Software Engineering Project Workshop (SENG202)

Matthias Galster

Introduction to Phase 2

August 10, 2020

UC
UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Reminders

- Time left for Phase 2
  - 4 weeks
- Submit your weekly individual reflection (Mondays, 5:00pm)
- Keep logging as you go
- Labs
  - Same allocation of teams as last week
  - Tutorial session and quiz: JavaFX; stand-ups and feedback sessions

Exercise to improve communication in meetings

Task 1: What do I know about communication in meetings?

Task 2: How do I perceive examples + tutorials on meeting communication?

Task 3: How do others perceive videos? Can I learn from them?

Task 4: How does my team communicate in our meetings?

Task 5: What do I think about our meetings as an “outsider”?

Task 6: What do my team mates think about our meetings?

Task 7: Did my understanding of communication in meetings change?

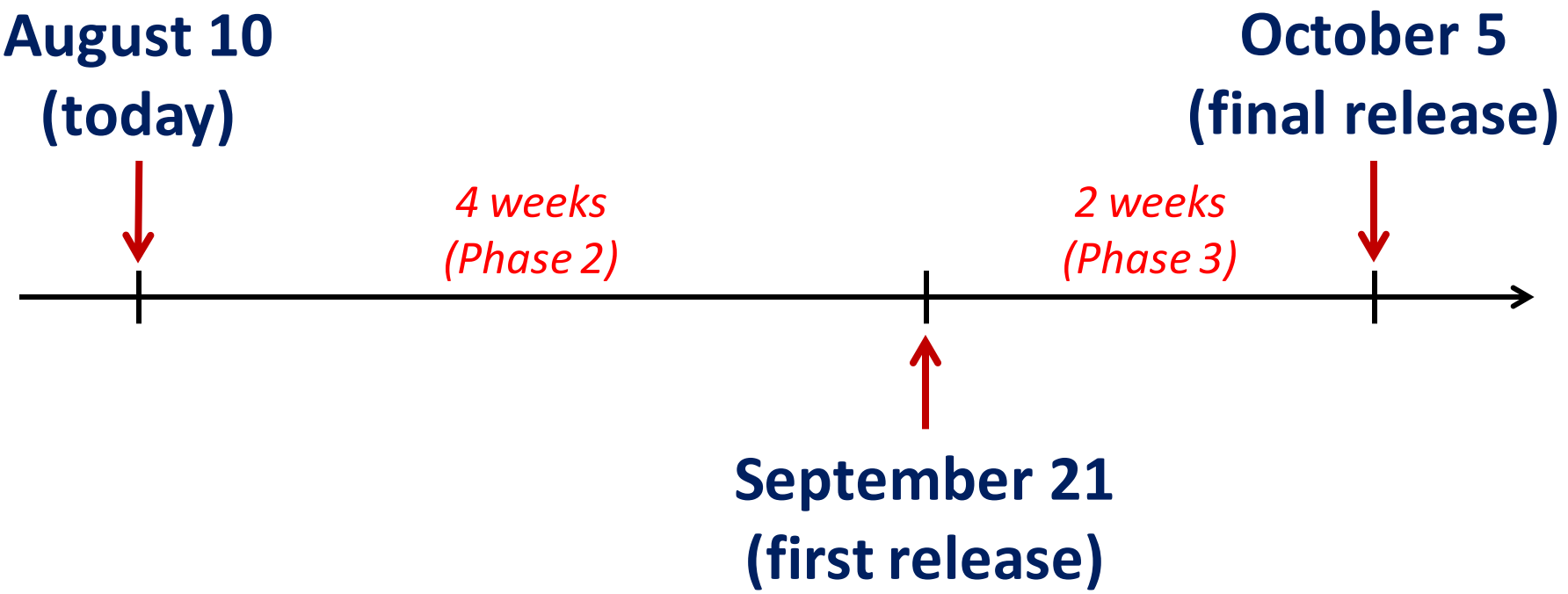
Exercise over the next 5 weeks[1]

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
July 20	July 21	July 22	July 23	July 24	July 25	July 26
[Task 1] Assess initial knowledge (questionnaire)[2]						
[Task 2] Watch and comment on brief tutorial and example videos on AVW platform (reflection 1)[2]						
July 27	July 28	July 29	July 30	July 31	August 1	August 2
[Task 3] Review and rate comments on videos made by your class mates on AVW platform (reflection 2)[2]						
August 3	August 4	August 5	August 6	August 7	August 8	August 9
				[Task 4] Video-record a team meeting		
August 10	August 11	August 12	August 13	August 14	August 15	August 16
[Task 4] Video-record a team meeting (all team members; teaching staff will provide support)						
[Task 5] Watch and comment on team meeting recording on AVW platform (for own team only)[2]						
August 17	August 18	August 19	August 20	August 21	August 22	August 23
[Task 5] Watch and comment on team meeting recording on AVW platform (for own team only)[2]						
[Task 6] Review and rate comments on meeting recording made by team mates on AVW platform (reflection 3)[2]						
[Task 7] Assess knowledge (questionnaire)[2]						

[1] Course exercise, but with your consent we will use anonymized data to study learning experience (consent form will be available at the beginning of the exercise); if no consent: still same activities, but data won't be used to study learning experience

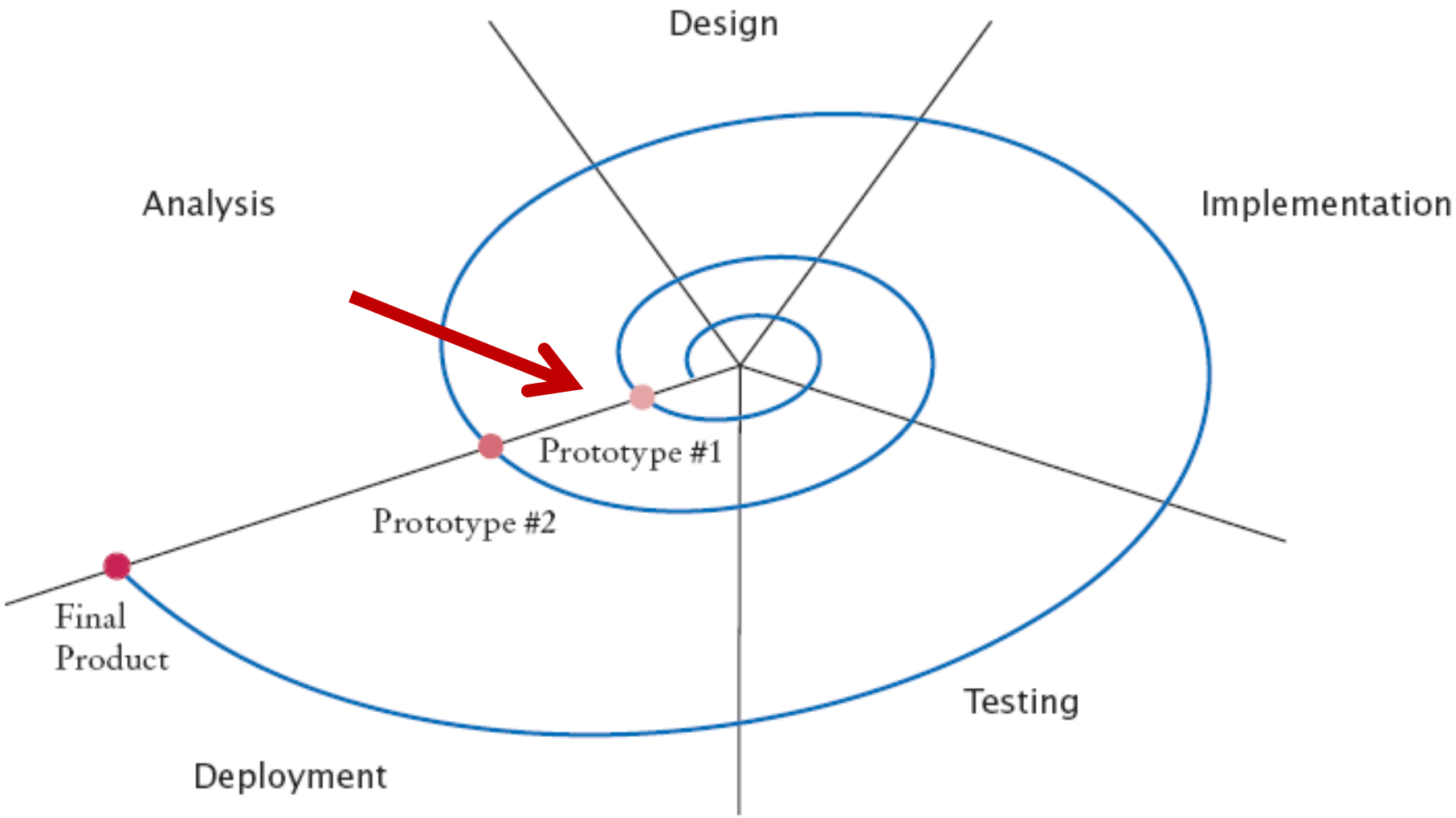
[2] Individual activity in your own time, but log as #class activity

What is left


August 10
(today)
October 5
(final release)
4 weeks
(Phase 2)
2 weeks
(Phase 3)
September 21
(first release)

Phase 2

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
August 10	August 11	August 12	August 13	August 14	August 15	August 16
August 17	August 18	August 19	August 20	August 21	August 22	August 23
August 24	August 25	August 26	August 27	August 28	August 29	August 30
August 31	September 1	September 2	September 3	September 4	September 5	September 6
September 7	September 8	September 9	September 10	September 11	September 12	September 13
September 14	September 15	September 16	September 17	September 18	September 19	September 20
September 21						


Design
Analysis
Implementation
Prototype #1
Prototype #2
Final
Product
Testing
Deployment

Focus: Implementation
(including testing)

Goals

- Produce first “potentially shippable” version of product
- Get better understanding of product

During implementation, understanding of problem and product will change (“*nothing is permanent but change*”)

Deliverables

- Updated (partial) design document
- Product
- Reflections and logs
- Presentation

Deliverables

- Updated (partial) design document
  - Testing procedures – new
  - Description of current product version – new
  - Updated risk analysis and project plan
  - Change log as table: section numbers, section titles, change, who changed
  - Same naming conventions as in Phase 1

- Product

- Reflections and logs

- Presentation

Updated (partial) design document

Executive summary (update)

Change log (new)

1. Business and system context (update)
2. Stakeholders and requirements (update)
3. Acceptance tests (leave empty but maintain*)
4. GUI prototypes (leave empty but maintain*)
5. Deployment model (leave empty but maintain*)
6. Detailed UML class diagram (update)
7. Testing procedures (new)
8. Current product version (new)
9. Risk assessment (update)
10. Project plan (update)

Map to tasks

References (update)

Appendix (update)

*Final version of design document (part of Deliverable 3) needs to include complete and up-to-date information.

Deliverables

- Updated (partial) design document
  - **Testing procedures – new**
  - Description of current product version – new
  - Updated risk analysis and project plan
  - Change log as table: section numbers, section titles, change, who changed
  - Same naming conventions as in Phase 1

- Product

- Reflections and logs

- Presentation

Testing procedures

- Test protocol to increase confidence in the quality of product

- Highlight three aspects in three different subsections
  - Functional testing
  - Quality testing
  - Discussion (overall test results and test coverage)

Recording tests

- Functional testing and quality testing
  - Manual testing (focus on high priority items)

Item	Description
Test (or ID)	Test case (link to acceptance tests, if applicable)
Description	Description (as in acceptance test, if applicable)
Tester	Name of tester (responsibility for acceptance test, if applicable)
Result	Outcome of test
Pass/fail	Based on result, did the test pass or fail (based on acceptance criteria)
Criticality	See acceptance tests
Use case	Link to use cases, requirements

  - Automated acceptance testing (Cucumber) – summary report

- Individual (automated) unit tests not recorded in test protocol

Discuss overall test results

- Were all (manual + automated) acceptance tests performed
  - If not, which ones were not performed and why
- Did (manual + automated) acceptance tests cover diverse cases
- What (manual + automated) acceptance tests did fail
  - Are failing acceptance tests “critical”
  - What will be done about failing tests
- Classes which are not complemented by unit tests (and why)
- Etc.

Discuss test coverage

- Test coverage for acceptance tests
  - Coverage of requirements, use cases, stakeholders, etc.
  - For whole system
- Test coverage of unit tests

  - Code coverage (can be obtained from coverage metrics and tools in IDE)
- Low test coverage of parts of the system should be explained
  - E.g., low coverage of acceptance testing could be due to lack of time, missing acceptance tests or test cases, etc.
  - E.g., low coverage of unit testing could be due to not unit testing GUI classes or classes that are not model classes, many getters and setters, etc.


We received your delivery on time, but we couldn't find the tests
Tests ?
What tests?
You know, the unit tests. The ones for testing the application layer, Quality Assurance and all
Ah, yes, the unit tests...
There aren't any unit tests
No tests ? But what about the guidelines...
How do you check that everything works correctly?
Um...
IT'S MAGIC
CommitStrip.com

Deliverables

- Updated (partial) design document
  - Testing procedures – new
  - **Description of current product version – new**
  - Updated project plan + other updates to design document
  - Change log as table: section numbers, section titles, change, who changed
  - Same naming conventions as in Phase 1

- Product

- Reflections and logs

- Presentation

Description of current product version

- Highlight three aspects in three different subsections
  - Requirements and features implemented in this release
    - Make use of use cases, requirements, etc. (helps update design document)
  - Features you are most proud of
    - Markers may pay particular attention to
    - Accompanied with a brief justification for why feature is noteworthy
  - Requirements and features not implemented in this release
    - From the product description (and the feature packages)
    - May be implemented later

Product scope after Phase 2

The release delivered at the end of Phase 2 should implement all core features of your product. Core features depend on how requirements have been prioritized. Your project plan should give you an idea of what features you can implement in Phase 2. There is no fixed list of expected features that we would check at the end of this phase. A high quality product is preferred over a product with lots of features. However, in order to keep on track with the implementation of your product your first release should at least implement a main GUI that allows users to load existing data, shows that crime data in a “Raw data viewer”, allows the user to filter data, perform some analyses, etc. You may need to implement more features, depending on your project plan. The scope of the product varies for each team, depending on how you allocated features to your available time, how you prioritized requirements and what difficulties and risks you identified.

Deliverables

- Updated (partial) design document

- Product (make sure it compiles, runs, can be imported)
  - Exported project (from IDE) + executable product (as .jar)
    - `seng202_2020_team<team>_<phase>`
  - Documented source code, Javadoc, JUnit tests, Cucumber tests
  - README (how to install/deploy/run app, how to import/build project)
    - `seng202_2020_team<team>_<phase>_README.txt`
  - Other resources required to run the app

- Reflections and logs

- Presentation

.jar

You do not need to perform any action inside of your IDE for this to occur. We are using a build automation tool (Maven). To generate a executable jar, navigate inside the terminal to the root directory of your project (where the `.pom` file is located). Execute `mvn package`. Once this is complete you will find your executable jar within your `/target` directory. If your jar does not execute it might be caused by not having the correct naming convention on the following lines.

```
<manifestEntries>
    <Main-Class>seng202.group#.Main</Main-Class>
    <X-Compile-Source-JDK>1.8</X-Compile-Source-JDK>
    <X-Compile-Target-JDK>1.8</X-Compile-Target-JDK>
</manifestEntries>
```

You must ensure that the main class variable is set to match where your main Java class file is located (the one containing the main). Additionally, if you have set up your continuous integration inside eng-git correctly, you could download the artifacts from the website as they will always be located there (see image below, on the right) after a successful build.

Deliverables

- Updated (partial) design document
- Product
- Reflections and logs
  - As in Phase 1
  - Re-read instructions (many invalid logs in Phase 1)
- Presentation

Deliverables

- Updated (partial) design document
- Product
- Reflections and logs
- Presentation
  - See next slide

Presentation (Deliverable 2)

- ~15 minutes
  - During the labs of the week of the due date; no need to submit slides
  - All team members present, 25% penalty for not presenting

- Content
  - Overview of project, i.e., purpose and what user expects to get out of it
  - Demo of features that are working
  - Testing and quality assurance procedures
  - High-level project code overview, likely via a UML class diagrams
  - Status of your implementation
  - Problems faced, lessons learnt, changes, etc.
  - What will be done next

Working in increments and iterations

After performing an initial analysis in Phase 1 and creating a first design document you should be able to understand your system, its scope and basic requirements. Also, you should now have an initial architecture of your system. Based on the artefacts created in Phase 1, the goal of Phase 2 is to implement, test and document a first version of your product.

During Phase 2 it will become necessary to update the deliverables from Phase 1 (e.g., major use cases, requirements, risks, project plan, team policies) since your understanding of the system scope, functionality and technical complexity will increase.

Based on this first release you can obtain feedback and suggest improvements for further analysis. Also, the first release will help you validate interfaces, test performance etc.

On quality

The release must be fully functional (with regard to the implemented features). Therefore, it must also implement exception handling to handle unexpected behaviour. However, this release will generally have more bugs and unexpected (or “exceptional”) behaviour than the final release. Also, it may still have some performance issues.

The release and all features must be tested. The release must be usable for demonstrations and previews to prospective customers. We expect unit testing and acceptance testing. We expect high quality unit tests which achieve high code coverage. Unit tests may include tests for getters and setters of classes, but the focus should be on testing the actual functionality of a class. For unit testing you should use tools provided as plugins for the IDE (JUnit testing framework, test coverage tools, etc.). It is good practice to write unit tests for classes before implementing classes (see also “Test-driven Development”). For acceptance testing functionality of your product, use acceptance tests as defined in Phase 1.

All features must be documented (Javadoc and inline comments).

On documentation

- Stakeholders do not want documentation, they want answers
  - Documentations should help answer questions of stakeholders
  - In other words, documentation should address concerns

- Why maintain documentation: to inform, demonstrate, persuade
  - Describe use, operation, maintenance or design of software
  - Facilitate maintenance, train new employees
  - Assign responsibilities for processes and practices
  - Help customer use services without contacting development organization
  - Get new customers (docs answer their questions about product)
  - Comply with regulations, requirements, standardize practices
  - Enforce thought process, “thinking”
  - Etc.


What was this function again? What the hell was I trying to do?
And where does that variable come from? I don't event remember...
Oh, good, I left a comment...
// Document this when back from vacation
CommitStrip

How to submit

- Through Learn
  - By 5:00pm on the submission date
  - No drop-dead date

- As ZIP containing all deliverables
  - `seng202_2020_team<team>_phase<phase>.zip`
  - Must contain all the previously mentioned files and resources

Reminder

Start now

Ask for feedback early

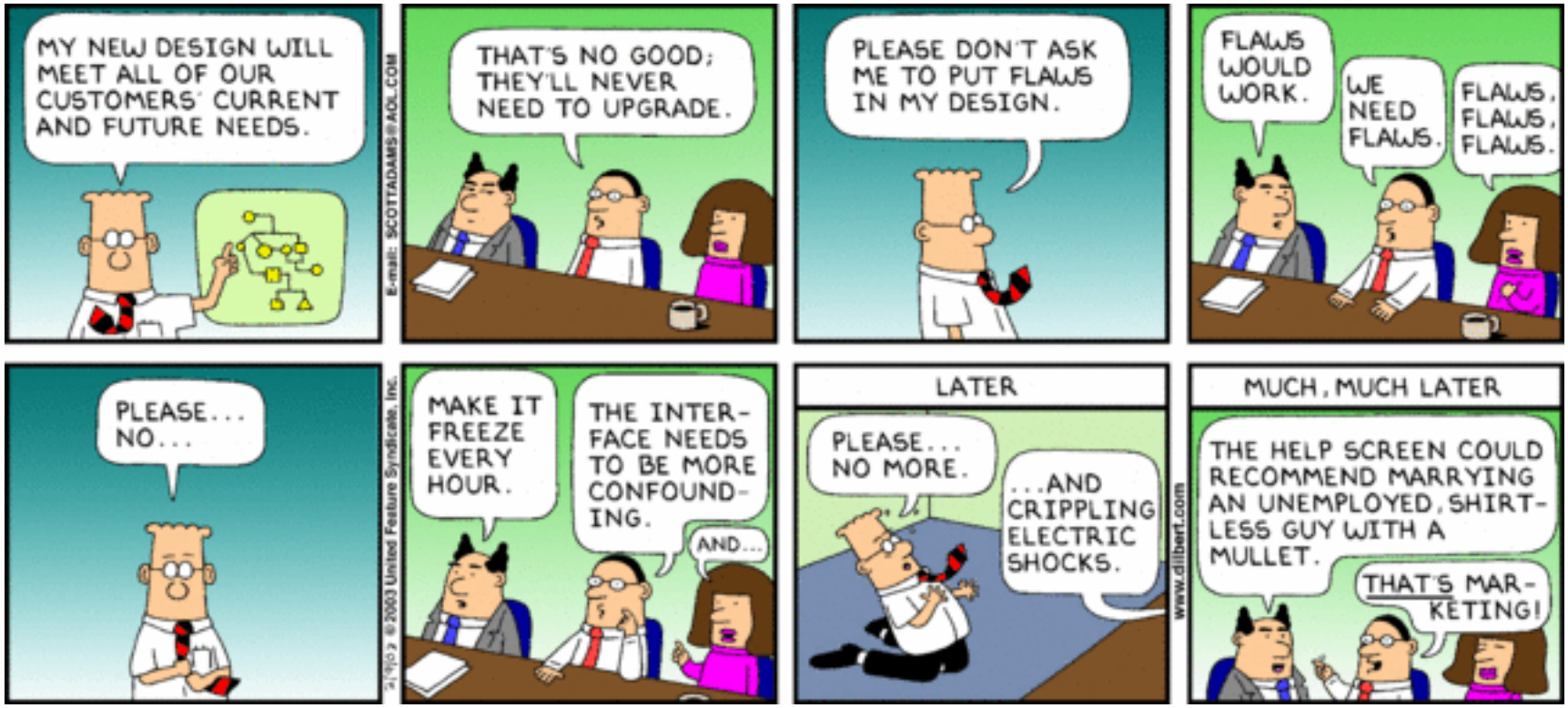
Next steps

- Identify features for release
  - Start with loading data, showing data, filtering
  - Create plan for Phase 2
- Update design as you go
- Keep track of problems to be avoided in final phase
- If running out of time
  - Focus on things that create “value” for the customer
  - Who are your important stakeholders, what are their concerns?
  - Depends on many factors (e.g., type of project, product, customer)


I'm finishing up the estimations, how much time do you need for the feature we discussed?
Wait, let me think about it for a second...
DODGE
5h
8h
10h
5 min
OK
1h
5h
2h
1h
TOO
2H
12h
WAC

Outlook to Phase 3 – some deliverables

- Complete and final design document
  - See Phase 1 and Phase 2 + two to three key lessons learned
- Product
  - See Phase 2
  - User manual
- Reflections and logs
  - See previous phases
- Demo


MY NEW DESIGN WILL MEET ALL OF OUR CUSTOMERS' CURRENT AND FUTURE NEEDS.
THAT'S NO GOOD; THEY'LL NEVER NEED TO UPGRADE.
PLEASE DON'T ASK ME TO PUT FLAWS IN MY DESIGN.
FLAWS WOULD WORK.
WE NEED FLAWS.
FLAWS, FLAWS, FLAWS.
PLEASE... NO...
MAKE IT FREEZE EVERY HOUR.
THE INTER-FACE NEEDS TO BE MORE CONFOUND-ING.
AND...
LATER
PLEASE... NO MORE.
...AND CRIPPLING ELECTRIC SHOCKS.
MUCH, MUCH LATER
THE HELP SCREEN COULD RECOMMEND MARRYING AN UNEMPLOYED, SHIRT-LESS GUY WITH A MULLET.
THAT'S MAR-KETING!
E-mail: SCOTTADAMS@AOL.COM
© 2003 United Feature Syndicate, Inc.
www.dilbert.com