README

vs. IEEE, RUP, SWEBOK, CMMI

Yegor Bugayenko

Lecture #1 out of 16
80 minutes

The slidedeck was presented by the author in this YouTube Video

Viewpoint | Peter Freeman and David Hart

A Science of Design for Software-Intensive Systems

Computer science and engineering needs an intellectually rigorous, analytical, teachable design process to ensure development of systems we all can live with.

More than 200 years after George Washington was elected the first U.S. president, the November 2004 presidential election will usher in a new experiment in democracy—the first widespread use of electronic voting technology in a national election. Whatever your attitude about e-voting systems and whatever you may have read about them in recent months, they are an example of the software-intensive systems that every industrialized society increasingly depends on for their basic operations. Software-intensive systems help protect and manage commercial air travel, operate the electric power grid, file tax returns, and may one day even uphold our democratic foundations.

Yet today, in spite of more than three decades of effort, we still lack what could be called a solid scientific basis for building any kind of large-scale system that includes software elements. Our software-intensive systems are so complex we often find ourselves struggling to understand and control them. New systems are created and adopted with high hopes, crossed fingers, and even controversy, as in the case of e-voting. Security breaches, privacy violations, and emergent (unexpected) properties are some of the regular, often unwelcome, results.

For computer scientists, being told that creating software-intensive systems is difficult is not news. Even attendees at the 1968 NATO Software Engineering Conference in Garmisch, Germany, recognized the need to build, for example, a foundation for the systematic creation of software-intensive systems [3]. Efforts have continued since then to develop a scientific basis and process for creating them.

Living in a world in which the number and diversity of devices, amount of software, and degree of connectivity in complex systems are all increasing by orders of magnitude, it is essential that we have a science of design on which to base our efforts to create these systems. Changing this situation won't happen overnight, but we also can't wait another three decades to begin in earnest.

While it's easy to assert the need for a science of design, defining what such a science entails and understanding what design really means are much greater challenges.

By way of analogy, the design of a large building is based on many scientifically discovered and validated facts, along with volumes of codified experience. A similar scientific basis exists for many engineered artifacts, ranging from integrated circuits to airplanes. Professional disciplines like architecture and engineering have been working on issues of design for far longer than a few decades. In general, they have a stronger scientific footing, and computer scientists can learn where that footing overlaps with computing. (The National Science Foundation has funded at least one ongoing project to research how information technology might better assist the architectural design process.)

On the other hand, software-intensive systems are complex, being composed of software, people, computers, and other devices. Though the other components' connections to the software and their role in the overall design of the system are critical, the core consideration for a software-intensive system is the software itself, and other approaches to systematizing design have yet to solve the "software problem"—which won't be solved until software

COMMUNICATIONS OF THE ACM August 2004/Vol. 47, No. 8 19

"Design encompasses all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems—not just the activity following requirements specification and before programming, as it might be translated from a stylized software engineering process."

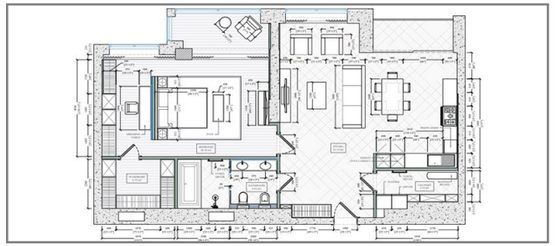
— Peter Freeman and David Hart. A Science of Design for Software-Intensive Systems. *Communications of the ACM*, 47(8):19–21, 2004. doi:10.1145/1012037.1012054

- Software vs. Interiors
- SDD at IEEE 1016
- SAD at RUP
- TS at CMMI
- SWEBOK
- README
- Books, Venues, Call-to-Action

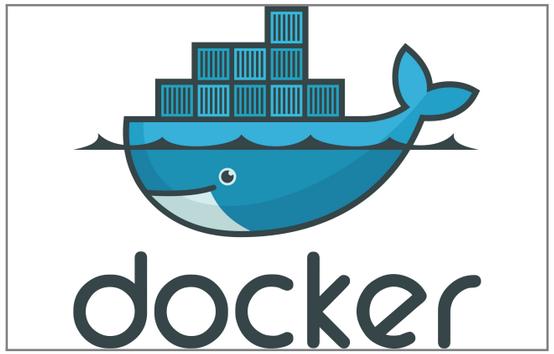
Chapter #1:

Software vs. Interiors

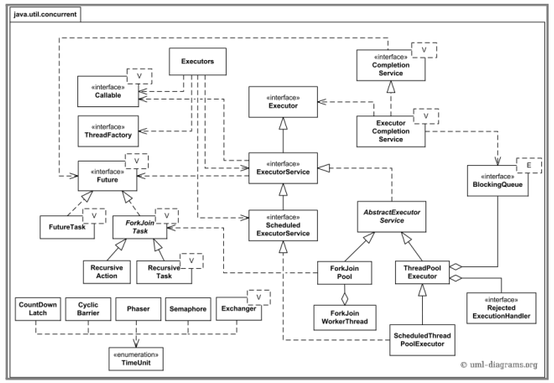
Interior



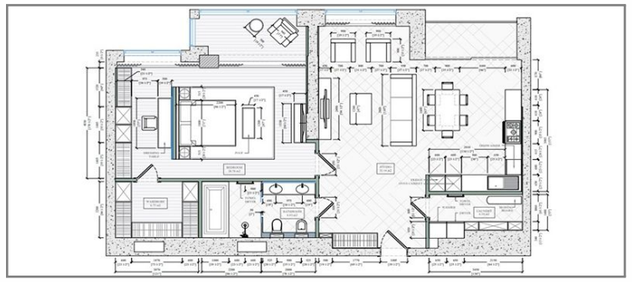
Interior Design



Software



Software Design



How to explain it?

Standards

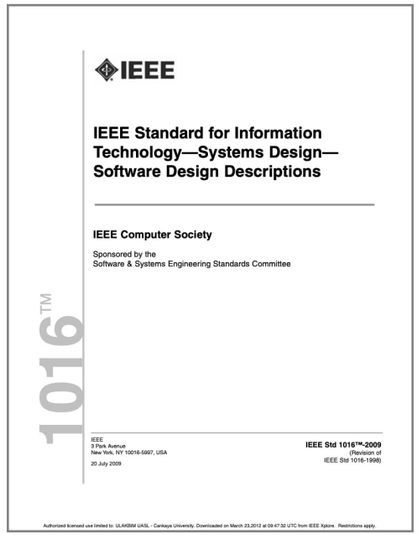
How to design?

Patterns

> 1. A good documentation
> is a precursor to a good design.

Chapter #2:

SDD at IEEE 1016



“An SDD is a representation of a software design to be used for recording design information and communicating that design information to key design stakeholders. This standard is intended for use in design situations in which an explicit SDD is to be prepared.”

— IEEE Standards Board. IEEE Std 1016-2009: Standard for Information Technology — Systems Design — Software Design Descriptions, 2009

Inactive-Reserved on March 2020

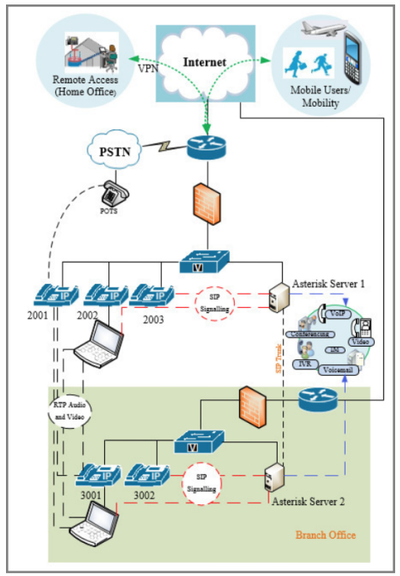
Glossary

A request is data package sent from a client to a server.
A client is a computer with a web browser.
A server is a computer with a software installed.

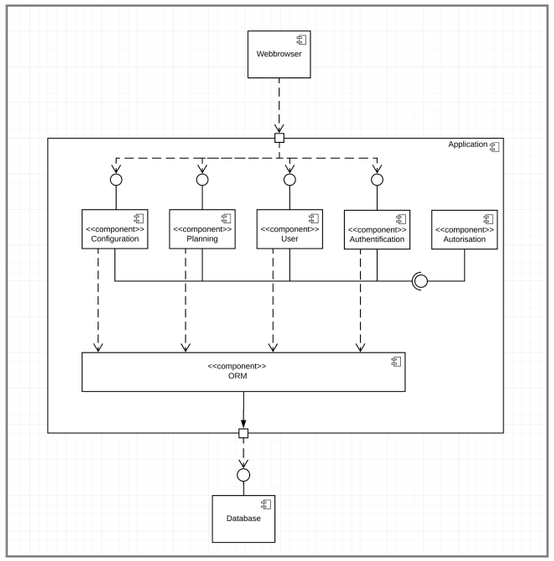
> 2. If I don't understand you, it's your fault!

Languages

NOT like this:

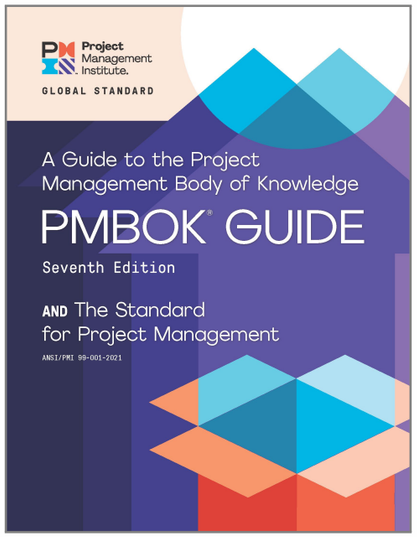


But like this:



UML + visual-paradigm.com

Stakeholders



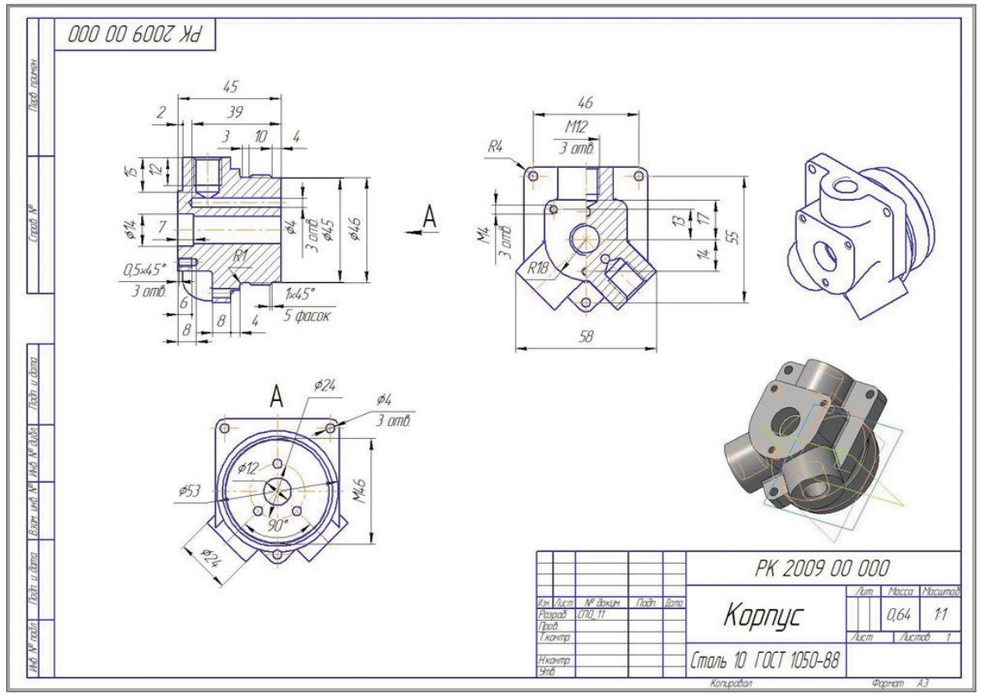
“Identify Stakeholders is the process of identifying the people, groups, or organizations that could impact or be impacted by a decision, activity, or outcome of the project.”

— Project Management Institute. *Guide to the Project Management Body of Knowledge (PMBOK® Guide) and the Standard for Project Management.* Project Management Institute, 2021

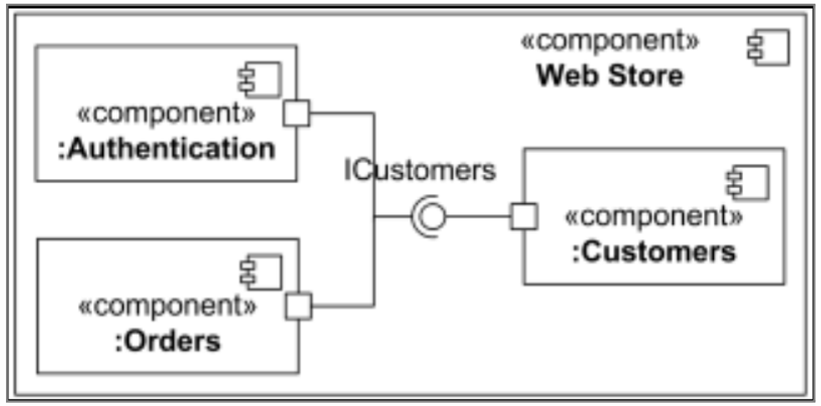
Concerns

Functional
and
Non-Functional Requirements

Viewpoints


РК 2009 00 000
45
2
39
3
10
4
15
12
ø4
7
R1
0,5x45°
3 отв.
6
8
8
4
ø4
3 отв.
ø45
ø46
1x45°
5 фасок
A
46
M12
3 отв.
R4
M4
3 отв.
R18
13
17
14
55
58
A
ø24
ø4
3 отв.
ø12
ø53
M46
90°
ø24
РК 2009 00 000
Корпус
Сталь 10 ГОСТ 1050-88
Масса
Масштаб
0,64
1:1
Лист
Листов 1
Копировал
Формат А3

Elements


«component»
Web Store
«component»
:Authentication
ICustomers
«component»
:Customers
«component»
:Orders

Rationale

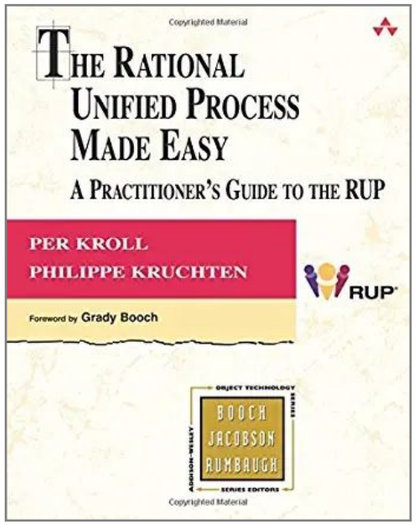
Why MongoDB, why not MySQL?

Multi-Criteria Decision Making (MCDM)
Architecture Trade-off Analysis Method (ATAM)
Decision Table
Multi Factor Analysis
Decision Matrix

> 3. Don't expect them to trust you, make them trust your decisions.

Chapter #3:

SAD at RUP

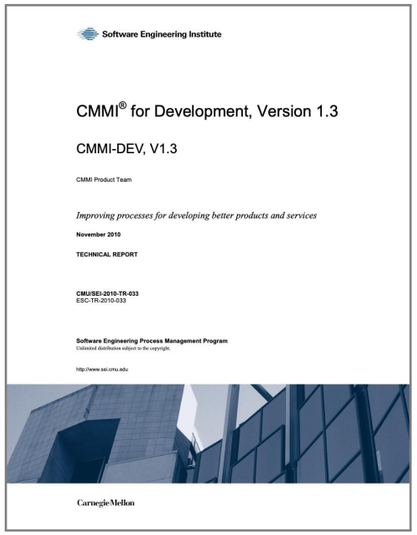


"The main responsibility of the architect is to describe the architecture of the system in a major artifact of the RUP product, called the Software Architecture Document (SAD). For many projects, this may be the only part of the design that is described in an actual document, as most design aspects can be documented in UML models and in the code itself."

— Per Kroll and Philippe Kruchten. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP.* Addison-Wesley, 2003

Chapter #4:

TS at CMMI

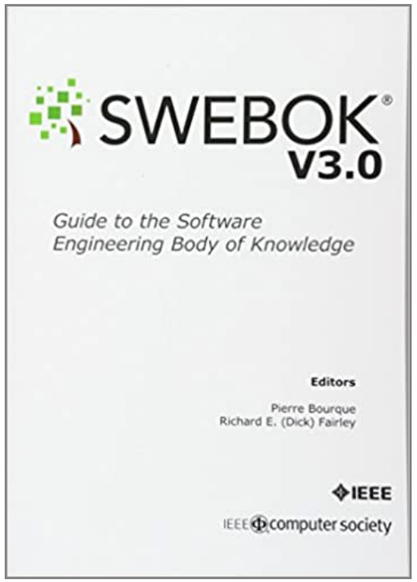


“Detailed design is focused on software product component development. The internal structure of product components is defined, data schemes are generated, algorithms are developed, and heuristics are established to provide product component capabilities that satisfy allocated requirements.”

— CMMI Product Team. CMMI for Development, Version 1.3, 2010

Chapter #5:

SWEBOK



“Viewed as a process, software design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software’s internal structure that will serve as the basis for its construction.”

— Pierre Bourque and Richard E. Fairley. *Guide to the Software Engineering Body of Knowledge (SWEBOK® Guide), Version 3.0.* IEEE, 2014. doi:10.5555/2616205

> 4. Stay away from MS Word, instead, use LaTeX with Git.

Chapter #6:

README

```
# Sample Markdown

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod.

## Text basics

This is *italic* and this is **bold**.  Another _italic_ and another __bold__.

> __Here is some quotation__. Lorem ipsum dolor sit amet, consectetur
> adipisicing elit, sed do eiusmod tempor incididunt ut labore et
> dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation.

## Links

This is an [example inline link](http://example.com/) and [another one with a
title](http://example.com/ "Hello, world"). And [another][someref] one.

## Code

This is inline code: `some code here`.

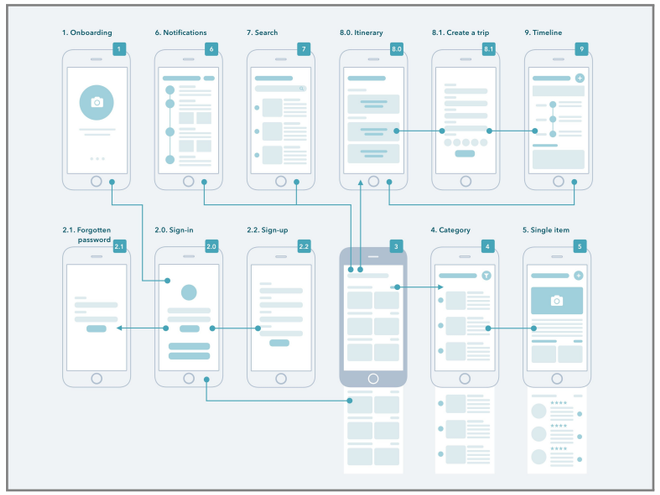
    <script>
        document.location = 'http://example.com/?q=markdown+cheat+sheet';
    </script>

```java
public class HelloWorld {
   public static void main(String[] args) {
       System.out.println("Hello, world!");
   }
}
```

[someref]: http://example.com "rich web apps"
[MarkdownREF]: http://daringfireball.net/projects/markdown/basics
[gfm]: http://github.github.com/github-flavored-markdown/
```

GitHub

Markdown
by John Gruber
since 2004



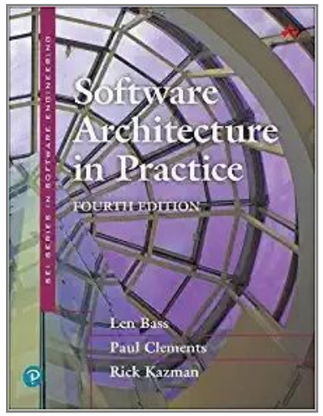
UI mockups

`moqups.com`, `balsamiq.com`, `sketch.com`, `dribbble.com`, etc.

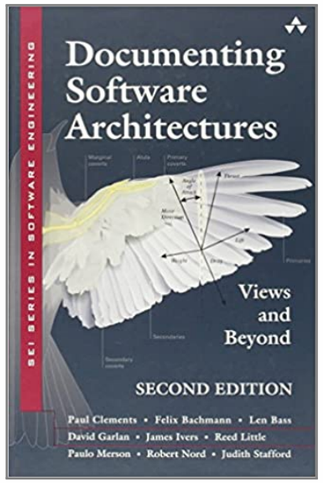
> 5. Brevity is a virtue, redundancy is a sin.

Chapter #7:

Books, Venues, Call-to-Action



Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Pearson Education, 4 edition, 2021



Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Paulo Merson, Reed Little, and Robert Nord. *Documenting Software Architectures: Views and Beyond.* Addison-Wesley, 2 edition, 2011

Where to publish:

IEEE International Conference on Software Architecture (ICSA)

Call to Action:

Create and explain the design of a QR-code generator app in the README.md file in a new GitHub repository. Sample: www.4qrcode.com

Still unresolved issues:

- How to synchronize an SDD with the source code?
- How to generate the code from an SDD?
- How to embed diagrams into the source code?
- How to validate source code vs. the SDD?

Bibliography

Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Pearson Education, 4 edition, 2021.

IEEE Standards Board. IEEE Std 1016-2009: Standard for Information Technology — Systems Design — Software Design Descriptions, 2009.

Pierre Bourque and Richard E. Fairley. *Guide to the Software Engineering Body of Knowledge (SWEBOK® Guide), Version 3.0*. IEEE, 2014. doi:10.5555/2616205.

Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Paulo Merson, Reed Little, and Robert Nord. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2 edition, 2011.

Peter Freeman and David Hart. A Science of Design for Software-Intensive Systems. *Communications of the ACM*, 47(8):19–21, 2004. doi:10.1145/1012037.1012054.

Project Management Institute. *Guide to the Project Management Body of Knowledge (PMBOK® Guide) and the Standard for Project Management*. Project Management Institute, 2021.

Per Kroll and Philippe Kruchten. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Addison-Wesley, 2003.

CMMI Product Team. CMMI for Development, Version 1.3, 2010.