Software Engineering Project

Morgan Ericsson

morgan@cse.gu.se

@morganericsson

morganericsson

UNIVERSITY OF GOTHENBURG

CHALMERS

"Working software over comprehensive documentation."

Document What?

1. Requirements - Statements that identify attributes, capabilities, characteristics, or qualities of a system. This is the foundation for what shall be or has been implemented.

2. Architecture/Design - Overview of software. Includes relations to an environment and construction principles to be used in design of software components.

3. Technical - Documentation of code, algorithms, interfaces, and APIs.

4. End user - Manuals for the end-user, system administrators and support staff.

(according to Wikipedia...)

Issues

- Software development versus documentation development
- Software developers have the knowledge, technical writers have the skill
- Project-level versus enterprise-level documentation
- Quantity versus quality
- ...

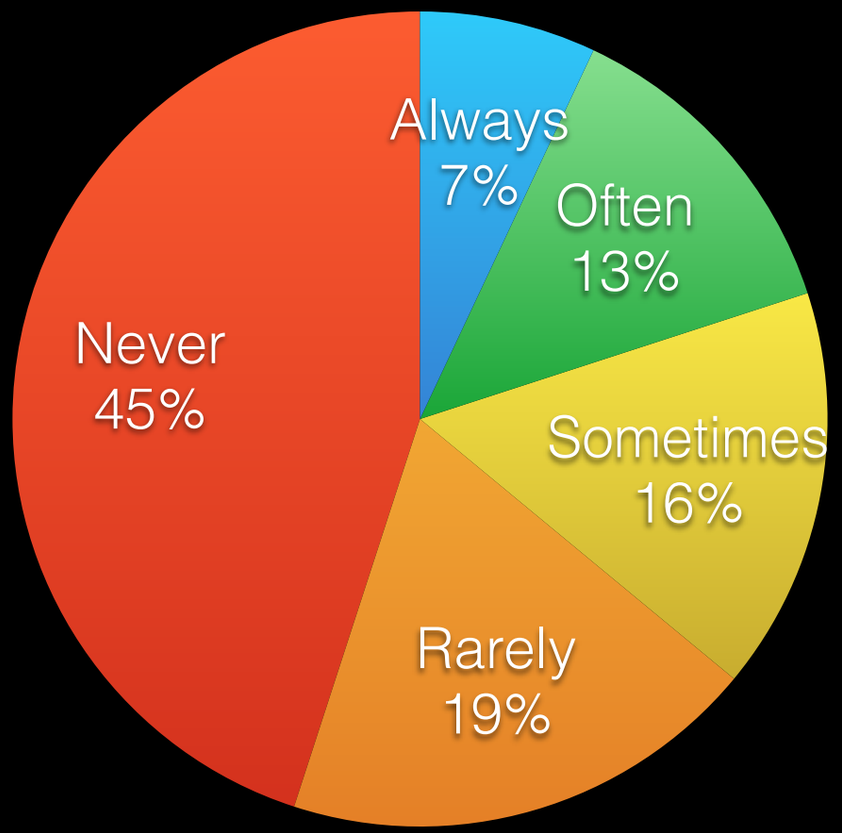
Why?

- Your project stakeholders require it
- To define a contract model
- To support communication with an external group
- To support organizational memory
- For audit purposes
- To think something through

Why? (Bad Reasons)

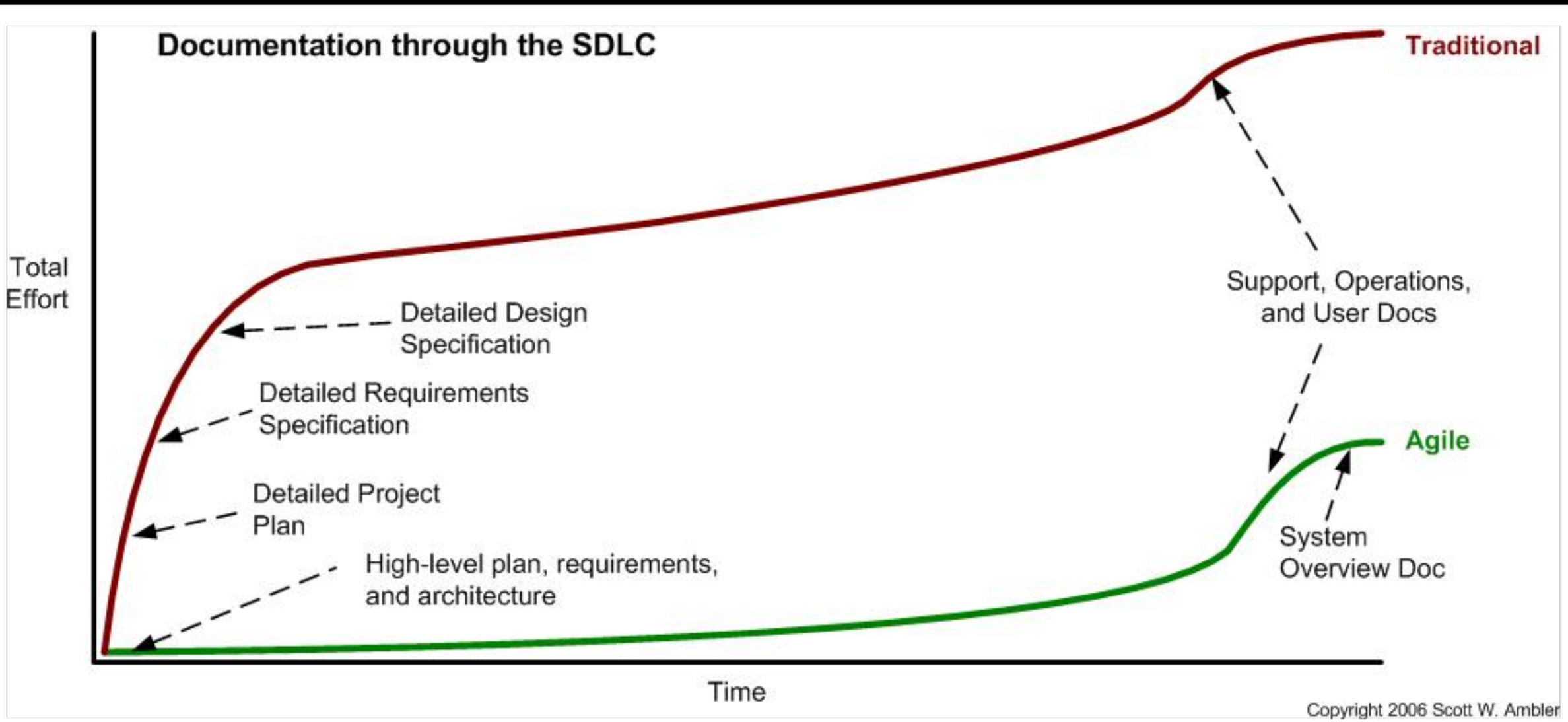
- The requester wants to be seen to be in control
- The requester mistakenly thinks that documentation has something to do with project success (*)
- The requester wants to justify their existence
- The requester doesn't know any better
- Your process says to create the document
- Someone wants reassurance that everything is okay

Really? (*)



Average percentage of delivered functionality actually used when a serial approach is taken on a “successful” IT project

Really? (*)

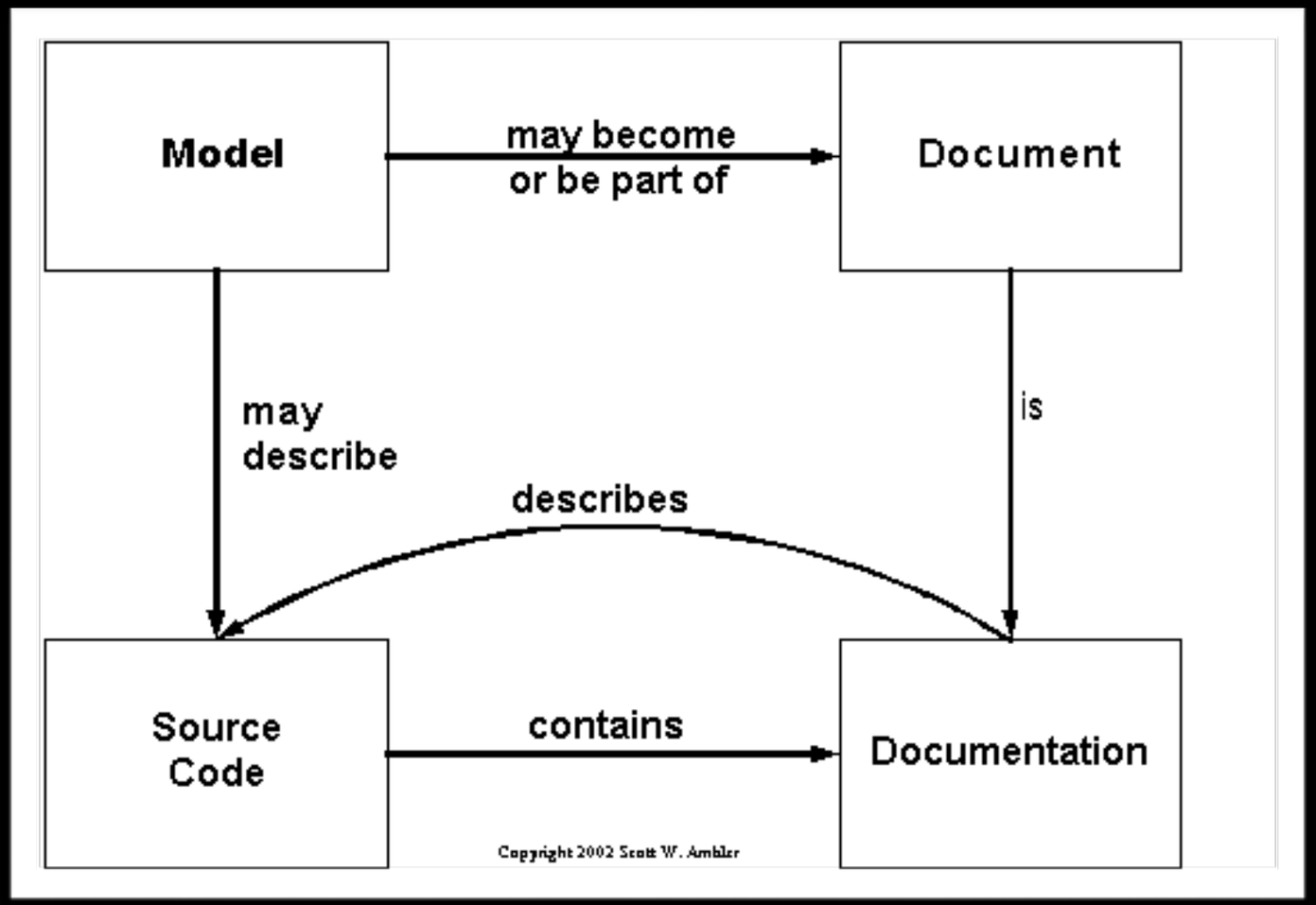

Documentation through the SDLC
Traditional
Total Effort
Detailed Design Specification
Detailed Requirements Specification
Detailed Project Plan
High-level plan, requirements, and architecture
Support, Operations, and User Docs
Agile
System Overview Doc
Time
Copyright 2006 Scott W. Ambler

But...

- You should understand the Total Cost of Ownership (TCO) for a document, and someone must explicitly choose to make that investment.
- The fundamental issue is communication, not documentation.

"Agilists write documentation if that's the best way to achieve the relevant goals, but there often proves to be better ways to achieve those goals than writing static documentation."

Relationships



So...

- Prefer executable work products such as customer tests and developer tests over static work products such as Plain Old Documentation (POD)
- With high quality source code and a test suite to back it up you need a lot less system documentation.

So...

- Models are not necessarily documents, and documents are not necessarily models
- Developers rarely trust the documentation, particularly detailed documentation because it's usually out of sync with the code

When/What

- Ask whether you NEED the documentation, not whether you want it
- Document stable things, not speculative things
- Create documentation only when you need it at the appropriate point in the life cycle

When/What

- Design decisions
- Vision Statement
- Operations documentation
- Project overview
- Requirements document
- Support documentation
- System Documentation
- User documentation

CRUFT

- Badness = 100% - CRUFT
  - C = The percentage of the document that is currently “correct".
  - R = The chance that the document will be read by the intended audience.
  - U = The percentage of the document that is actually understood by the intended audience.
  - F = The chance that the material contained in document will be followed.
  - T = The chance that the document will be trusted.