IOWA STATE UNIVERSITY

Deadly Sins

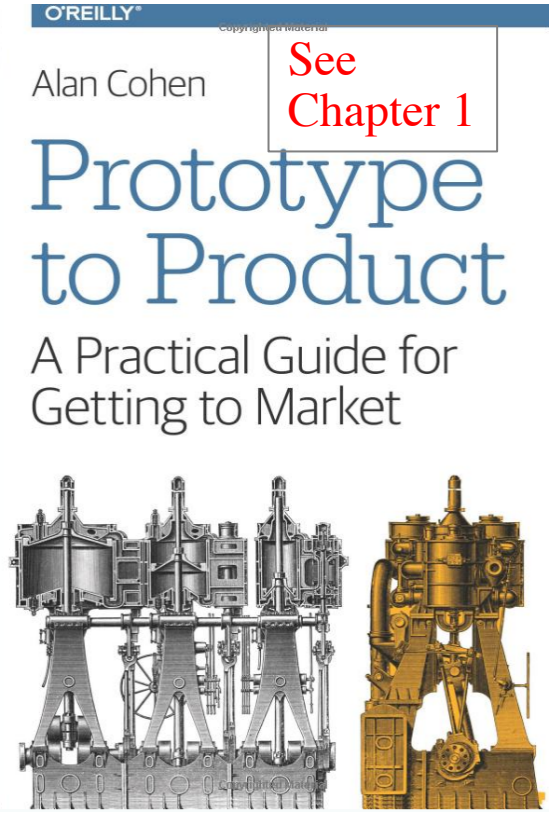
Lotfi ben Othmane

Course Outcomes

1. Understand project management processes
2. Develop project schedule
3. Identify risk and possible solutions
4. Manage human resources
5. Understand how to manage quality in software projects
6. Plan for communication
7. Control project progress
8. Understand intellectual property

What is “deadly sins”?

The “deadly sins” are mistakes that most often doom a project



Inspiration vs. Perspiration

- In 2 words: Why a project can fail although the idea is excellent?

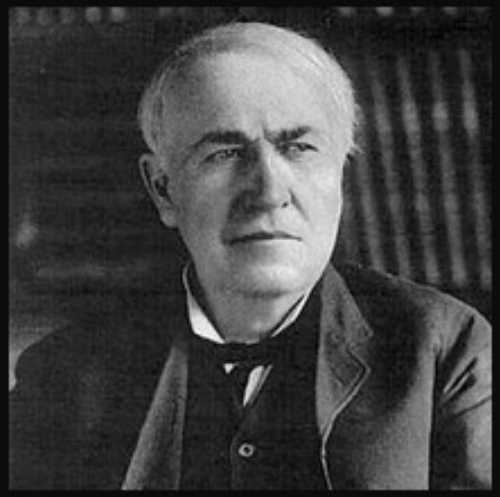
Inspiration vs. Perspiration

Inspiration - A product of creative thinking and work

Perspiration - Work that benefits from insight and cleverness but is about not screwing things up

- "Dotting the i's and crossing the t's."

Inspiration vs. Perspiration



Genius is 1 percent inspiration,
99 percent perspiration.

~ Thomas A. Edison

AZ QUOTES

Inspiration vs. Perspiration

"Most product development failures are of good product concepts that fail during the process of turning the concepts into a product"

Cohen 2015

- Most products fail
  - Not because the idea is bad
  - But the execution is wrong

DS 1: Putting off “serious” testing

- What is the impact of delaying integration tests? (We assume that unit tests are done well)

1. No problem since the architecture is validated
2. The software components may fail to work
3. Some quality requirements cannot be satisfied
4. Customers are not satisfied

DS 1: Putting off “serious” testing

- Delaying testing delays discovering what you don’t know
  - What if the specification from the client are wrong?
  - What if the architecture assumptions turn out to be not valid
- Solution: Demo prototypes to the client
  - Identify new “unknowns” and repeat

DS 1: Putting off “serious” testing

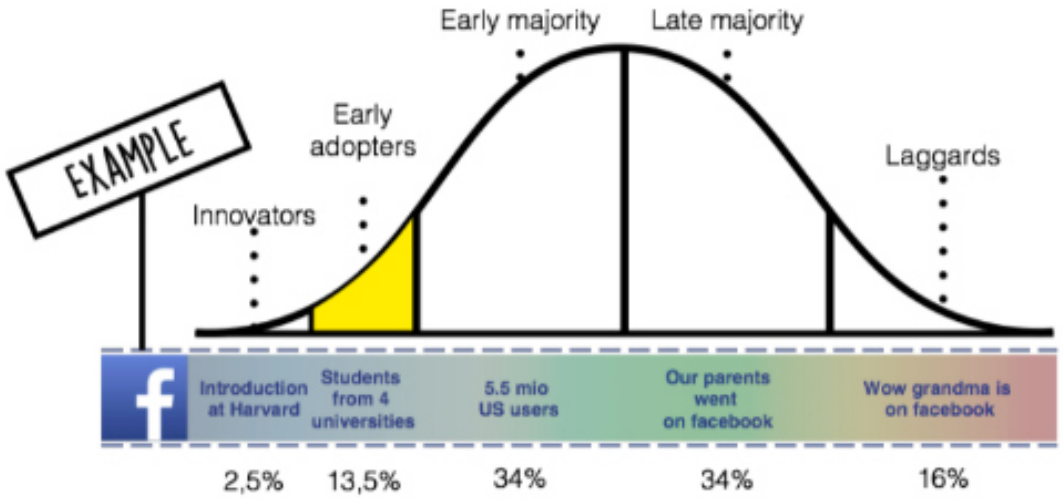
DS 1: Putting off “serious” testing

- Product development is largely an exercise in uncovering surprises as soon as possible.

- Surprises get more expensive to fix if discovered later.
  - Happy surprises are rare
  - Real world surprises lead to changes (and cost)

DS 2: Assuming what the client wants

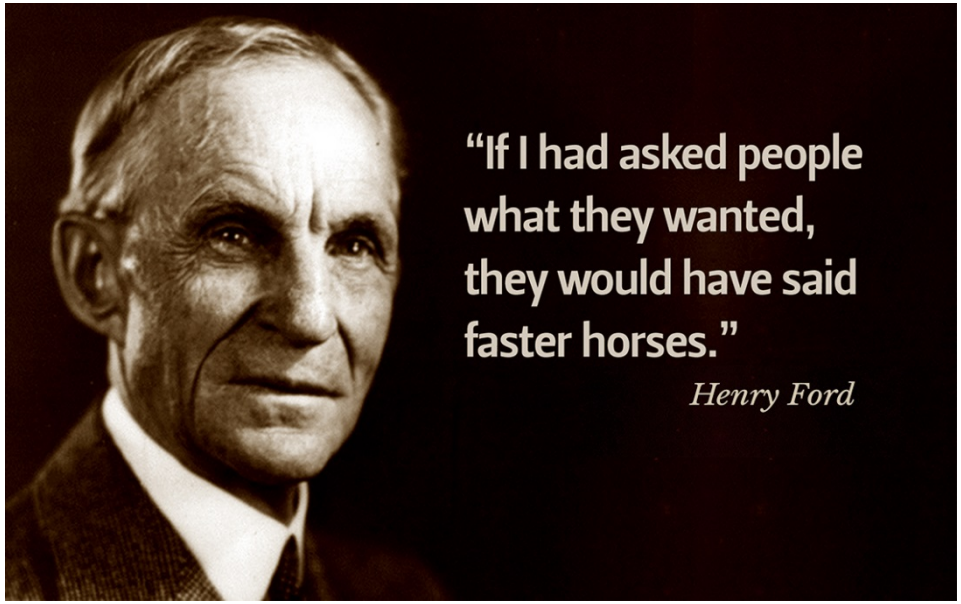
- Several products are often developed with assumptions about the client needs
  - E.g., Digg.com of Xerox
- Products are tested in the market place
  - USA has ~330 million people
  - 3.1 million software engineers



DS 2: Assuming what the client wants

A lot of times, people don't know what they want until you show it to them.

Why did Steve jobs or Henry ford succeed if this is a sin?

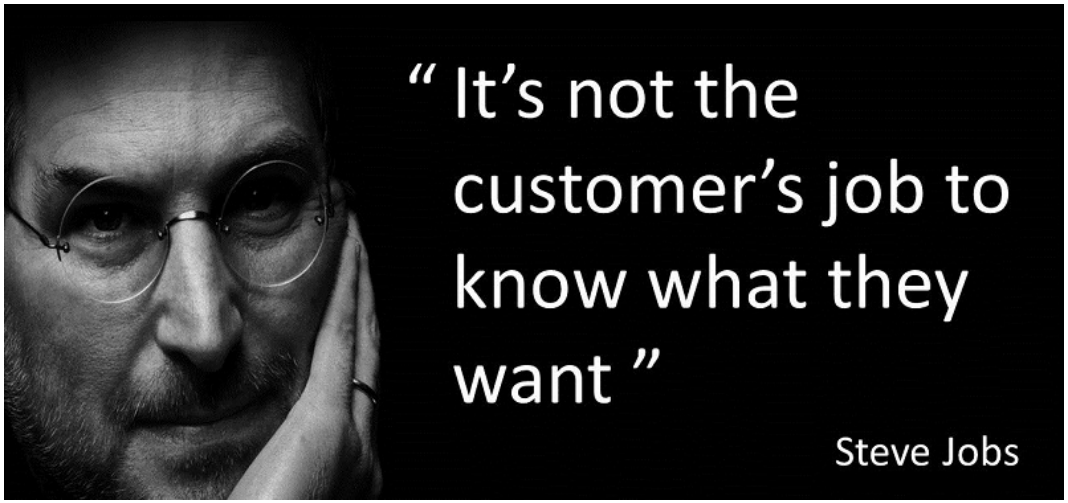


DS 3: Assuming that the users know what they want in a product

How to ensure that the users know what they want?

1. Sign a contract
2. See similar products
3. See project demos
4. I have a better idea

DS 3: Assuming that the users know what they want in a product



- "If you ask customers what they want, you'll get an answer. That answer might be right, or it might be wrong, and you'd better find that out."
- Client imagine their future needs – which is uncertain

DS 4: Lack of comprehensive requirements

Without requirements there can not be a common vision

Using the requirements below, which computer should be build?

1. Runs my programs
2. Has a CPU/memory
3. Can connect to the network

The Guardian.com

DS 4: Lack of comprehensive requirements

- Capture all important “things” to client
  - Compare to what’s practical
  - Negotiate and inform as needed

- Avoid vague requirements like
  - “attractive” and “world class”

  They are interpreted differently by different people

DS 5: Lack of a good project plan

Project plans will be wrong, quickly

Why do we plan then?

DS 5: Lack of a good project plan

- Engineers “dislike” them
- Their point is to
  - Create tasks list
  - Estimate resources
  - Understand dependencies
  - Argue for additional staff
  - Identify as many of the unknowns as possible
- Update with progress

DS 6: Not assigning responsibilities

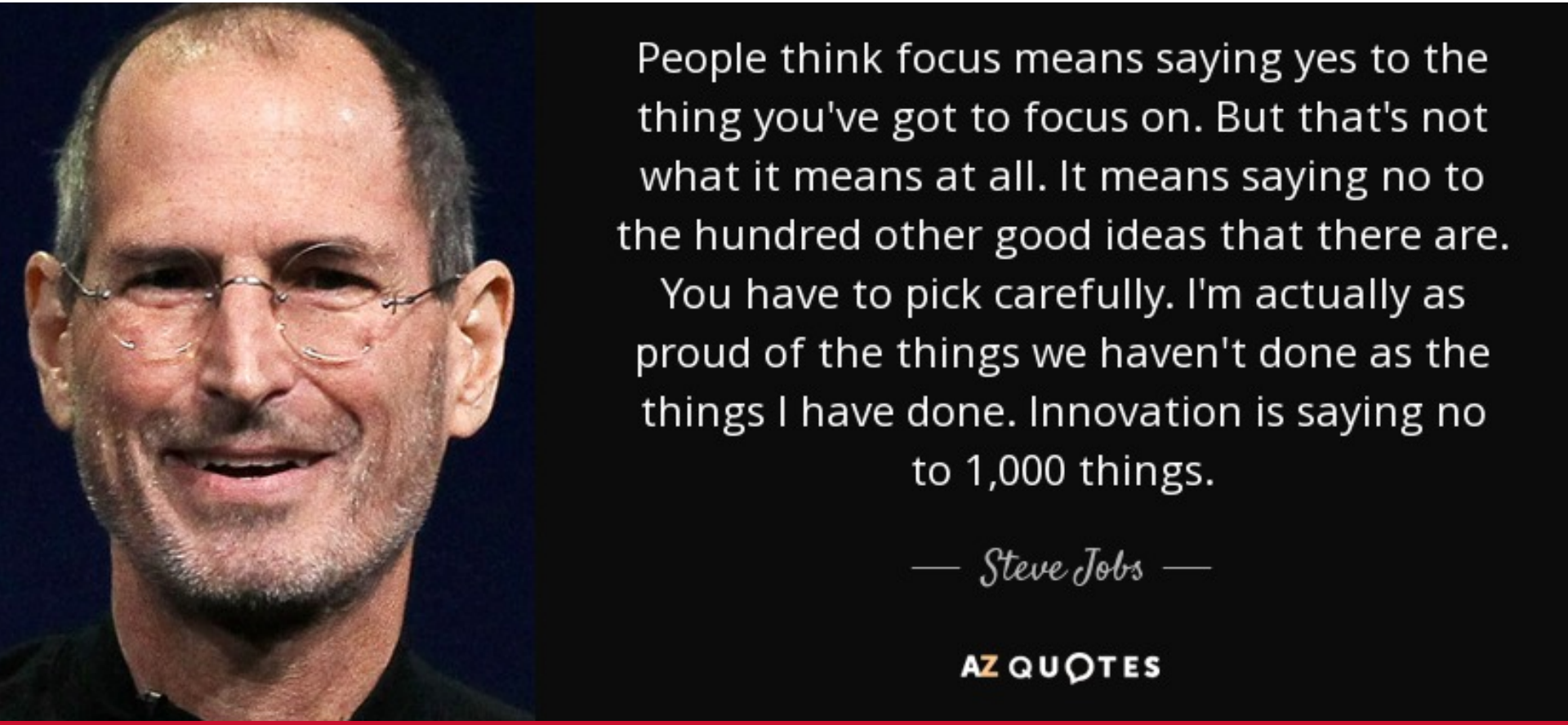
- We need to ensure that each task will be done when the start date comes

- Every task needs a champion
  - Not necessarily the implementer
  - Someone to stay on top of the progress & challenges
  - Monitor prerequisites and inform dependents

DS 7: Not addressing regulations

- Selling some of the products may require certification
  - Federal Communication Commission (FCC)
  - Payment Card Industry Data Security Standard
  - Export regulation – e.g., security
  - Domain-based standards: ISO 26262 for automotive
- Find an experienced expert to coordinate the compliance
- Case of Vehicle status and compliance with MISRA

DS 8: Sin of new-feature-itis



DS 8: Sin of new-feature-itis

What is the direct impact of adding new features later in the development?

1. impact the project plan
2. break existing features
3. improve customer satisfaction
4. require changes to the architecture
5. increase test effort

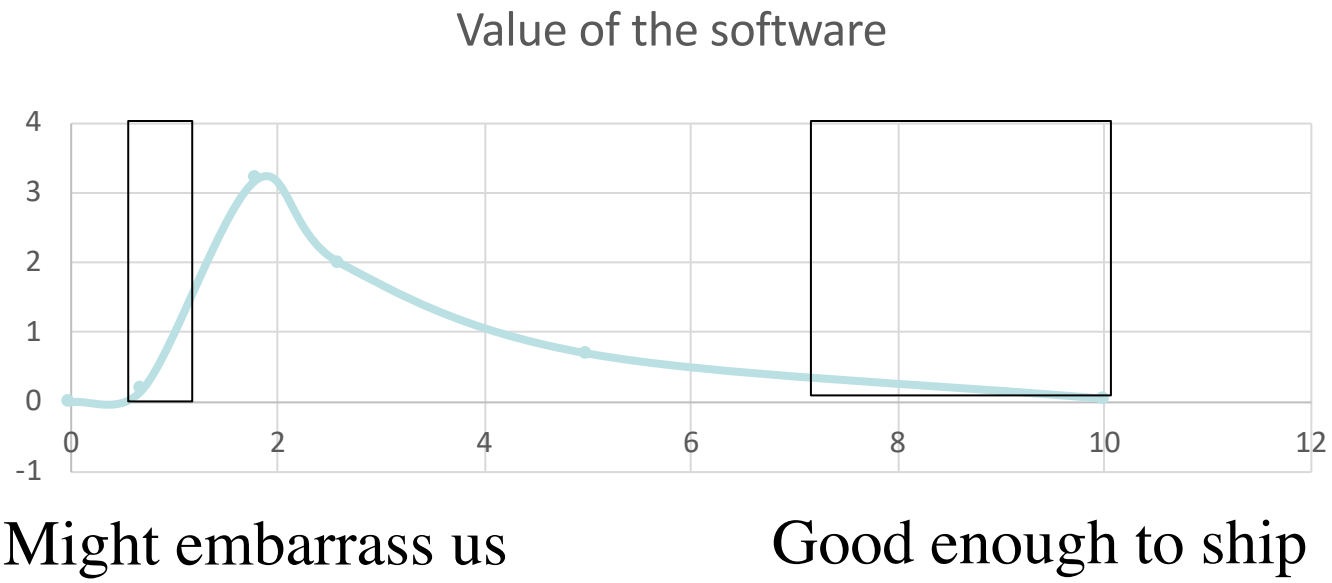
DS 8: Sin of new-feature-itis

- Late features impact the time and budget estimation, risk identification, etc.

- Product - correctness is more important than new features

DS 9: Not knowing when to quit polishing

- The more we test and improve a software the better it becomes.
- After some point: ship the product and “patch it later”



DS 9: Not knowing when to quit polishing

- What are the impacts of delaying quitting polishing?
  1. Customers are deprived from the benefits of the software
  2. Competitors may get early “technology” adaptors
  3. Delivering software with no bugs
  4. Gain in the budget

DS 10: Not planning to fail

- Recall: Project plans will be wrong, quickly

➔ Mitigation: padding with 25%-30% ----- 100%-200%

Did you plan 100% extra “for I do not know ”?

➔ Plan for what can go wrong to justify the extra.

DS 11: Developing technology instead of products

- The goal of each company is to develop product for their customers

- You will need to develop technologies for your own use

  ➔ Your own customers will not pay you for that

- Take the case of Microsoft and Apple
  - Windows is based on QDOS
  - MacOS and iOS are based on FreeBSD

“Deadly Sins” Summary

- Golden Rule of Product Development:

  “Product development is largely an exercise in uncovering surprises as soon as possible.”

1.	Putting off “serious” testing	Vice of laziness
2.	Assuming what the client wants	Vice of assumption
3.	Assuming the client knows what they need	Vice of assumption
4.	Lack of requirements	Vice of fuzziness
5.	Lack of a good project plan	Vice of fuzziness
6.	Not assigning responsibility	Vice of fuzziness
7.	Not addressing regulations	Vice of cluelessness
8.	The Sin of New-Feature-It is	Vice of perfectionism
9.	Not knowing when to quit polishing	Vice of perfectionism
10.	Not planning to fail	Vice of hubris
11.	Developing tech instead of products	Vice of ego