typically has enough spare change to make a purchase and passes by the area several times a day but has brought his lunch. He avoids free snacks.

User scenarios - misuse

- People who use the system in a way it was not intended to be used for

Tor

Hacker

Tor knows how to hide his IP and listen in to unencrypted conversations. He sells data to make a living and will try to steal your data through viruses if he can.

Waldo

Social eavesdropper

Waldo will watch your social accounts and figure out when you are gone so he can visit you and take more than your identity He specializes in cities near Los Angeles.

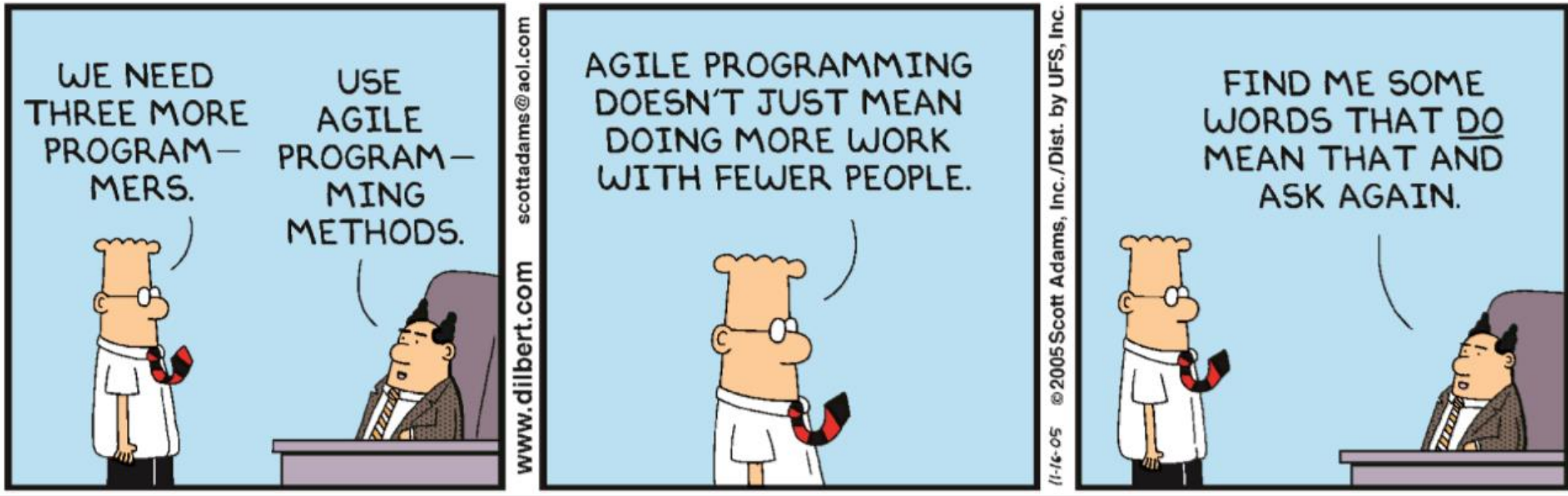
User scenarios

- A user scenario tells a story about a main character with a problem or goal
  - Describes how that character reaches their goal
  - contains important facts
  - describes external context
  - describes goals and concerns of our character
- include interesting plot points that help us envision important aspects of the system
- A scenario can gloss over uninteresting details

Exercise – User scenarios

- User scenarios
- User tasks

Testing


WE NEED THREE MORE PROGRAM-MERS.
USE AGILE PROGRAM-MING METHODS.
www.dilbert.com scottadams@aol.com
AGILE PROGRAMMING DOESN'T JUST MEAN DOING MORE WORK WITH FEWER PEOPLE.
11-14-05 ©2005 Scott Adams, Inc./Dist. by UFS, Inc.
FIND ME SOME WORDS THAT DO MEAN THAT AND ASK AGAIN.

Testing

- Plan for testing
- Test early, test often
- Test a little at a time
- Collaborate with developers and users

Testing types

- Verification during the process / modeling
  - are we doing this right?
- Validation after the process
  - did we get what we said we wanted?
  - Lessons learned / Post mortem
  - Retrospective
  - After each iteration

Verify analysis language

- Can the **system** functionality be applied to a phone interface?
  - If you talk about a GUI then that's design.
  - Design language is appropriate when there is a project constraint to use a specific design

Verify testability

- Does your complete scenario meet the walk-away test?
  - Walk up, do it, walk away
- Does the system return to the **same state** it was in before you started the use case?
  - I can do it, then you can do it.
  - If not, then it's part (task, function) of a bigger use case

Verify testability

- Programmers will code to your requirement and then test with it. Test case documentation is unnecessary if use cases are done well.
- If you talk about **how** something is done, that's a rule.

Verify completeness

- Can you define a requirement (at a higher level) that summarizes a group of requirements?
- Can you define a requirement (at a lower level) that is a part of the requirement?
- Are there other higher scenarios that would use this use case?
- Are there other lower scenarios that would use this use case?

Verify completeness

- After completing use case – role playing game
  - Person = role
  - Data represented on Post-it notes / listed on pad
  - Designs sketched on paper, hold up when active

Verify granularity

- Strong actionable verb
  - Vague verbs indicate a **group** of use cases
- The business gains **value**
  - What was the **goal** that was achieved?
  - No value at the end **to the business** indicates that this small group of tasks is a part of another use case

Verify granularity

- Any use of conditional logic for workflow will indicate separate use cases when
  - The outcome is different
  - Steps are skipped
  - Steps are not always included
  - A rule is used to alter workflow
- Break use case in to individual use cases and then structure later