Fachhochschule Nordwestschweiz
Hochschule für Technik

Software Architecture

Exercise – Software Architecture Definition

BSc

Ingo Arnold

Exercise Opening
Overview

There are many different answers to the question of **what exactly software architecture is** — wickedly put, as many as there are software architects.

One world-renowned institution that has long been involved in software architecture is the **Software Engineering Institute (SEI)** at Carnegie Mellon University in Pittsburgh, Pennsylvania.

Various such **software architecture definitions** have been collected and made available to the general public ([SEI Software Architecture 2021])

Exercise Agenda

- Software Architecture Definition

Software Architecture Definition
Exercise

There are many different answers to the question of **what exactly software architecture is** — wickedly put, as many as there are software architects.

One world-renowned institution that has long been involved in software architecture is the **Software Engineering Institute (SEI)** at Carnegie Mellon University in Pittsburgh, Pennsylvania.

Various such **software architecture definitions** have been collected and made available to the general public ([SEI Software Architecture 2021])

Software Architecture Definition
Exercise

However, the SEI also offers its own definition[1] (text 1):

"The software architecture of a program or computing system is a depiction of the system that aids in the understanding of how the system will behave. Software architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by design and implementation teams. The architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. Architecture is an artifact for early analysis to make sure that a design approach will yield an acceptable system. By building effective architecture, you can identify design risks and mitigate them early in the development process."

[1] http://www.sei.cmu.edu/architecture/

Software Architecture Definition
Exercise

Does this definition contradict common agile process attitude and models? Architecture (as defined by the SEI) sounds like a lot of *planning ahead*. Some thoughts worth reading on the role of architecture and architects in the article *who needs an architect* ([Fowler 2003]) in which Martin Fowler embeds postings by Ralph Johnson (text 2):

"In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called *architecture*. This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers."

Software Architecture Definition
Exercise

What are the relationships between architecture and code? What significance does the architecture still have when a system has been completely programmed? In this context, the abstract of a presentation at the Java User Group Switzerland (JUGS) by Simon Brown (software architecture vs. code[1]) is worth reading (text 3):

"Software architecture and coding are often seen as mutually exclusive disciplines, despite us referring to higher level abstractions when we talk about our software. You've probably heard others on your team talking about components, services and layers rather than objects when they're having discussions. Take a look at the codebase though. Can you clearly see these abstractions or does the code reflect some other structure? If so, why is there no clear mapping between the architecture and the code? Why do those architecture diagrams that you have on the wall say one thing whereas your code says another?"

[1] https://www.jug.ch/html/events/2015/software_architecture_vs_code.html

Software Architecture Definition
Exercise

Finally, another interesting angle via which Ralph Johnson marks an elementary difference between traditional architecture or engineering disciplines and software architecture in Martin Fowler's article ([Fowler 2003]) (text 4):

"Software is not limited by physics, like buildings are. It is limited by imagination, by design, by organization. In short, it is limited by properties of people, not by properties of the world. We have met the enemy, and he is us."

An angle Melvin Convey had already picked up in *how do committees invent?* ([Convey 1968]) (text 5):

"The basic thesis of this article is that organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations."

Software Architecture Definition
Exercise

What is the essence of texts 1, 2, 3, 4, and 5? Summarize the key messages of each text compactly.

Software Architecture Definition
Exercise

What is the essence of texts 1, 2, 3, 4, and 5? Summarize the key messages of each text compactly.

Text 1:

- Architecture contributes building blocks to software design.
- The realization of targeted quality attributes central to the building block design .
- The building block design supports the organization of the process to develop them.

Text 2:

- Individual contributions of all those involved in the building process must fit into the overall picture — this overall picture is the architecture.

Text 3:

- The structures of the architecture, its building blocks, and their relationships to each other must be discoverable in the code.
- In other words, code and architecture must be mappable to each other.

Software Architecture Definition
Exercise

What is the essence of texts 1, 2, 3, 4, and 5? Summarize the key messages of each text compactly.

Text 4 and Text 5:

- The basic building material of software systems is humankind itself — humankind as an individual as well as humankind in societal organization.
- The immediate building material of software is of linguistic nature — the intermediate one is determined and limited by the biological, cognitive, imaginative or communicative ability of all of us.

Software Architecture Definition
Exercise

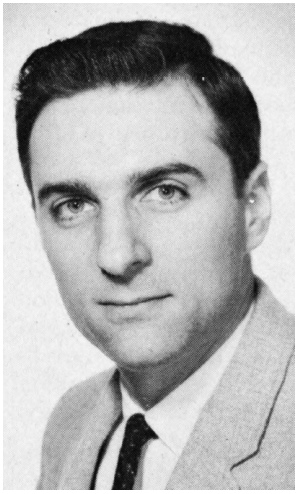
Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).

From **Conway's** publication "**How do committees invent?**" from 1968!

"**Any organization** that **designs** a **system** will inevitably **produce a design** whose structure is **a copy of the organization's communication structure**.

The very act of **organizing** a design **team means** that **certain design decisions have** already **been made**, explicitly or otherwise.

Given any design **team organization**, there is **a class of design alternatives** which **cannot be** effectively **pursued** by such an organization **because** the necessary **communication paths do not exist**."



Software Architecture Definition
Exercise

Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).

Christopher Alexander presents a compelling **analogy** in *"The Timeless Way of Building"*.

Alexander **compares** the **relationship** between the **architecture of a system** and the **architecture of the organization** responsible for **designing it** with the **relationship** between a **living flower** and the **seed from which it grew**.

"If you **want** to **make** a **living flower**, you do **not build it physically**, with tweezers, cell by cell. You **grow it from the seed**." [Alexander 1979]

Software Architecture Definition
Exercise

Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).

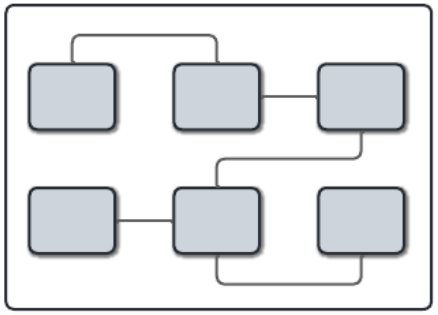
Organizations are **like seeds**, and the **technical systems** they **use** or **produce** are **like flowers**.

<<Social System>>
Organization

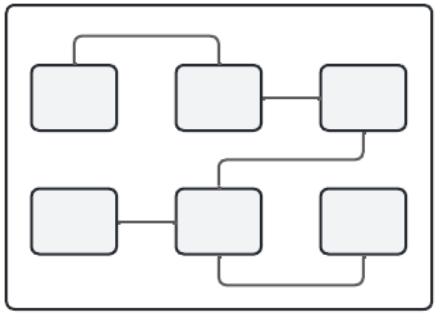
<<Technical System>>
IT Solution

Organizations replicate their inherent (communication) **designs** in the **technical systems** they **use** or **produce**.

This means that an **organization's** (communication) **design inevitably influences** and **significantly shapes** the **design** of their **technical systems**.



Organizational Replication



Communication Design of Social System

Component Design of Technical System

Software Architecture Definition
Exercise

EXAMPLE

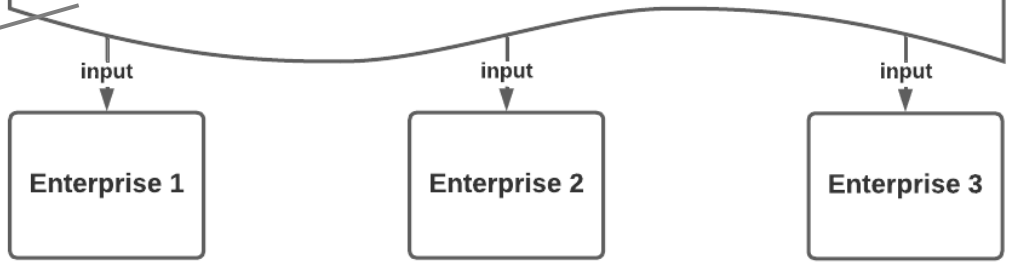
Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).

A **compiler** for a formal language is to be **developed by three companies**.

All three companies **receive** the **same specification** of the language (grammar + semantics).

A **simple grammar** for a **language** that **supports conditional repetition** and **conditional branching**.

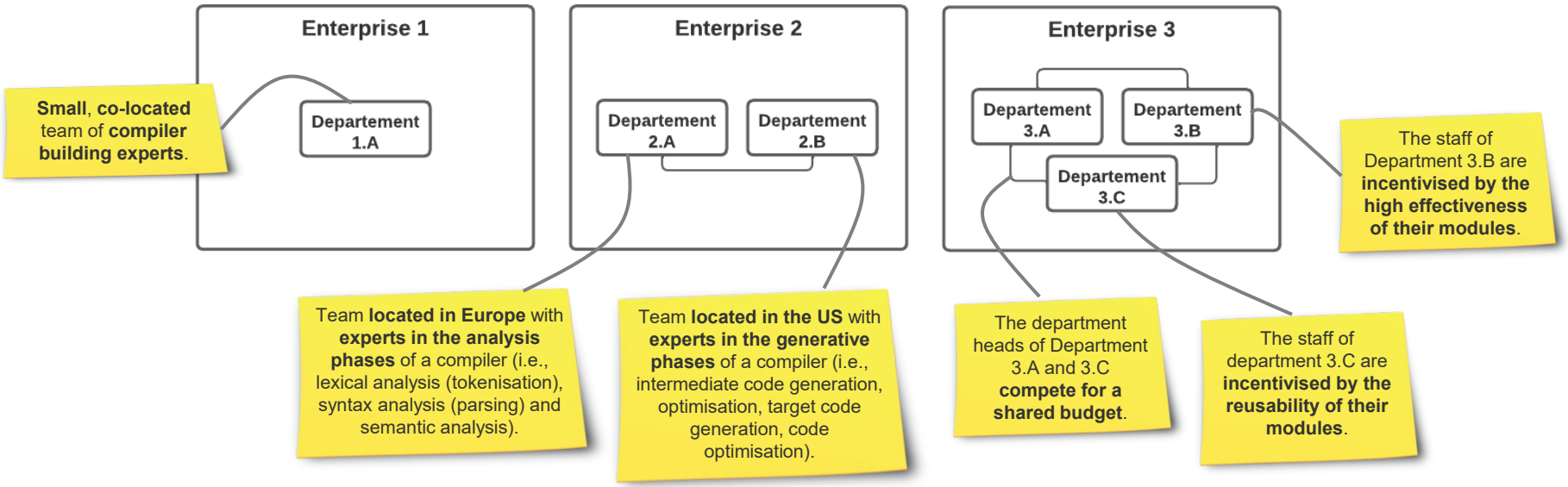
```
<program>       := <statement> | <statement> <program>
<statement>     := <assignment> | <conditional> | <loop>
<assignment>    := <identifier> "=" <expression>
<conditional>   := "if" <expression> "then" <program> "else" <program> "endif"
<loop>          := "while" <expression> "do" <program> "endwhile"
<expression>    := <identifier> | <constant> | <expression> <operator> <expression>
<identifier>    := [a-zA-Z]+
<constant>      := [0-9]+
<operator>      := "+" | "-" | "*" | "/"
```



Software Architecture Definition
Exercise

EXAMPLE

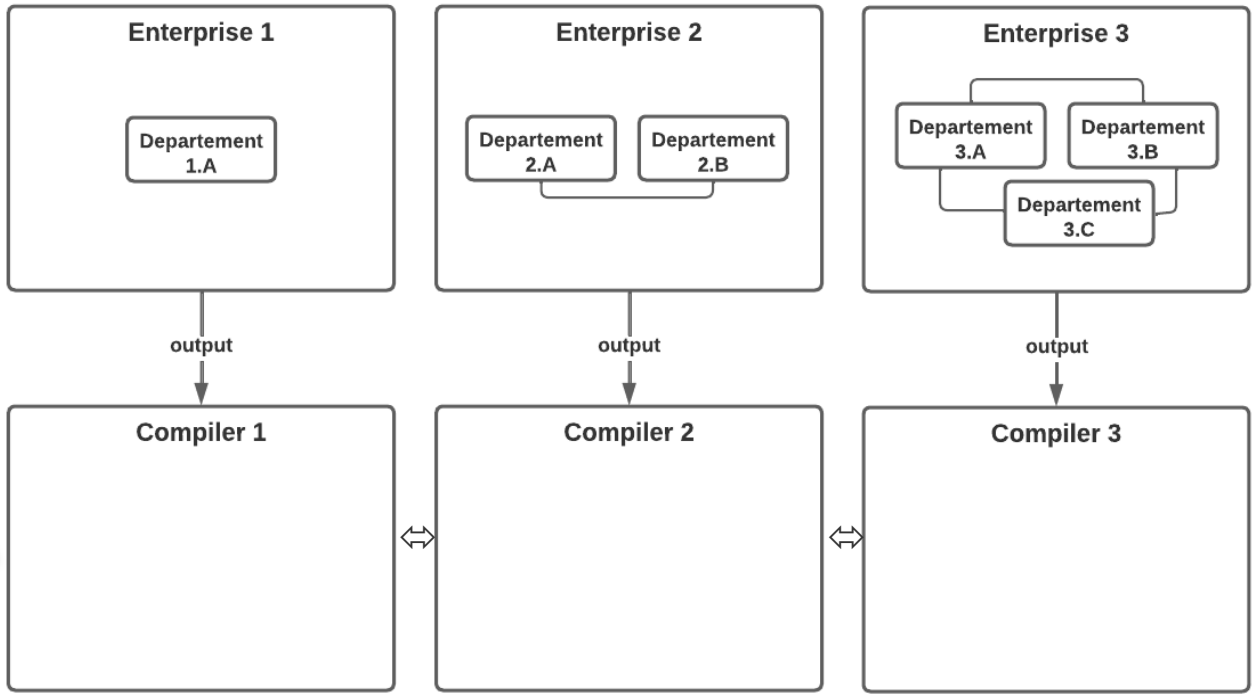
Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).



Software Architecture Definition
Exercise

EXAMPLE

Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).



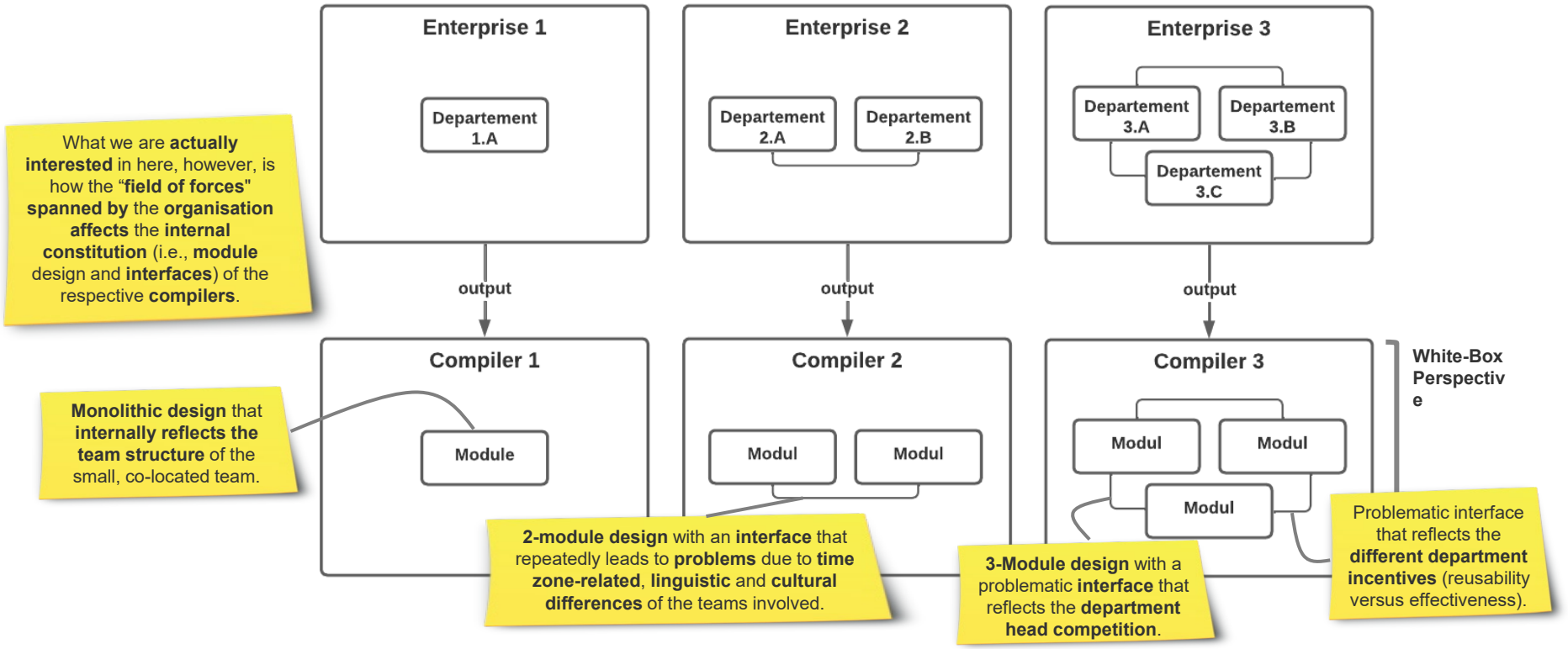
The compiler **systems realised by the three companies** are **functionally complete**, and **correct**.

Functionally (i.e., from a black-box perspective) they are **equivalent** to each other.

Software Architecture Definition
Exercise

EXAMPLE

Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).



Software Architecture Definition
Exercise

EXAMPLE

Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).

An **Enterprise comprises** the 3 **divisions**: Business Unit **A**, Business Unit **B**, and Business Unit **C**.

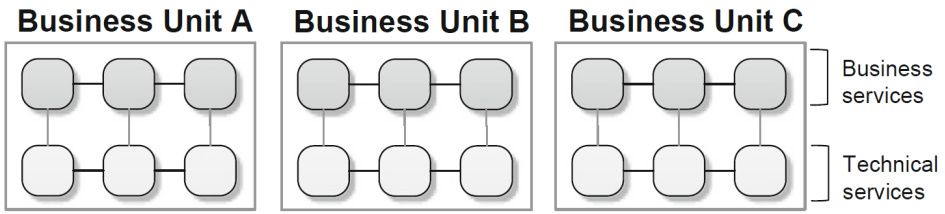
Business unit **A** has to cope with the **smallest budget**.

Business unit **B** is positioned **between A and C** in budget terms.

Business unit **C** is the unit with the **largest financial capacity**.

At the same time, **all markets** are **extremely demanding** and **dynamic**, requiring **specialized** and **regularly adjusted business capabilities**.

Their **business models** are **similar**, with **each operating** in its **own regional market**.



The **business capabilities** in all **business units** are **provided** by **business services**, while business services, in turn, are **based on technical services**.

The **business units** have relatively **large budgetary autonomy** to optimally **adapt** to their individual **markets**' conditions.

Software Architecture Definition
Exercise

EXAMPLE

Melvin Conway researched and described the **phenomenon** of **organizational replication** (i.e., the impact organizational forces have on Systems) already in 1968 ([Conway 1968]).

Assume no countermeasures are **taken to** influence or **mitigate organizational replication**.

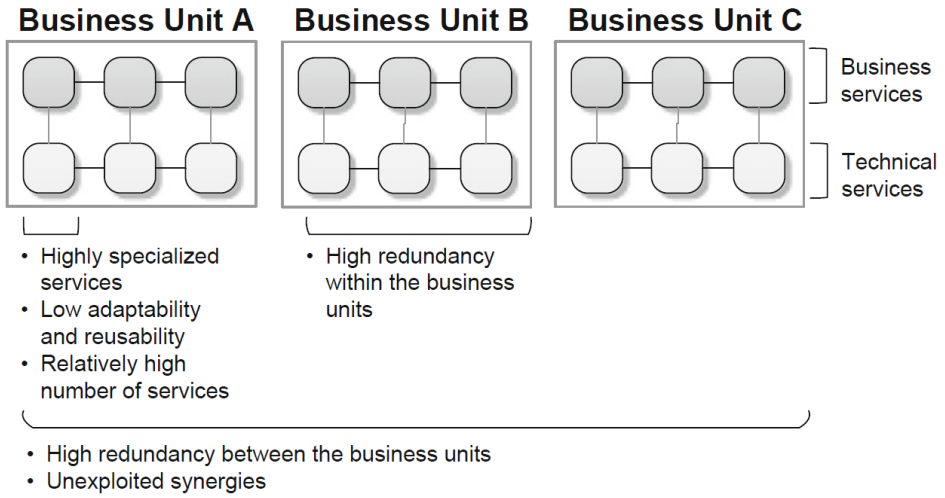
In this case the **organizational forces** described **above affect all business** and **technical services** – both **within individual business units** and **at the enterprise level**.

The **need** for **specialized business capabilities** likely **leads** to **correspondingly specialized business services**.

Specialized business services, in turn, **place high demands** on the **services upon** which they **operate**, requiring **highly specialized** and **individualized technical services** accordingly.

Demand for **highly specialized business** and **technical services** tends to **lead** to **more systems than** are otherwise **needed**.

This is **because generalized services cover** a **broader** spectrum of **requirements**, which is why, **in principle**, a **smaller number of generalized services** would **address** a **proportionally larger number of requirements**.



The **organizational forces** also lead to **less adaptable services**.

Specialized services are **optimized** in terms of their **specialization**, which **contrasts** to **generalization** and an associated need for **adaptability**.

High **service specialization coupled with** correspondingly **low adaptability leads** to increased **redundancy** both within business units and at the enterprise level.

The **increased redundancy** ultimately **causes** an **inefficiently high investment** and **operating cost** profile, which may be **acceptable** in the **short term**, but is **highly undesirable** in the **long term**.

Bibliography
Lecture

[SEI Software Architecture 2021]
Software Engineering Institute, *What is your definition of software architecture?*, https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=513807, 2021

[Fowler 2003]
Fowler, Martin, *Who needs an Architect?*, https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf, 2003

Questions