Developing an Educational Tool for Teaching Software Architecture

A.J. de Vries

BSc Thesis

August 4, 2021

Supervisor
Dr. A. Fehnker

Critical Observer
Dr. C.M. Epa Ranasinghe

Bachelor of Science Creative Technology
Faculty of Electrical Engineering, Mathematics and Computer Science at the University of Twente

Abstract

Developing skills and knowledge on the topic of software architecture is of value for more than just software architects. Creative Technology students for example should be able to explain their designs to stakeholders, including architectural designs. For these people a basic understanding of software architecture is sufficient.
The aim of this project was to develop an educational tool that teaches these basics. To make the tool easily accessible for individuals the decision was made to place the tool on the internet and let people use it on their own.
To develop the tool the Creative Technology design process, consisting of four phases, was applied. In the first phase, the ideation phase, a concept was chosen that was further specified in the specification phase. The realization phase was then used to build the actual tool. This tool consists of videos and exercises that engage the user in learning about software architecture. The last phase, the evaluation phase, showed that the tool could still use some improvement when it comes to the design. The usability and engagement of the tool however were found to be good already.
Overall the developed tool shows potential to be of great use for people who want to learn the basics of software architecture. Future work on this tool can focus on making the tool more engaging or extending the tool by addressing the soft skill element of software architecture.

Acknowledgement

First I want to thank my supervisor Dr. A. Fehnker who guided me in the process of this graduation project. His feedback and guidance helped me stay motivated, which was tough at times during a pandemic. Next I want to thank my critical observer Dr. C.M. Epa Ranasinghe who provided useful input during the brainstorming sessions we had. I want to thank both my supervisor and critical observer for allowing me to extend my graduation project into the summer. Without this extension I would not have been able to finish this project.
Up next I want to thank all the respondents to the survey I sent out. With their feedback I was able to evaluate and improve the developed tool. At last I want to thank all the people I have spoken with who shared their knowledge and experience when it comes to software architecture. It has been of great help in the understanding and ideation of this project.

Table of Contents

List of Figures and Tables

Figures

Tables

Chapter 1 - Introduction

The ability to design, understand and discuss the software architecture of a technical system is an important skill to build secure and maintainable products. There are specific jobs dedicated to this task. Besides these experts there are also people who need to have a basic understanding and the ability to discuss some architectural choices. This could be a developer, UX designer or the management for example.

The goal of this Graduation Project is to build a tool that teaches the basics of software architecture. A participant should become able to understand and discuss software architecture on a high abstraction level and therefore needs to know how to communicate about the architecture.

To reach this goal an online open educational tool will be developed. Users should be able to follow the course alone at any time and any location that has an internet connection. This means there is no specific audience but instead a tool that can be widely used by different types of people. Because of the easy accessibility it could also be used by teachers or companies to direct their students or employees to.

The contents of the tool should be engaging while maintaining effectiveness. This will be done by implementing visual elements. For instance the use of (interactive) video or images. Besides that the application of gamification should provide engaging content. To develop the end product this aspect will also be researched.

1.1 Definition of software architecture

The concept of software architecture is seen as a fuzzy concept. Early definitions use the terms components and connectors as the fundamental concepts. A more modern approach would be to define software architecture as a set of architectural design decisions that are hard to change. This is not always the case though. The book Software Architecture in Practice[1] uses the following definition, *“The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.”*. This definition states that software architecture consists of a set of structures. These structures are usually more focussed on the higher abstraction level of a

system. When talking about the lower level structures the term software design is often used. The software design is much more about the implementation of the higher level structures and could be seen as the actual code that is written. This code could also hold structure, but is not as relevant for the relation and communication with other components on this level.

1.2 Online Open Educational Resources

The goal of this Graduation Project is to develop an open educational resource also known as an OER. This term is used for learning resources that according to UNESCO *"reside in the public domain or have been released under an open license that permits no-cost access, use, adaptation and redistribution by others with no or limited restrictions.*"[2] On the internet there are several platforms that allow OER courses to be published. This could be in the form of an online tool built on the platform itself, but also through a link to another website that hosts the tool. For this Graduation Project the latter one will most likely be used as it provides more freedom of creativity in the implementation.

Freedom of creativity in the implementation is important as one of the goals of this Graduation Project is to make the tool engaging using media and gamification. Though it is not a requirement for the tool to be a website on its own it is the most likely form. Using media in the tool mainly refers to videos. Since this technology has become available to the world it has been used in teaching material as well. For example Khan Academy on Youtube has been making educational videos since 2007 very successfully [3]. If done correctly, videos can be engaging in learning new subjects.
Another aspect that will be explored to integrate in the envisioned tool is gamification. By the use of game design elements the user will be more engaged in trying to solve problems for example. These game design elements can be seen as lenses, as described in the book *The art of game design* [4], which can be applied to exercises in the tool.

1.3 Target Audience

Since the envisioned tool for this Graduation Project is an OER it will be widely available and could serve several target audiences. As stated before it could be used by teachers or employers to direct someone to. They form a target audience to distribute the course and should therefore be able to easily find it. The actual target audience that will be using the tool is different. Anyone who needs to obtain a basic level of knowledge about software architecture

falls within this group. To make it more concrete an illustration of Creative Technology students as a specific group that falls in the target audience is illustrated as follows.

According to the Bachelor of Science Creative Technology Education and Examination Regulations Creative Technology students are globally minded societal problem solvers [5]. Among other things they acquire skills and knowledge in understanding and using technology. More specifically in the domains of programming, web technology and physical systems to name a few. Knowledge and skills in software architecture can be of great benefit for them. They should be able to develop ideas and concepts into working prototypes and evaluate them. These prototypes can contain relatively complex software systems that need to be designed and documented well. A software architecture course can be of great benefit for that. Especially learning how software architecture is done in agile projects as Creative Technology students learn to develop products with circular design methods.

1.4 Research questions

To conclude the aim of this Graduation project the following research question should be answered.

> *How do you develop an introductory software architecture teaching unit for intermediate level users in an online open educational tool?*

The underlying questions that need to be answered are therefore.

> *What does an introductory software architecture teaching unit entail?*
>
> *How do you teach software architecture using media and gamification in an online educational tool?*

The Creative Technology design process[7] will be used to develop the tool. Each of these questions will be used in this design process to come to a final design for the envisioned tool. In the realization phase of this design process a real world version of the tool will be made. The exact methodology of this project is described in Chapter 3 - Methods and Techniques.

Chapter 2 - Background

This background chapter identifies first the learning objectives of the target audience to help determine the contents of the course. After that more general research is done on how software architecture is currently taught and what the key components in there are. Then the possibilities of using media and gamification is briefly discussed and at last an analysis of related work is done.

2.1 Identifying the learning objectives

To determine the teaching material of the envisioned tool the learning objectives of the target group for this tool should be taken into consideration. Since the target group of this tool is quite broad there will be a focus on Creative Technology students as an example target group. Based on the intended learning outcomes in the education and examination regulations of Creative Technology [6] and the book Software Architecture in Practice by L. Bass et al [1] an analysis will be made on what teaching material will be important for the target group.

2.1.1 Creative Technology

To understand the intended learning outcomes of Creative Technology students as an example target group of the envisioned tool it is important to get some understanding of the main purpose of Creative Technology within the professional industry. Asking what Creative Technology students are actually educated to become.

Creative Technology is a study focused on the design aspect of bringing new products to the market. A term often used is product designer. Product design mainly focuses on developing new concepts while actively involving the user in the design process. An example of such a design process that is used within Creative Technology can be seen in Figure 2.1.

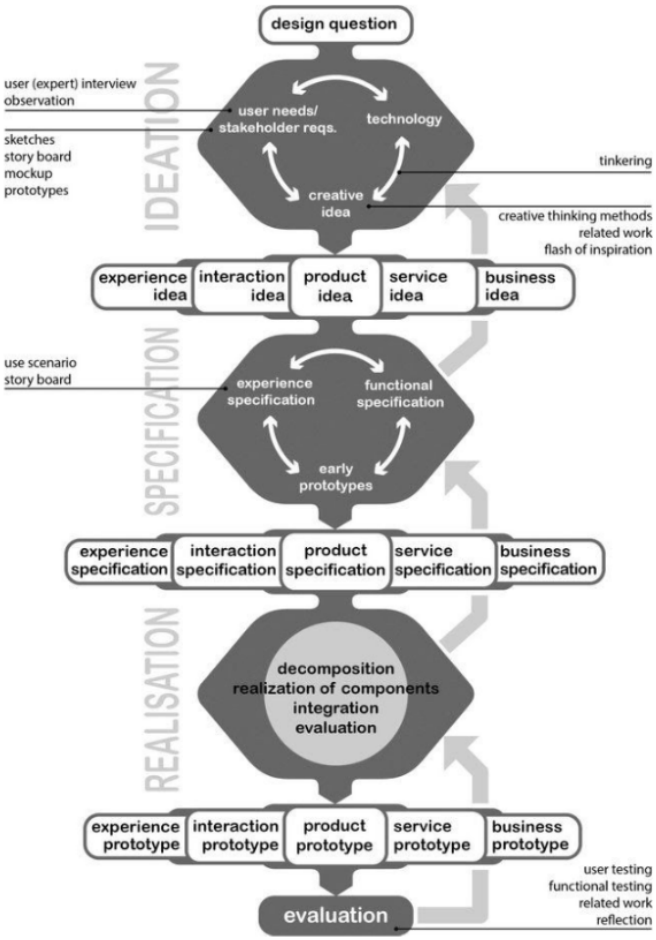


Figure 2.1: Design Process of Creative Technology, A. Mader and W. Eggink [7]

The intended learning outcomes of Creative Technology are tied to the design process. Software architecture can be beneficial within this process. The five areas where students acquire skills and knowledge according to the education and examination regulations [6] are as follows.

- *Self-managing design process,*
- *Understanding and use of technology,*
- *Designing for interaction, expression, impact and experience,*
- *Societal, economic and global competences and*
- *Academic and professional skills.*

Software architecture can be especially helpful in the understanding and use of technology, but also in explaining your product design to stakeholders which is more related to the societal, economic and global competences.
These traits described above do not only apply to current Creative Technology students but also to graduates. If they find themselves in need for more knowledge on software architecture, then they can use this to improve that.

2.1.2 Software Architecture in Practice by L. Bass et al

Now that there is a bit more understanding of an example target group and what the learning objectives of this example group are, it is time to get some understanding of what is being taught within software architecture. To do this the book Software Architecture in Practice by L. Bass et al [1] will be used to discuss several elements within software architecture. This book was selected as it is a great book for beginners who want to touch upon software architecture. It is the first book from a series of books on software architecture and covers the basics which will be enough for this project. The book Software Architecture in Practice consists of five parts. Each part will be briefly discussed to get an idea of what that part is about.

Introduction
Part one is the introduction. This part mainly explains what software architecture actually is. Since the concept is often seen as too abstract for students to understand [3] this is very helpful to give some context to the subject. This part also stresses the importance of software architecture.

Quality Attributes
The second part of the book is about quality attributes. To be able to design good architecture for software one has to understand which quality attributes are important to take into consideration. Examples of quality attributes are things such as how secure the software is or how easy it is to modify parts of the system.

Architecture in the Life Cycle
In the third part of the book software architecture is, as the title of the book states, taken to practice. The life cycle of an architecture is discussed by describing the different elements within software architecture teams have to work on in a project. For example gathering and

documenting the requirements or when a system has been designed documenting this for different stakeholders.

Architecture and business
Architecture and business is the fourth part in the book. This part describes how architectural decisions can have economical implications and how architecture is really important in software-intensive product lines. Besides these elements also the role of the architect within an organization is discussed.

The Brave New World
The fifth and final part of the book dives into some new trends that have become more and more important. One of these trends is the cloud and the other is the edge. Both are here to stay. Therefore the architectural implications that these technologies bring are discussed.

This book discusses a lot of the technical elements within software architecture. Besides these technical elements there are also soft skills for instance being able to function in a team and share ideas to discuss them. These elements within software architecture will be much harder to learn through a book, but instead come from experience[3].

2.1.3 The Relevance of Software Architecture in Practice by L. Bass et al for Creative Technology

Having discussed both the learning outcomes, the bigger goal of Creative Technology and the book Software Architecture in Practice by L. Bass et al we can identify what parts of the book are potentially relevant for Creative Technology students to incorporate in the envisioned tool. Each part of the book will be briefly discussed in relation to the Creative Technology learning objectives.

Introduction
The first part of the book is relevant for Creative Technology students as it provides the context needed to understand the importance and purpose of software architecture. Especially the many contexts of software architecture can be relevant to illustrate how communicating about software architecture can be relevant for Creative Technology students. When students

understand how architecture in the business context can be helpful in bringing a product to the market they might also be more motivated to do so.

Quality Attributes

Understanding all the different quality attributes of software architecture and how to design for them might not be very interesting for Creative Technology students. Just a basic understanding of what quality attributes are and how architectural design choices influence these quality attributes seems sufficient. Making very sophisticated software systems is not really a quality that Creative Technology students should obtain. For them it is more important that they understand how a system is put together. This is done by for example being able to understand certain diagrams of the system and the context of those diagrams. Students should be able to tell for which target group a certain diagram is made.

Architecture in the Life Cycle

Making diagrams is part of the documentation of software architecture. This is the most relevant chapter from part three of the book. Besides the chapter on documentation also the chapter on architecture in agile projects, architecture and requirements and designing and architecture are relevant for Creative Technology students. In the book software architecture for agile projects has been specifically taken . However for Creative Technology students architecture in projects in context of the Creative Technology design methodology would be interesting. This has just like agile design a similar focus on frequent iterations, prototyping and explorative design. The envisioned tool could therefore cover this chapter more broadly taking the Creative Technology design process in consideration.

Architecture and business

Part four of the book seems to be out of scope for Creative Technology students, however it can still be interesting since entrepreneurship is being stimulated within Creative Technology. So for students who start their own business or decide to shift more towards a business type of job this part can be relevant. As for now though this part is not as relevant for the majority of Creative Technology students.

The Brave New World

Relatively new technologies such as the cloud in part five of the book are definitely of interest to Creative Technology students. Coming up with innovative ideas using the newest technologies

is at the core of Creative Technology. Therefore being able to understand how these technologies are realised in architectural designs will be relevant for those students to help them build high-fidelity prototypes.

2.2 Teaching Software Architecture

Software architecture is a difficult topic to teach [8]. Based on several reports on teaching software architecture some of the key difficulties and teaching strategies will be discussed. Besides that teaching in a specific context namely online will be discussed. What are the challenges and possibilities of teaching in an online tool are the main questions to be answered.

2.2.1 Teaching software architecture in general

By analysing several reports on teaching software architecture a few observations have been made. First of all the fact that teaching software architecture is difficult. There are a few reasons why this is the case. Galster and Angelov[8] list a total of eight challenges of teaching software architecture that were identified. These challenges were also recognized by van Deursen et al [9], Rupakheti and Chenoweth [10] and Lieh and Irawan[11]. An important example of these challenges is the abstractness of the concept software architecture. This abstractness has several challenges that come along with it. For example it happens to be that especially technical students are not very good at abstract thinking. They are used to very concrete problems and solutions which are not found within software architecture. This can become very frustrating and demotivating to continue learning.
Another problem found in the mentioned reports[4, 5, 6] is the lack of experience in the industry among students. This lack of experience prohibits them from making informed decisions. Yet another challenge described by Galster and Angelov is the difficulty of documentation. It requires students to understand what view to use and based on which stakeholders are involved. UML is suitable for lower level design documentation, but when it comes to the higher level architecture clear guidelines are missing. A proposal for such high level documentation has been done by S. Brown in his C4 model [12]. These are some of the major challenges within teaching software architecture and could be considered when designing a new course.

Besides these challenges there are several elements that all three of the teaching reports describe as important aspects to teach in a software architecture course. One of these aspects is soft skills or social skills. This entails being able to communicate well about architectural

decisions for example. When designing software architecture one should be able to present and justify their architectural decisions. This could be within a team or to stakeholders for example. It is a really important aspect of software architecture that is hard to learn from paper and comes primarily from experience. The previously mentioned teaching reports therefore all advocate a course where students take on a project in a team. This way students will be stimulated to discuss different architectural design ideas. In both the course from van Deursen et al [9] and the course from Lieh and Irawan[11] students also presented their architectural decisions.

2.2.2 Teaching software architecture in an online tool

Now that a brief summary of the challenges and tactics to teach software architecture in a regular course has been given, the decision to and implications of teaching software architecture in an online tool will be discussed.

The decision to investigate the development of an online tool arose from discussions with the stakeholders. As previously mentioned in Chapter 1 - Introduction the degree of freedom in Creativity for the implementation of the tool is broad. An online tool also allows for more easy accessibility by individuals especially.

There are however some implications of this decision. The main one being that so-called soft skills will become more complicated. As described in section 2.2.1, soft skills are taught by letting students work in teams where they can discuss architectural decisions with each other. In an online tool working in teams will be much more difficult to realize. Especially when you want people to use the tool individually whenever they want. Therefore when designing an online tool that teaches software architecture there are decisions to be made on this matter.

2.3 Applying media and gamification in teaching

As the envisioned tool will be online it is interesting to explore how this can be used to make the tool more engaging for users. Both media and applying gamification will be discussed as opportunities to accomplish this goal.

2.3.1 Media

When talking about using media in teaching there are two main technologies to distinguish. These technologies are images or photos and video. Here the focus will be on how video can be used to make engaging content for students.

Videos are widely used current teaching material especially for online courses. They vary from long recorded lectures to short snappy explanation videos. In a paper by Guo et al [13] they looked into how student engagement was affected by video production decisions. With their research they came up with a list of findings and corresponding recommendations (see Figure 2.2) to help creators of educational videos to make more engaging content. These findings will be very helpful in the implementation of the envisioned tool.

Finding	Recommendation
Shorter videos are much more engaging.	Invest heavily in pre-production lesson planning to segment videos into chunks shorter than 6 minutes.
Videos that intersperse an instructor's talking head with slides are more engaging than slides alone.	Invest in post-production editing to display the instructor's head at opportune times in the video.
Videos produced with a more personal feel could be more engaging than high-fidelity studio recordings.	Try filming in an informal setting; it might not be necessary to invest in big-budget studio productions.
Khan-style tablet drawing tutorials are more engaging than PowerPoint slides or code screencasts.	Introduce motion and continuous visual flow into tutorials, along with extemporaneous speaking.
Even high quality pre-recorded classroom lectures are not as engaging when chopped up for a MOOC.	If instructors insist on recording classroom lectures, they should still plan with the MOOC format in mind.
Videos where instructors speak fairly fast and with high enthusiasm are more engaging.	Coach instructors to bring out their enthusiasm and reassure that they do not need to purposely slow down.
Students engage differently with lecture and tutorial videos	For lectures, focus more on the first-watch experience; for tutorials, add support for rewatching and skimming.

Figure 2.2: Screenshot of the main findings in the paper by Guo et al [13]

2.3.2 Gamification

Besides the media element to make the tool more engaging also gamification can be used to make learning more fun and interesting. Gamification is already used in education to motivate students and encourage more involvement.

Using gamification to make exercises more fun can be put to practice by applying different lenses as described in the book The Art of Game Design by J. Schell [14]. These lenses can be seen as game elements that make the exercises more fun. Especially in an online environment gamification can be applied very well. In an online environment such as a website for example there are a lot of possibilities to add dynamic elements. These dynamic elements would be much more difficult to implement in real world exercises on paper for instance. Therefore adding gamification to an online teaching tool is a great opportunity to increase student engagement.

2.4 Related work

To get some more understanding of the context where this proposed tool should be deployed, related software architecture courses were analyzed. These software architecture courses can all be done online by individuals. The courses discussed here are not free courses, this is already different from my envisioned tool but can still be helpful in understanding the context. They were analyzed asking what is being taught, for whom and how is this done. What focuses on the contents of the course. So is it more technical or more focused on soft skills? Whom dives into whom this course is targeted at and what is the goal to become after completion of the course? Finally, how is asked to focus on the type of teaching. Does it consist of videos, text, exercises etc. How is the teaching material implemented?

2.4.1 The Complete Guide to Becoming a Software Architect (Udemy)

This course on Udemy by M. Lavi [15] claims to be a complete guide to become a software architect. It states that becoming a software architect is the holy grail for almost all developers. The contents of this course are therefore really focused on the software architecture role. It discusses different topics around software architecture starting with the question what is a software architect. Then several technical sides of the task of a software architect are discussed. For example what system requirements, design patterns are and what an architecture document is. At the end also soft skills are discussed.

The course is taught in video lectures with corresponding exercises. Towards the end of the course the user will work together with the teacher on a real-world case study. In this case study the student has to design its architecture. The student will also have the opportunity to obtain a full blown architecture document which he or she can modify and use.

The target audience according to the course itself is to “developers who want to go to the next level” and “system analysts who want to achieve technical knowledge”. It also states that two years of developer experience is preferable. Compared to the target audience of my envisioned teaching resource this course seems to be for a different audience who already has some significant experience within software development.

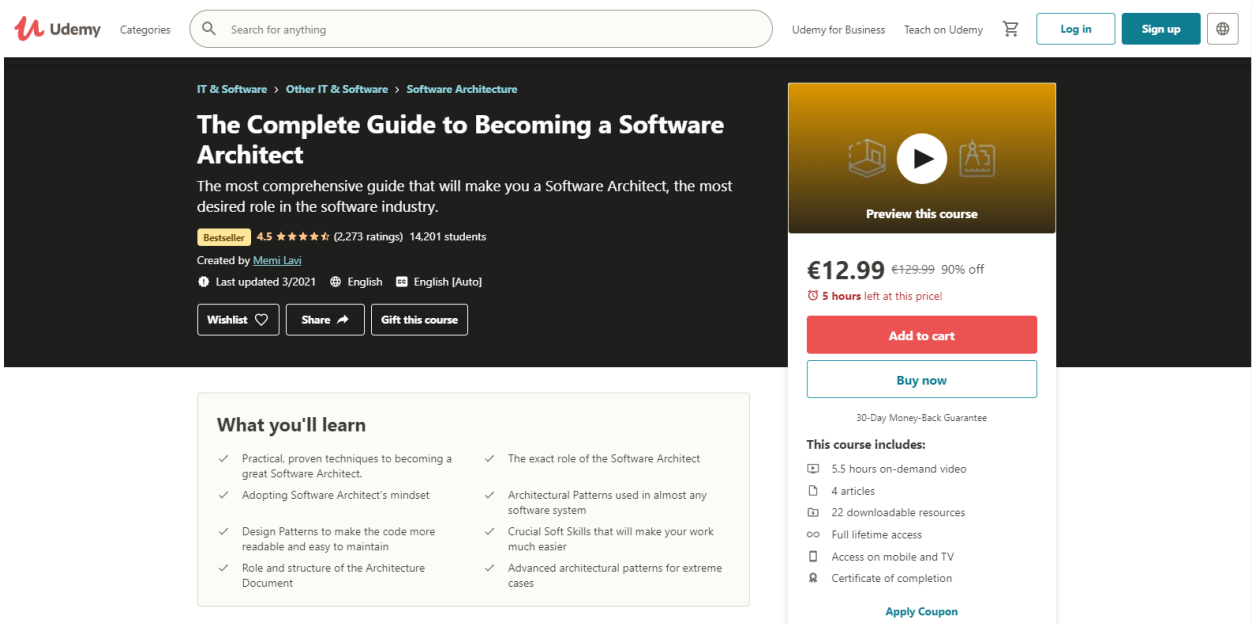


Figure 2.3: The Complete Guide to Becoming a Software Architect by M. Lavi [15]

2.4.2 Developer to Architect (Pluralsight)

The online course Developer to Architect by C. Simmons [16] teaches you about the role of the software architect. The course consists of three parts. First the role of the software architect in the enterprise is treated. This part discusses technical and non technical duties of a software architect in the enterprise in general. Then the second part focuses on the role of the architect in the life cycle of a project. Different phases are discussed such as the design, testing and implementation for example. The third and last part is about designing the solution. This part discusses the design process and how to communicate about the solution. You will learn how to communicate about a solution to both technical and non-technical stakeholders.

As the title of the course says, this course is aimed at developers who want to transition from a developer position to a role as a software architect. Also software architects who want to learn more on this subject can do this course. This means that the course is not for beginners who

have very little experience, which is also stated in the course info that says the level is intermediate. If you do have the developer experience this course states that it can be a roadmap to becoming a capable and successful software architect. The contents of this course consist of video lectures only with a total duration of two hours and forty one minutes.

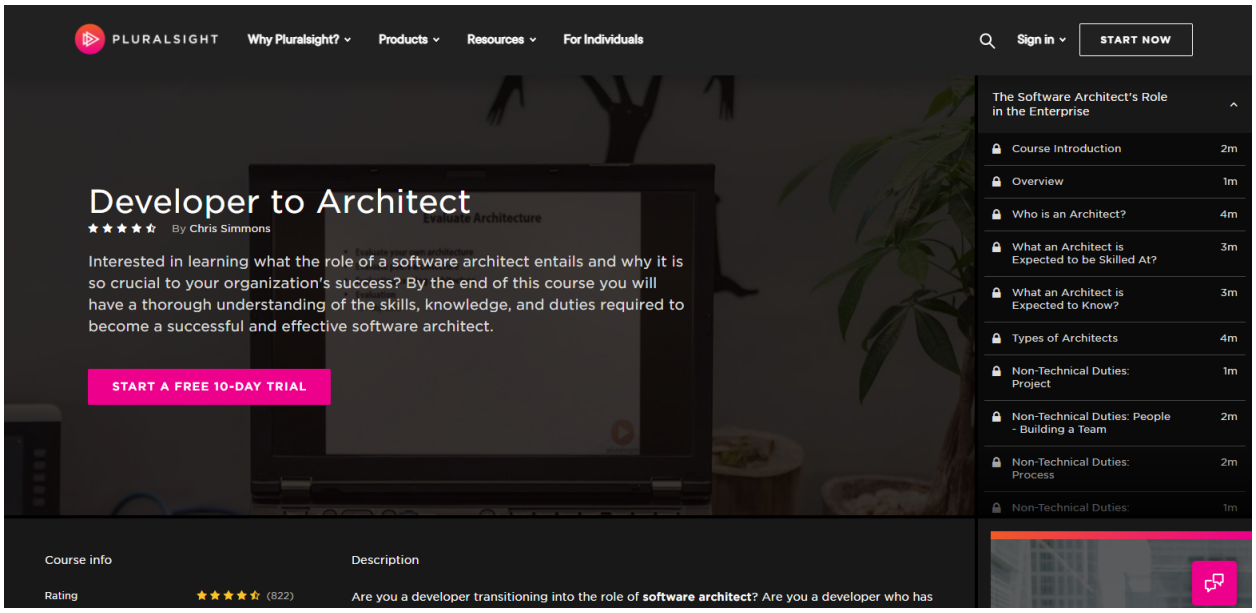


Figure 2.4: Developer to Architect by C. Simmons [16]

2.4.3 Software Architecture (Coursera)

This course Software Architecture by K. Wong [17] is actually offered by the University of Alberta. The course software architecture is part of the Software Design and Architecture Specialization as it states. The first week of the four week course revolves around UML Architecture Diagrams. In the second week different architectural styles are discussed. The third week focuses on how architectures are put in practice by discussing how teams describe, plan and evaluate the architecture. In the final week students will evaluate a proposed architecture. The course does not discuss soft skills as a part of this course.

It is not clear to whom exactly this course is aimed at. The only indication that can be found is that it is an intermediate level course, which suggests that some experience within programming is a prerequisite. Especially when looking at the project where students have to evaluate the architecture of an android code base program basic java knowledge will be handy. There is not a career outcome such as becoming a software architect as the goal of this course. Instead

there is a list of learning objectives which mostly revolve around the technical side of software architecture.

The way this course was implemented is in a mixture of video lectures, reading material and exercises. In total the course takes about ten hours to complete. What is unique about this course is the weekly schedule where each week a different topic is discussed. This gives a clear guideline for the pace of the course.

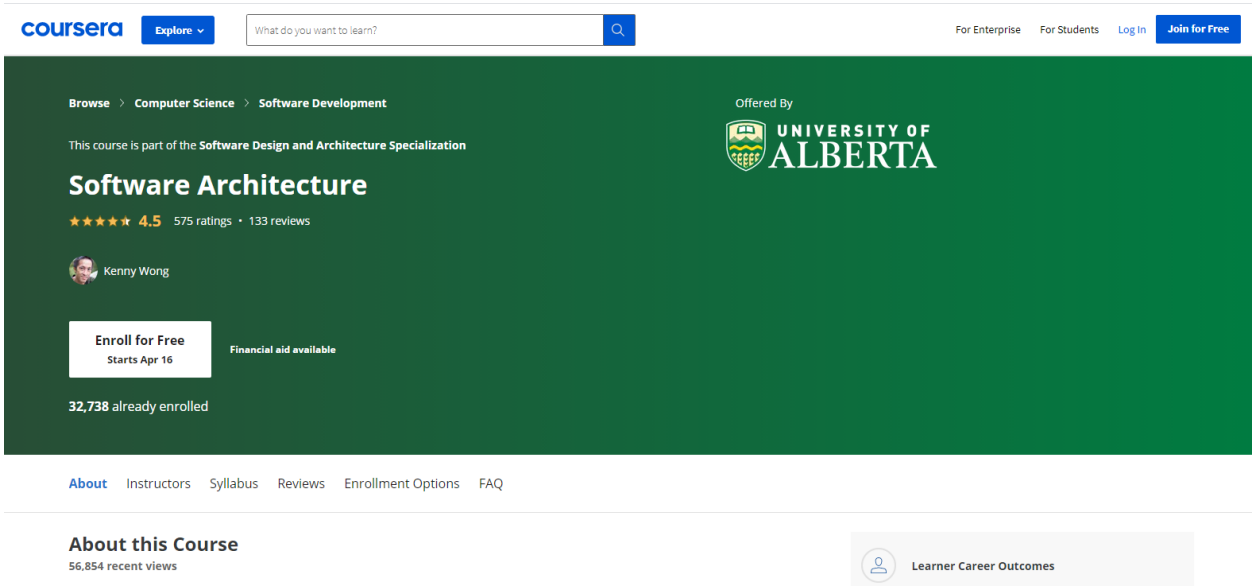


Figure 2.5: Software Architecture by K. Wong [17]

2.4.4 How to Become An Outstanding Solution Architect (Udemy)

The course How to Become An Outstanding Solution Architect by M. Farragher [18] claims to teach both the soft and the hard skills to take your architect design skills to the next level. The course covers several aspects of the role of the architect and the team. Besides that more technical aspects are covered, for example design patterns and designing using UML.

The course claims to be for a wide range of people as it states that beginners, intermediates, but also advanced IT professionals can learn from it. The only requirements besides being at least a beginner are some practical requirements to fulfill the exercises. When it comes to the outcomes of the course there is a list of several learning outcomes going from soft skills to learning UML and understanding what a solution architect is. With the help of this course you

should be able to become a solution architect. It is questionable how well the soft skills are being taught as there does not seem to be any group exercise to practice those soft skills.

The content of this course consists, as most of these online courses, of video lectures. Besides these lectures there are assignments where students make their own architecture designs for example. Each subject also has its own quiz to test your skills as it says. When it comes to the duration of the course it is not clear how long it will take to complete. Given the fact that there is almost eight hours of video lecture content it is safe to assume that this course takes at least ten hours to complete.

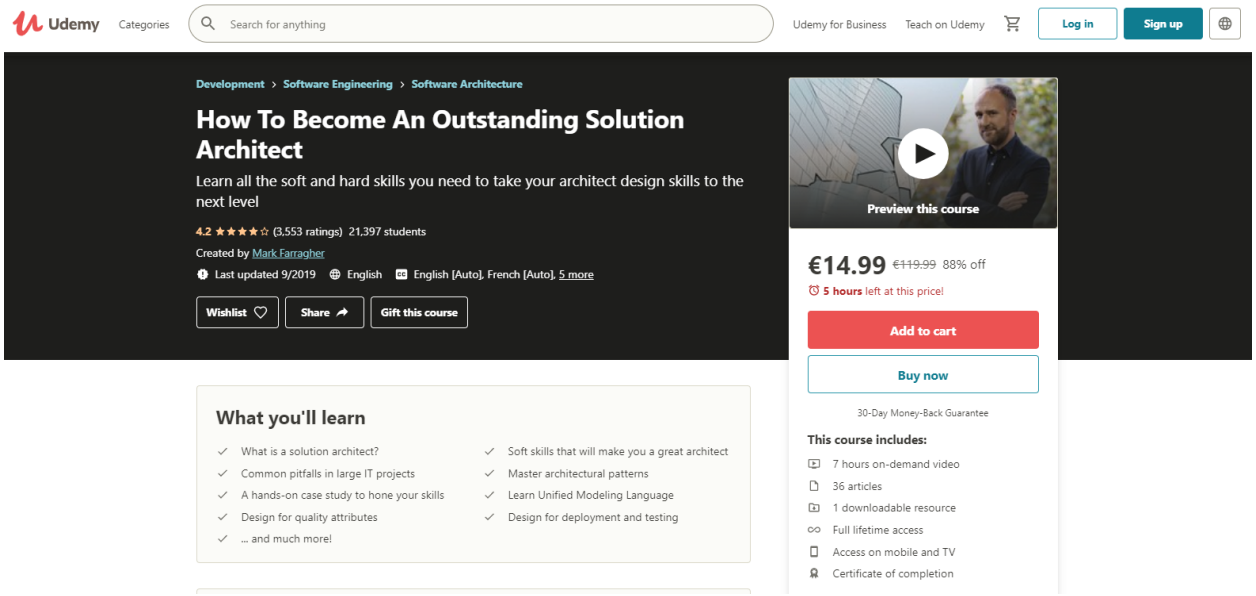


Figure 2.6: How to Become An Outstanding Solution Architect by M. Farragher [18]

2.5 Conclusions

Based on the state of the art background research a few points should be taken into consideration when designing the envisioned online teaching tool. Based on the Creative Technology students as an example target audience and the book Software in Practice by L. Bass. et al [1] it is wise to limit the scope of the course to the basics of software architecture with an emphasis on the communication aspect within software architecture.
Another point of consideration is how to implement the soft skills of software architecture into an online teaching tool. Courses on software architecture typically use team projects to let students develop soft skills. This is much more difficult in an online teaching tool.

A great opportunity of the online element of the envisioned tool is the possibility to increase engagement by applying media and gamification.
And finally when looking at related work there are two main conclusions that can be drawn. First the focus of these online courses is mostly on the role of the software architect. And second the content of these courses is mostly about the more technical aspects of software architecture.

Chapter 3 - Methods and Techniques

In this chapter the methods and techniques used in the development of the tool will be presented. These methods and techniques are done according to the Creative Technology design process, which will be discussed in this chapter.

3.1 Creative Technology Design Process

For the development of the educational tool the Creative Technology Design Process (Figure 3.1) was used. This design process combines two popular design processes in one model. First the concepts of divergence and convergence [19]. Where in the divergence phase the design space is widened and in the convergence phase the design space is narrowed. The second design process incorporated is the application of spiral models. Together they build the foundation for this design process as described by Mader and Eggink[7]. Each phase represents a certain step within the design process.

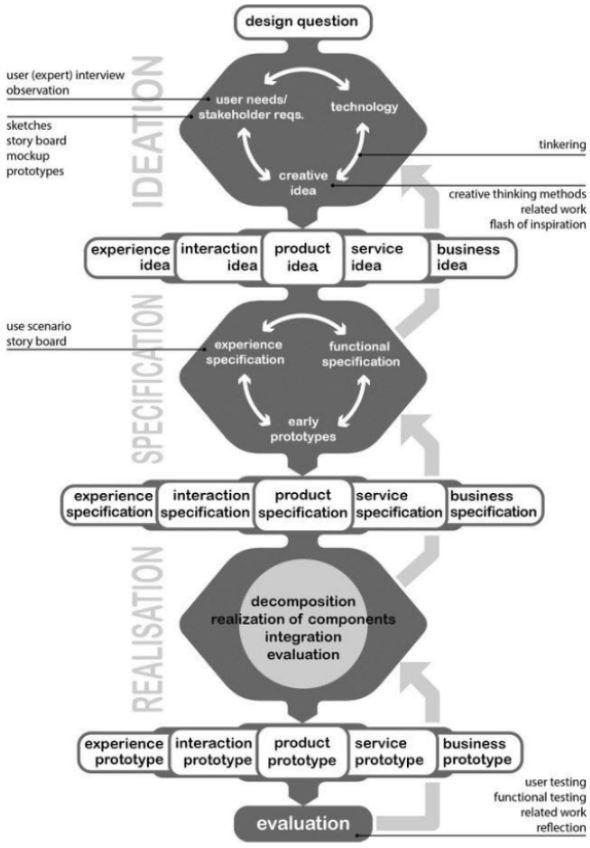


Figure 3.1: Design Process of Creative Technology, A. Mader and W. Eggink [7]

Ideation phase

In the first phase, the ideation phase, the goal is to come up with an initial concept for the product which can be further specialized in the specification phase. This first phase often starts with a design question. Different divergent methods are used to come up with several ideas for a potential product. These methods can be for example a brainstorming session, interviewing stakeholders or going through different use case scenarios. They are focussed on exploring a broader view of the design question and coming up with different creative solutions for this question. The second part of the ideation phase is to convert these concepts or ideas to a single concept or idea which can be specialized in the specification phase. This is called the convergent phase.

Specification phase

The specification phase is about exploring different prototypes. Now that an initial concept for the product has been established in the ideation phase this concept will be improved by an evaluation and feedback loop. In the specification phase prototypes are made to obtain feedback from the user group. With this feedback these prototypes could be thrown out, merged or improved. The goal is to come to a final design for the product. Coming to this final design usually requires multiple iterations of evaluating and improving prototypes. The final design is also called the product specification.

Realization phase

This phase is all about the actual implementation. Now that there is a clear specification of what the product should look like it is time to actually build this product. This requires technical knowledge and skill on how to achieve the requirements for the product. When the product is built it should be evaluated to see if it meets the specifications.

Evaluation phase

The last phase of this design process model focuses on evaluating how well the initial product requirements are met. Most important for this is the user testing. Functional testing is also important but should be done primarily in the realization phase. The goal of the user testing is to see how well the product actually satisfies the user requirements. Another aspect of the evaluation phase is to reflect on the academic progress. This can be done by going over the design process and critically reflecting on the decisions made.

Chapter 4 - Ideation

In the first phase of the Creative Technology design process, the ideation phase, divergence and convergence techniques were used to develop a concept of the educational tool. With the help of a user needs analysis, user scenarios and the conclusions from the background research several concepts for the product were obtained. At the end of this phase one final concept was chosen to be further specified in the specification phase, which can be found in Chapter 5 - Specification.

4.1 User Needs Analysis

When coming up with new concepts for a product, analysing the needs of potential users can be a good starting point. There are two main types of users that can be identified. First the students who will actually be using the tool to develop their skills and knowledge in software architecture and second the teachers or employers who can use the tool to incorporate it into their teaching course. These groups have different and also overlapping needs that need to be incorporated into the tool.

4.1.1 Types of Usergroups

First let's discuss the actual users of the tool, the people who will be using the tool to learn something about software architecture. Their interest mainly lies in the tool being useful and fun to use. Some people will only be using the tool because they were assigned to by their teacher or employer. They might not be as interested in the topic of software architecture as others who find the tool themselves. It will therefore be important that the tool is motivating or in other words fun to use. To achieve this need it would be nice if the tool has interactive components that engage the user in the learning process. Besides that the content should be manageable, preventing large chunks of information.

For the teacher, employer or anyone else who might use this course to implement it in his or her teaching course there are other needs that would be nice if they were met. The most important being the need that the tool will be effective in teaching the basics of software architecture. Besides that the tool should also be easy to implement in a course. It would therefore be nice if the tool requires only minimal to no work from this type of user.

4.1.2 Use Case Scenarios

To illustrate the needs of the different user groups several use case scenarios have been written out.

Scenario 1

As part of an educational course a Creative Technology student is building a social media board for a company. On this board leds are used to display the social media activity of the company. The company for whom this board is made wants to understand how they realised this system. It should for instance be clear how social media activity is obtained and displayed to the board. There is no clear diagram of the system yet. Therefore they decide to complete the software architecture introductory tool. After completion of the tool they have become able to make such diagrams explaining the crucial elements and structures of the system. A diagram was made which was used in the pitch to the client.

Scenario 2

A Industrial Engineering and Management (IEM) student is participating in a Computer Science minor. Within this minor students are working on a project where they have to design a software system for a social networking application. To obtain more understanding of the architectural decisions being discussed in the project group the IEM student follows the software architecture introductory teaching unit. After completion of the teaching unit the student still has difficulty to understand the specific technical choices being made. However the context of these choices is much more clear and especially on a higher abstraction level the student is able to explain the architectural structure of the envisioned system for the social networking application.

Scenario 3

A small business owner who sells cheese to people at home is planning on expanding his services online. He reaches out to a software development company to make a website with an ordering system. The IT architect of the software development company is explaining how they would build such a system with the use of a few diagrams. The small business owner does not have enough knowledge to understand the type of choices being made, but wants to be very much involved in the development process. Therefore he decides to educate himself a bit using the software architecture teaching unit. After completing the teaching unit the small business owner has much more understanding of the architectural structure and steps taken by the

software development company. He is comfortable with the new system and is able to ask the right questions in case of a bug in the system.

4.2 Concepts

Based on the background research and the user needs analysis different versions of a general main concept were created. First the main concept will be described then the different versions of this main concept will be discussed and at last a conclusion with a description of the final concept will be presented, explaining why this concept was chosen.

main concept

The main concept for the educational tool is to build a website that incorporates videos and exercises that teach software architecture. Users can visit the website whenever they want and go through the videos and exercises that are hosted on the website. The reason why this main concept was chosen is because of the accessibility of the tool and the ability to insert interactive components into the tool.

The accessibility is important to allow teachers to easily implement this tool into their course. This can be done by just sharing a link. Besides that users all around the world can use the tool free of charge provided that they have an internet connection. A physical educational tool would not provide this accessibility. A website also allows for incorporation of interactive components which will be especially useful in making the tool engaging.

Sub Concept One

Given the main concept there are several ideas on how to design the content of the website. In the first concept the interactive component will be very important. By the use of gamification the exercises should become very engaging. This would be a tool that is not something you complete once and never look at again, but instead something where you can train your software architecture skills by playing educational games.

Sub Concept Two

In this concept the tool would not only be a static educational tool that one can do on their own, but also an interactive online platform to share ideas and discuss architectural designs. By sharing and discussing architectural choices users can train themselves to obtain the soft skills

that are needed for good communication about architectural design. This aspect of teaching software architecture is difficult to include in the main concept of the tool as it requires a social element.

Conclusion and the Final Concept

The final concept that was chosen for this project is the main concept without the implementation of any of the sub concepts. The main reason for this decision is to make the development of such a tool feasible. It was estimated that the development of a high fi prototype for either of the subconcepts was not possible in the given period of time for this project. Besides that the belief is that most people would already benefit very greatly from just the main concept.

Chapter 5 - Specification

In this phase the requirements and product specifications of the educational tool are further specified. An initial prototype was made to evaluate and improve upon received feedback.

5.1 Requirements

To prioritize the requirements for the educational tool the MoSCoW method[20] was used. This method prioritizes the requirements in four different groups. First the must haves, which are the requirements that must be satisfied before the tool can be considered a success. Then there are the should have requirements, which are the requirements that should be included if it is possible. The third group is the could have requirements, that can also be described as nice to have requirements. These requirements are desirable but not necessary. The last category is the won't have requirements. These are requirements that will not be in the first version of the product but may be implemented in the future.

Given the method described above Table 5.1 was made categorizing all the requirements identified from the background research as well as the ideation phase.

MoSCoW	Requirement
Must	The content of the tool must cover the basics of software architecture.
Must	The tool must be effective in teaching the basics of software architecture
Must	The tool must be accessible on a OER website
Must	The tool must include videos to cover the content
Must	The tool must include exercises
Should	The exercises should be engaging by providing feedback to the user
Should	The tool should be fun to use
Could	The tool could be part of the teaching material in the Creative Technology program
Could	The tool could include gamification elements to increase engagement.
Could	The tool could include animated videos to increase the production value
Won't	The tool won't be saving data from user input

Table 5.1 The product requirements categorized using the MoSCoW method

5.2 Product Specification

Keeping the product requirements in mind a product specification was made. This specification defines the design choices that were made for the videos, content and website.

5.2.1 Videos

When making educational videos the length of a video is an important aspect that has an impact on the engagement. Guo et al.[13] analyzed data from 6.9 million video watching sessions and found that longer videos are less engaging than shorter videos. In their findings the engagement, measured in watching time, dramatically decreased for videos longer than nine minutes. They recommend sticking to videos no longer than six minutes as there was also a slight decrease in engagement for videos longer than six minutes. For this educational tool the choice was therefore made to try to keep videos under six minutes and restrict them to a maximum length of ten minutes.

Besides recommendations on the length of educational videos Guo et al.[13] also provided recommendations for the video production style. Based on their findings they recommend introducing motion and continuous flow into tutorials, along with extemporaneous speaking. In their findings they found that Khan-style[3] live drawing videos were more engaging than PowerPoint slides or code screencasts. Since the content of the educational tool did not lend itself for live drawing Khan-style videos the choice was made to go with slides while maintaining flow in the video. This meant going through many slides and directing the eyes of the viewer by highlighting the aspects that are talked about.

5.2.2 Content

As described in Chapter 2 - Background Research, the book that was decided on as the main source of content for the educational tool is the book Software Architecture in Practice by L.Bass et al. The chapters in this book are split into different sections. These sections were a convenient way of dividing the content into separate videos. A summary video for each section would result in videos of at maximum ten minutes and on average around five minutes. This fitted perfectly with the earlier established specification that videos should not take longer than ten minutes and be preferably under six minutes.

Besides a video every section will have one to two different exercises that test the gained knowledge from the educational video. These exercises focus on the most important elements that are being taught in the specific section. The reason why there are only one to two exercises is to keep the flow of the tool going. This way the sections are manageable and therefore easier to go through.

5.2.3 Website

An important requirement for the website is that it is mobile friendly. In 2020 mobile website visits were responsible for 68% of the global total website visits[20]. It is therefore important to make sure the tool works on both desktop and mobile. Besides this the website should be functional. Navigation must be easy and the use of the tool should be intuitive.

5.3 First Prototype

Using the final product specification a first prototype was realized. The technical realization of this prototype is described in Chapter 6 - Realization. Figure 5.1 provides some screenshots of this first prototype. The first prototype can also be found at edu-tool.arno-de-vries.nl/index-old.html.

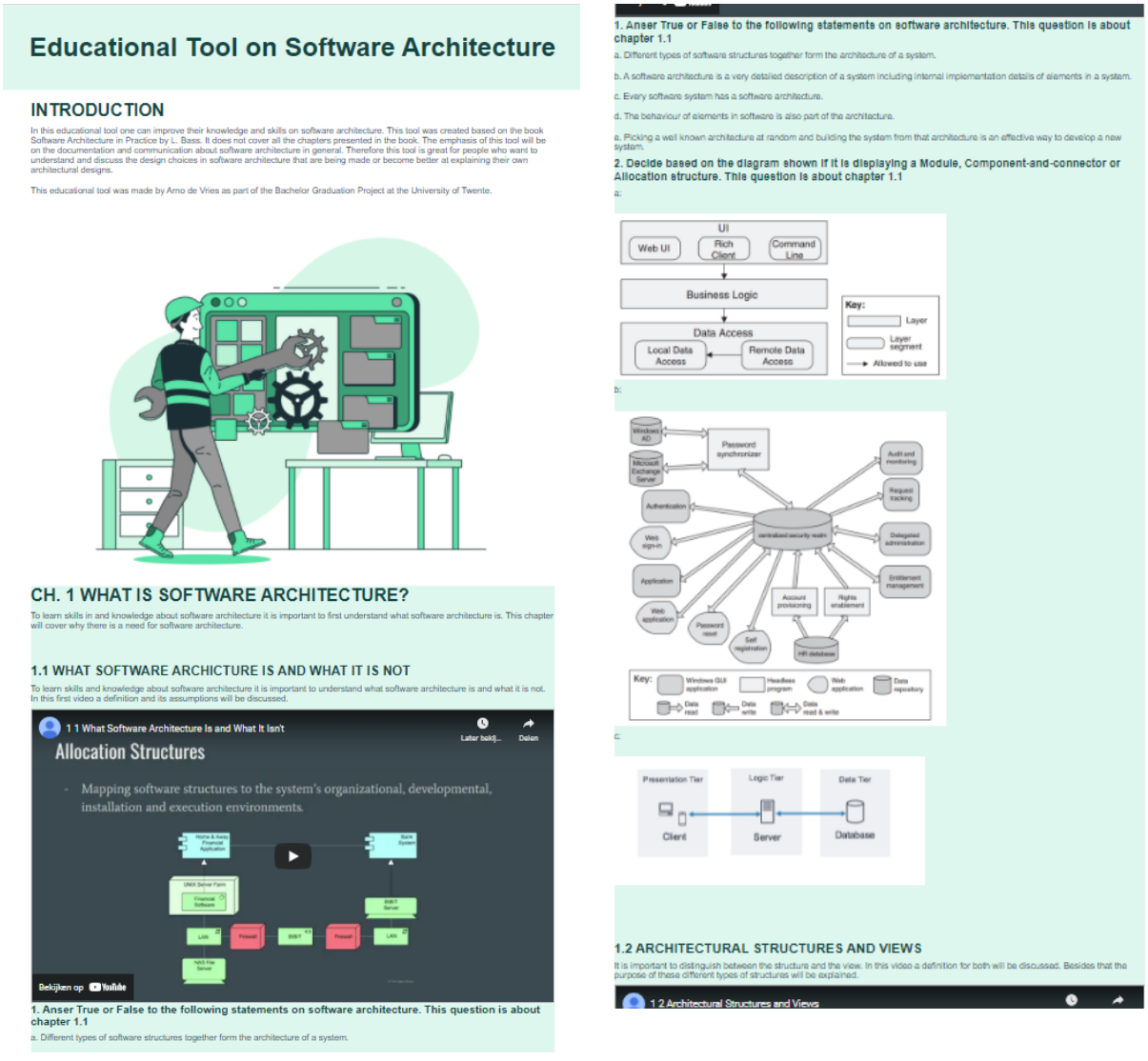


Figure 5.1 Screenshots of the first prototype

Chapter 6 - Realization

In this chapter the realization phase will be discussed. First the implementation of the videos and website will be described. Then the first evaluation that led to some changes in the design of the tool will be reported.

6.1 Videos

The realization of the videos was done in a four step procedure (Figure 6.1) for each video. This procedure starts with writing a script that summarizes a chapter section from the book Software Architecture in Practice [1]. This script was then used to design the corresponding slides. The slides were made in Google slides and using a dark blue grey template. After the preparation was done a video was recorded by going over the slides and reading through the prepared script. OBS Studio and a Fifine K669b microphone were used to record the screen and voice over. This video was then edited using the DaVinci Resolve 16 video editing software.

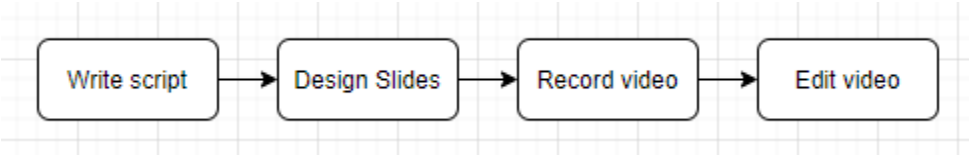


Figure 6.1 Four step video realization procedure

After completion the videos were uploaded to Youtube so they could be embedded in the website of the educational tool. The scripts and slides of the four videos that were made can be found in Appendix A. The videos themselves can be found on the website of the educational tool (edu-tool.arno-de-vries.nl).

6.2 Website

The realization of the website will be broken down into two sections. First the Technical Design will be described and then the Visual Design will be discussed.

6.2.1 Technical Design

The website was realized by writing in HTML, CSS and Javascript using Sublime Text 3 as the code editor. The choice to write it from scratch instead of using an OER tool maker or

Wordpress was to allow for more Creative Freedom. By writing in HTML and CSS you have full control over the styling and layout of the tool.

Since the tool does not save any data from the user, the website only required a frontend. The simple file structure for this frontend website can be found in Figure 6.2.

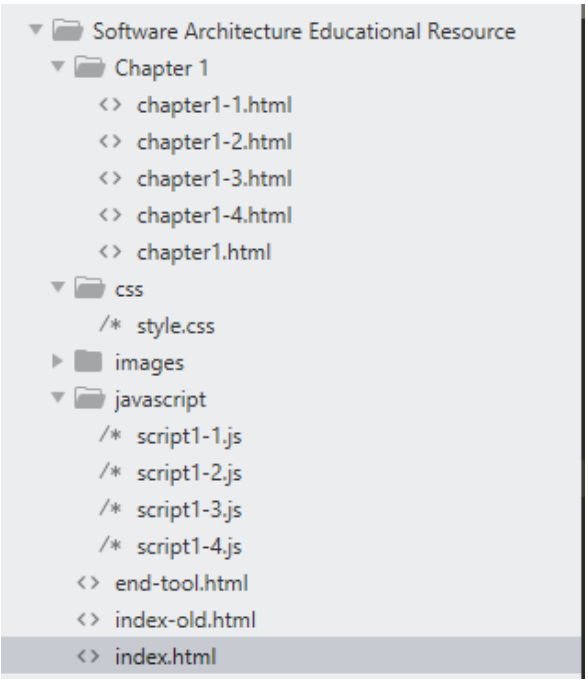


Figure 6.2 File structure of the website

As can be seen in Figure 6.2 each chapter section has a corresponding Javascript file. This Javascript code is responsible for the implementation of the interactive aspect of the exercises. These exercises are all multiple choice exercises to allow for easy validation (see Figure 6.3). Besides that they are also less time consuming than writing out full answers.

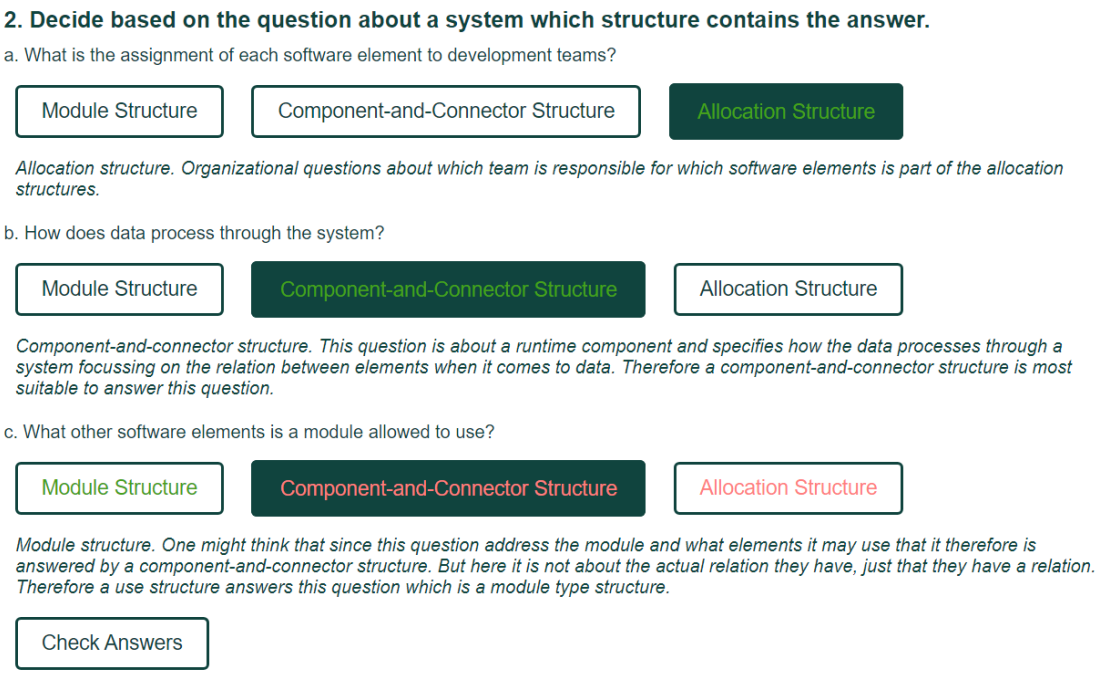


Figure 6.3 Example of one of the multiple choice exercises that can be validated

The website is hosted on the domain name edu-tool.arno-de-vries.nl at a Strato web server. This choice was made as there is no sensitive data being stored and because it was convenient. When this tool would be published on an OER platform this domain name can be changed. For full inside into the files that make up this website see Appendix B.

6.2.2 Visual Design

As part of the website a visual design had to be created. To make the tool cohesive a simple color scheme was chosen that was used throughout the whole tool. This color scheme consisted of white as the main color and green and grey as the accent colors. The reason for this simple color scheme was to create a peaceful and non distracting environment so the focus can be on the content of the tool. Figure 6.4 illustrates examples of the design of the tool.

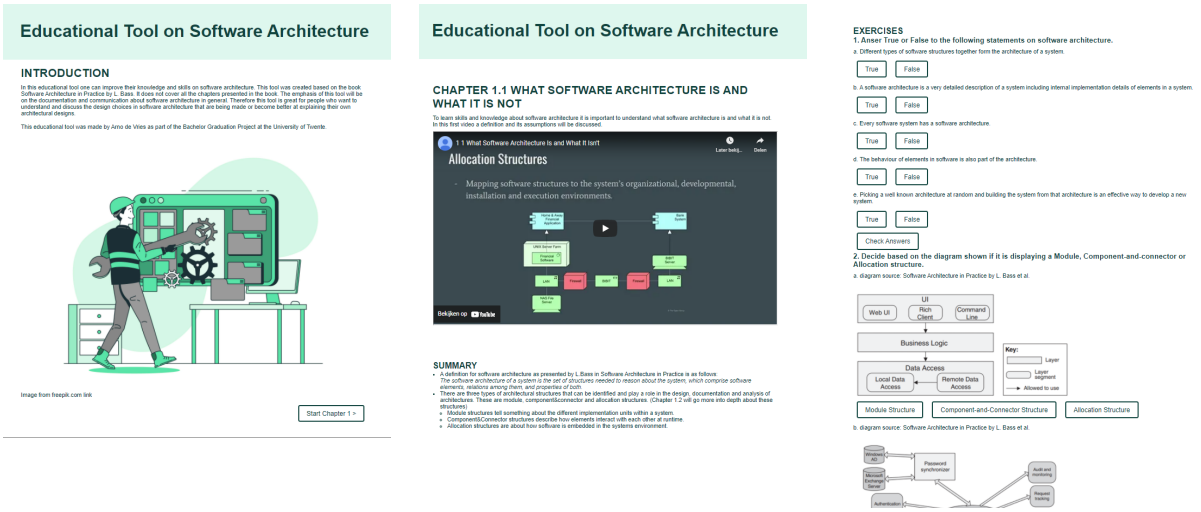


Figure 6.4 Screenshots of the design of the educational tool

6.3 Evaluation I

As part of the realization phase an evaluation was done on the first prototype of the tool. This prototype can be found at edu-tool.arno-de-vries.nl/index-old.html. See Figure 5.1 for screenshots of this first prototype. This evaluation was done in the form of a survey. This survey and the results will be discussed.

6.3.1 Survey I

The goal of this survey was to obtain usable feedback that can be used to improve the tool. In this survey participants were asked to first visit the educational tool and then answer some questions about this. Three different aspects of the tool were questioned. These are the usability, design and effectiveness of the tool. After that some general and a few demographic questions were asked.

The survey used two types of questions, likert scale questions for quantitative feedback and open questions for more elaborate feedback. The likert scale questions were partially derived from the Evaluation of Interactive products meCUE 2.0 questionnaire[21]. Besides these questions some questions were added to make the survey more suitable for the tool.

The survey was made in a Google Forms document. This survey was then spread to Creative Technology students through Whatsapp and ran for the period of a week. The survey can be found in Appendix C.

6.3.2 Survey I Results

Over the period of a week five responses were obtained. This was very minimal, but enough to improve the tool based on the obtained feedback. The results for each aspect of the survey will now be summarized.

Usability

Overall the usability was found good. The tool was found very self explanatory and therefore easy to use as well. Most of the critique on the usability was on the exercises. According to a respondent it would have been better if the exercises were directly below each video. Another respondent suggested that an answer fill in form and submit button to check the answers would be a good addition to improve the usability of the tool.

Design

The design of the tool was found to be average. Also the engagement was not found to be above average. The main critique from the open questions was again on the exercises. The show button looked outdated according to one respondent and another respondent found the questions to be too much text.

Effectiveness

Overall the effectiveness of the tool was found to be good. Most respondents found the tool to be useful and all the respondents thought this tool could help them learn more about software architecture. For some respondents the learning objectives were not quite clear though. One respondent commented that it is not clear who the target audience for this tool is and what knowledge is needed beforehand.

The complete survey results can be found in Appendix D.

6.4 Applied Changes

After obtaining and analyzing the results from the survey a list of proposed changes was made. These changes were implemented into the second prototype version of the educational tool. They are the following.

Each chapter section has its own designated webpage including the accompanying exercise(s).
This was one of the suggestions from the respondents and was added to cut the content into more manageable pieces. By splitting the exercises as well the big clump of text is taken away.

A separate end page and front page with an illustration was added.
Since the chapter sections have been split up it made more sense to add a separate front and end page for this tool.

Each video is followed by a small written summary underneath.
One of the participants suggested an option to see a written version of the video. As I did not want the tool to have a big chunk of text I decided to add a small summary. This summary can function as a small recap so users can repeat for themselves what they have learnt from the video.

Answers to the exercises can be filled in and validated by the use of a check answers button.
Most of the critique on the tool was about the exercises. They were not really engaging. Therefore as suggested by one of the respondents the exercises are now filled in and checked by the user to provide a more interactive experience.

Applying all these changes resulted in the second prototype of the educational tool. Figure 6.4 illustrates this version with some screenshots of the new front page, the summary that was added and the interactive exercises.

Chapter 7 - Evaluation

In the evaluation phase a final evaluation on the educational tool was performed. A similar survey to the survey in evaluation I was conducted to be able to compare the results while also evaluating if the user requirements were met. At the end of this phase the Creative Technology Design Process was being reflected upon.

7.1 Evaluation II

The goal of evaluation II was to both validate that the applied changes from evaluation I improved the tool as well as evaluating how well the user requirements were satisfied.

7.1.1 Survey II

As mentioned in the introduction of this chapter a survey similar to the survey in evaluation I was conducted. This survey contained the same questions and also some extra questions regarding the exercises which were dramatically changed in the second prototype. Also the procedure of the survey(visiting the tool and afterwards filling in the form) remained the same. The survey was again conducted over the period of one week. The complete survey can be found in Appendix E.

7.1.2 Survey II Results

For the survey in evaluation II five responses were obtained, all from Creative Technology students. A brief summary of the results will be given below.

Usability
The usability was overall found to be good. The majority of the respondents found the interactive component of the exercises to be intuitive and except for only one respondent no one was frustrated by using the tool. The written feedback was mainly concerned with some minor bugs and small suggestions to improve the usability even more.

Design
The design of the tool was found to be good by some respondents and bad by others. When it came to the design of the exercises and the engagement of the tool the majority of the

respondents was positive. Only one comment was given stating that the design looks boring and was not sparking interest.

Effectiveness

The rating on the effectiveness of the tool was quite diverse, but leaning more towards a positive valuation. All the respondents answered that the exercises helped them become more engaged with the tool.

The complete results of the survey can be found in Appendix F.

7.1.3 Comparing Results Survey I and II

To validate if and where the second prototype of the tool is better compared to the first prototype the results of survey I and II have been compared. The results from all the Likert scale questions that have been asked in both surveys have been averaged out and presented Table 7.1. By the use of a two tailed two sample t test the results have been tested on whether there is a significant difference or not.

	Survey I (N=5)		*Survey II (N=5)*		
Question	***Mean***	***SD***	***Mean***	***SD***	***P value***
The tool is easy to use	4.00	0.71	4.60	0.55	0.1720
It is clear how to use the tool	4.60	0.55	4.60	0.55	1.0000
Using the tool frustrated me	2.00	1.00	1.80	1.30	0.7924
The design of the tool is attractive	3.20	0.84	3.20	1.64	1.0000
The tool is engaging	3.00	1.00	4.00	1.22	0.1950
The tool is interesting	3.20	1.30	3.60	0.55	0.5447
The tool is very useful	3.80	0.84	4.40	0.89	0.3052
I understand the contents of the videos	3.80	1.30	3.60	1.14	0.8028
This tool can help me learn more about software architecture	4.60	0.55	4.60	0.55	1.0000
The learning objectives of the tool are clear	4.00	1.00	4.40	0.89	0.5237
This tool will achieve the proposed learning objectives	4.00	0.71	4.20	1.10	0.7404

Table 7.1 Comparing the Results of Survey I and II by the use of a Two Sample T Test

As can be seen in Table 7.1 there are no significant differences found between survey I and II. All the P values are higher than a significance level of 0.05. The main reason why there were no significant differences found can be the small sample group for both surveys. When the sample groups are bigger it would be easier to find the significant differences if there are any. While these survey results have not proven a significant difference, this does not mean that they are not there. A bigger sample size can give more insight into whether or not there are significant differences between prototype one and two.

7.1.4 Survey II Results Compared to the Intended User Experience

To verify whether the tool has met the user requirements and facilitates the intended user experience the results of the second survey have been put against the user needs analysis done in Chapter 4 - Ideation.

The first group that was identified are the actual users of the tool. Two main aspects were of importance to this group.

The tool must be useful

The survey results showed that the majority of the respondents found the tool to be useful and easy to use. Besides that all five respondents indicated that the tool could help them learn more about software architecture. Taking all these answers into consideration this aspect of the intended user needs seemed to be fulfilled.

The tool must be fun to use

In the ideation fun to use was described as the goal of developing a tool that motivates users. The most important factors were therefore engaging and manageable content. The results from the survey showed that the majority of the respondents found the tool to be engaging. The results also indicated that the exercises were helpful in reaching this goal. Aesthetically the results were mixed. Some respondents liked the design while others did not. When it comes to the manageable factor the second prototype seemed to be better as multiple respondents liked the way how videos were directly followed with exercises to test the obtained knowledge. Taking all these results from the survey into consideration it seems that overall this user experience goal was reached. There is however still room for improvement, especially when it comes to the design of the tool.

7.2 Reflection

As part of the Creative Technology design process there is a reflection on this process in the evaluation phase. This section will reflect upon the implicit decisions in the design process and the execution of this process compared to my own standards.

In the process of developing the first prototype the initial goal was to develop a complete tool including thirteen different chapters. This was, looking back, way too much given the timeframe of this project. Therefore the decision was made to reduce the content of the tool to only one chapter which included four sections with one video each. Reflecting on this choice now it could have been better to reduce the content of the tool to just one single section of a chapter. This would result in a smaller workload and allow for more iterations. Also the quality of the prototypes would be higher as there is more time to focus on the realization.

Another aspect that I want to reflect upon are the transitions between certain phases of the design process. The diagram of the design process as shown in Figure 3.1 might suggest that the different phases do not overlap while in practice they do. Especially in the case of this project the specification and realization phase overlapped quite extensively. In my opinion the transition between phases should really be adapted to the type of product that is being developed.

Chapter 8 - Conclusion & Future Work

In this chapter the final conclusions regarding the development of the tool and possible options for future work will be discussed.

8.1 Conclusions

In this graduation project a start for the full development of an online educational tool that teaches software architecture was made. Given the obtained feedback from the user testing that was done it can be assumed that there is a good potential for the success of such a tool. This tool can provide effective learning while keeping the users engaged. It should be noted however that these conclusions are only preliminary. The groups on which the user tests were performed consisted of only five people. Besides that, real world applications are always different. Real world practice has to decide whether or not this tool will be of use.

8.2 Future Work

There are two main suggestions for future work that will be emphasized. These suggestions focus on potential improvement and extension of the tool.

First the tool can be improved by making the videos and exercises more engaging. There is still plenty of room to experiment with different types of interactive exercises to make them more interesting and engaging. Also the videos are still limited and were not extensively evaluated in the two surveys that were performed. When testing different approaches of video production style for example these videos could be made more engaging.

The second suggestion to improve the current tool is to extend the tool with an element that focuses on teaching soft skills within software architecture. One option that can be explored is the development of an online platform where users can discuss their architectural designs as described in the second subconcept in Chapter 4 - Ideation. Ideally a concept for this extension element will still allow the tool to be used by individuals at whatever time they wish. This way the accessibility of the tool is preserved.

Appendix A Video Scripts and Slides

Scripts

Chapter 1.1, what software architecture is and what it isn't.

Next slide

To understand what software architecture is and isn't we can take a look at the definition as presented by L. Bass. This definition is as follows:

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

This definition does not state anything about early or major design decisions like other definitions tend to do. The reason for this is that not all architectural decisions are made early on and one does not always know which decisions are major and which are not. Structures are more easily identifiable and therefore form the basis of this definition.

Next slide

A structure can be seen as a set of elements held together by a relation. A software system usually consists of many different structures. Together they form the architecture of the system. There are three types of architectural structures that can be identified and play a role in the design, documentation and analysis of architectures.

Next Slide

First the module structure. A module refers to different implementation units of a software system. For example the database or the user interface. Often these different modules are implemented by different teams. In large projects modules can become really big. They are then separated in different parts. This is called a module decomposition structure. Other types of module structures are class diagrams for example.

Next Slide

The second category of structures are dynamic structures. These structures focus on how elements interact with each other at runtime. When systems are build with a set of services for example then these structures describe how these services interact with each other and the infrastructure around them. Runtime structures are usually called component and connector structures. An example of such a structure is a server client diagram.

Next Slide

The third kind of structure is the allocation structure. These structures describe how the software structures are embedded in the systems' organization for example. They described how components are deployed to different hardware for execution.

Besides these three there are more types of structures to be found within software systems. Though they are only architectural if they support reasoning about the system or it's properties.

Next Slide

Another trait of Software architecture is that it is an abstract description of a system. This means that certain details about elements within a system are not described as they are not relevant for the architectural description of a system. To reduce complexity of the architecture it is necessary to omit these details.

All software systems can be described in a certain software architecture. The complexity can differ but they all comprise elements and relations among them. This architecture is not always known though. Therefore it is important to document the different structures within a software system.

When documenting an architecture it is important to include the behaviour of elements. One might imagine the behaviour of certain elements based on a box and line drawing, but it's important to explicitly specify how these elements interact with each other.

And finally not all architectures are good architectures. Whether or not an architecture is good or bad also depends on the requirements and needs of a system. It's therefore important to design and evaluate a good architecture that fits the system.

Reading Script
Chapter 1.2 Architectural Structures and Views

Next Slide

The terms structure and view are used when discussing architecture representation. In the book by L. Bass these terms are described as follows.

A view is a representation of a coherent set of architectural elements, as written by and read by system stakeholders. It consists of a representation of a set of elements and the relations among them.
A structure is the set of elements itself, as they exist in software or hardware.

In short the view is actually a representation of a structure. A view can be seen as the documentation of certain architectural structures. They are presented in certain templates with a specific notation used by the stakeholders for example.

Next Slide

As stated in section 1.1 there are three categories of structures. The module, component and connector and allocation structure. These types of structures all correspond to different kinds of architectural decisions.

Let's start with the module structure. As stated earlier module structures consist of different implementation units. For example classes or different layers in a program. These modules all represent a static way of describing the system. They are assigned different tasks they are responsible for. As presented in the book by L. Bass the following design questions can be answered.

- *What is the primary functional responsibility assigned to each module?*
- *What other software elements is a module allowed to use?*
- *What other software does it actually use and depend on?*
- *What modules are related to other modules by generalization or specialization (i.e. inheritance) relationships?*

Because module structures describe how tasks are divided in different implementation units in a system, a module view is excellent to reason about the modifiability of a system.

Next Slide

Component-and-connector structures define how elements in the system behave and interact with each other at runtime. As stated before these elements are runtime components like a server or client for example. They have connectors to interact with each other. Component-and-connector structures help us answer the following questions.

- *What are the major executing components and how do they interact at runtime?*
- *What are the major shared data stores?*
- *Which parts of the system are replicated?*
- *How does data progress through the system?*
- *What parts of the system can run in parallel?*
- *Can the system's structure change as it executes and, if so, how?*

Component-and-connector views are important to answer questions about the software's performance, security, availability and more.

Next Slide

Allocation structures embody information about the relation of the system to its non software environment. These structures describe the relationship of elements within the system to one or more external environments. An allocation view helps us answer the following question about an allocation structure.

- *What processor does each software element execute on?*
- *In what directories or files is each element stored during development, testing and system building?*
- *What is the assignment of each software element to development teams?*

All the different views of these structures help determine the quality attributes of the system. For example the module structure tells something about how easy it would be to extend the system. These structures can actually be designed to improve these certain quality attributes like modifiability as an example here.

Next Slide

Now we will look at a list of useful module structures. There is the decomposition structure where the units are modules that are related to each other by a submodule type of relation. This is the decomposition of modules into smaller submodules. These submodules have a specific task within the system and are often described as segments or subsystems.

The use structure describes as it says how modules are related to each other based on if they have a use relation. This relation is defined as a unit of software that cannot work correctly without the presence of the correctly functioning related unit.

The layer structure describes modules as layers. Each layer has a set of services. In a strictly layered structure layers are only allowed to use layers that are directly below them.

In the Class or generalization structure module units are called classes. These structures describe relations among classes in two forms. First the inherits form and second the instance of form. These relations describe how classes are reused and functionality is added.

And last is the Data model. This model describes how certain static data is structured in a system. For example a banking application that holds information about the number of accounts one can have.

Next Slide

Here you can see a list of useful Component and Connector structures or C&C structures in short. These structures deal with the dynamic aspects of a running system. They show how the components and connectors are hooked together as an attachment type of relation.

First the service structure. These consist of services that work together by some sort of service coordination mechanism. These structures are typically useful when working with components that are developed anonymously elsewhere for example.

Concurrency structures focus on the opportunities for parallelism and identifying possible problems in this parallelism. The components are usually threads which can run simultaneously. These concurrency structures are therefore important to identify where these threads might interfere with each other where they were not supposed to.

Next Slide

Now we will look at some allocation structures which describe how the module and component and connector structures are placed in their non software environment. For example hardware components or development teams assigned to specific software modules.

The first useful allocation structure here is the deployment structure. This structure shows how software is assigned to certain hardware processing and communication elements. The type of relation is a allocated to and migrates to relation. These describe on what physical unit software is placed, the allocation, and where software is moved if it is dynamic, the migration.

The implementation structure describes how software elements are mapped to the file structure in the system's development, integration or configuration control environments.

The work assignment structure defines how different software modules are assigned to different teams. This structure will be really important for the management and the architect to see who makes decisions for which module and to make sure that they are capable. This structure is also really important for good communication among teams.

Next Slide

Check the Useful Architectural Structures table from the book by L. Bass for a summary of all the structures that were briefly covered in this video. These structures all use some kind of elements. These elements actually overlap between different structures. A good example is a server client system as can be seen here. On the left there is a module view and on the right there is a component-and-connector view. They both show the same elements except one shows the client as a single element and the other has ten different clients communicating to the server. Both types of structure views are useful, they only serve different purposes. For example the view on the right can be used to identify performance issues.

Next Slide

In the end you probably don't have to document all these types of structures. For example when your software is deployed on a single processing unit it is not relevant to make a deployment

diagram. Designing and documenting a structure should only be done if it's beneficial for the maintenance or cost or maybe because it's mandatory by your teacher.

Reading Script
1.3 Architectural Patterns
Next Slide

As discussed in the previous video architectural elements are composed in ways to solve a certain architectural problem. Sometimes these solutions are very useful and are therefore documented to reuse them. These compositions of architectural elements are then called architectural patterns. A definition by Wikipedia is as follows: An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context.

Next Slide

These patterns describe very precisely what type of elements are used and how they interact with each other. They can be characterized by the type of architectural elements they use. Let's go through some examples of architectural patterns.

Next Slide

For instance a common module type pattern is the layered pattern. A uses relation can be found in a layered pattern, only here it is one directional. In a layer pattern only services of the layer directly below it can be used. In practice this structural restriction doesn't always hold. There are actually many variations in how this pattern is implemented.

Next Slide

If you look at component and connector type patterns you have the shared-data (or repository) pattern for example. The repository usually takes the form of a database. The managing of this database is often done by an SQL protocol for example. In this example different applications access the same database.

Next Slide

In a client-server pattern, which is also a component-and-connector type of pattern the elements consists of clients and servers. They interact with each other by an internet socket connection. Together they make the system work.

Next Slide

When looking at allocation patterns you have the multi-tier pattern for example. This patterns describes how different elements of the system are hosted by different hardware components as can be seen in the example.

Next Slide

The competence center and platform are also allocation type patterns. They describe the software system's work assignment. In the competence center work is split based on skill and expertise of different teams. For example letting the user experience designers work on the user interface. In the platform pattern the work is split based on components that need to be made. One team then uses the components of the other team for example.

Next Slide

There are many more types of useful patterns that we can discuss, for instance the ones listed here. We will not do that right now. If you are interested to learn more about architectural patterns take a look at chapter 13 of the book Software Architecture in Practice. This chapter will go much more in depth on this topic.

Reading Script
1.4 What Makes a "Good" Architecture?

Next Slide

A particular architecture is not good or bad by itself. It really depends on the purpose of the architecture. One architectural solution might be great for system A, but does not fit system B. For example it would not make sense to build a highly modifiable system for a throwaway prototype. Systems require different quality attributes. We can actually evaluate systems based on these quality attributes. Despite that the quality of an architecture highly depends on it's purpose, there are a couple of rules that should be followed when designing most architectures. These recommendations have been split up in two clusters, the process recommendations and the product recommendations.

Next Slide

Lets first go through the process recommendations. These recommendations are concerned about the way the architecture is designed. They form the guidelines to follow in the process of designing an architecture.

Next Slide

To start it is recommended to let the architecture be designed by a single or small group of people with a clear architectural leader. This way there is technical consistency and conceptual integrity throughout the design. You also want a strong connection between the architect and the development teams to make sure everybody is doing their part on the bigger thing.

Next Slide

Second the architecture should be designed based on a prioritized list of quality attributes. This way you will find the design trade-offs that come about.

Next Slide

You also want to document your architecture using different views that are relevant for the system. These views should address the most important aspects for the stakeholders.

Next Slide

Once you have considered the quality attributes in the design of your architecture you should also evaluate the system for its ability to meet these quality attributes. You should not wait for this till the end but do it early and repeat as appropriate.

Next Slide

And last the architecture should lend itself for incremental implementation. This way you avoid having to implement everything at once. You can also find potential problems earlier.

Next Slide

Now that we have looked at the process recommendations we take a look at the product recommendations. These are recommendations concerning how the system or product should look or be structured.

Next Slide

First the architecture should feature modules that are well defined. It should be clear what the interface of a certain module is so it is clear how other modules use or are used by the certain module.

Next Slide

Another product recommendation is using well known architectural patterns to achieve the system's quality attributes. The patterns used should therefore fit these attributes well.

Next Slide

Next, the architecture should not depend on a particular version of a commercial product. One can imagine that such a dependency can have various implications. If there is no way around it make sure that switching versions is easy and not very expensive.

Next Slide

Then, modules that produce and consume data should be separated. Often either the production or consumption of data changes. If these are separate it will be easier to modify.

Next Slide

Next, don't expect a one to one correspondence between modules and components. You can see this in a multithreading application for example where each thread can use services from several components.

Next Slide

Up next, every process should be written so that its assignment to a specific processor can be easily changed. This means the process is not dependent on the specific processor it was written for.

Next Slide

And now, the architecture features a small number of ways for components to interact. This reduces the complexity and makes the architecture more understandable and easier to implement. It will therefore also be more reliable and easier to modify.

Next Slide

And last the architecture should contain a specific set of resource contention areas. This means that for example teams have guidelines in how they can use the network to reduce network traffic issues for instance.

Next Slide

We have come to the end of chapter 1. It is now time to make the exercises and see what you have learned. You can use the book software architecture in practice for extra information regarding the covered topics.

Slides

Chapter 1.1

1.1 What Software Architecture Is and What It Isn't

•••

Based on Chapter 1.1 of the book Software Architecture in Practice by L. Bass et al.

A Definition for Software Architecture

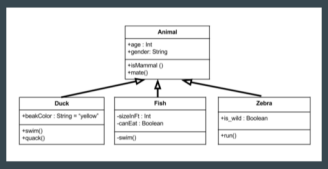
The Software Architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

Architecture Is a Set of Software Structures

- A set of elements held together by a relation
- Software structures together form the architecture of the system
- Three categories of structures can be identified

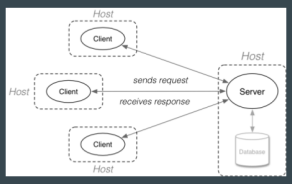
Module Structures

- Partition systems into different implementation units with their own specific computational responsibilities.



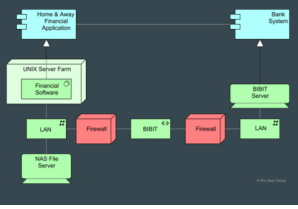
Dynamic Structures

- Describes how different elements interact with each other at runtime. They are called component-and-connector (C&C) structures.



Allocation Structures

- Mapping software structures to the system's organizational, developmental, installation and execution environments.



Software Architecture is abstract

All Software Systems have an Architecture

Software Architecture Describes Behaviour of Elements

Not All Architecture Is Good Architecture

Chapter 1.2

1.2 Architectural Structures and Views

•••

Based on Chapter 1.2 of the book Software Architecture in Practice by L. Bass et al.

Structures and Views

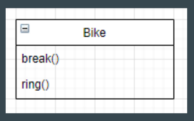
- A view is a representation of a coherent set of architectural elements, as written by and read by system stakeholders. It consists of a representation of a set of elements and the relations among them.
- A structure is the set of elements itself, as they exist in software or hardware.

```
class Bike {

    public void break(){
        //break bike
    }

    public void ring(){
        //ring bike bell
    }
}
```

The Structure



The View

Module Structure

- What is the primary functional responsibility assigned to each module?
- What other software elements is a module allowed to use?
- What other software does it actually use and depend on?
- What modules are related to other modules by generalization or specialization (i.e. inheritance) relationships?

Component-and-Connector Structures

- What are the major executing components and how do they interact at runtime?
- What are the major shared data stores?
- Which parts of the system are replicated?
- How does data process through the system?
- What parts of the system can run in parallel?
- Can the system's structure change as it executes and, if so, how?

Allocation Structures

- What processor does each software element execute on?
- In what directories or files is each element stored during development, testing and system building?
- What is the assignment of each software element to development teams?

Useful Module Structures

- Decomposition structure
- Uses structure
- Layer structure
- Class structure
- Data model

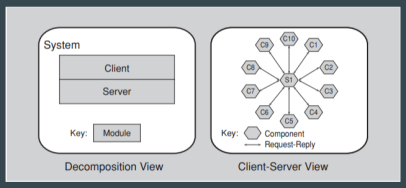
Useful Component-and-Connector (C&C) Structures

- Service structure
- Concurrency structure

Useful Allocation Structures

- Deployment structures
- Implementation structures
- Work assignment structures

Relating Structures to Each Other



The views of a client-server system
Source: Software Architecture in Practice by L. Bass et al. (Figure 1.2)

Which Structures to Design and Document?

Fewer is Better

Chapter 1.3

1.3 Architectural Patterns

•••

Based on Chapter 1.3 of the book Software Architecture in Practice by L. Bass et al.

What are Architectural Patterns?

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context.

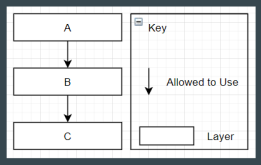
Source: https://en.wikipedia.org/wiki/Architectural_pattern

Examples of Architectural Patterns

- Layered pattern
- Shared-data pattern
- Client-server pattern
- Multi-tier pattern
- Competence center and platform

Examples of Architectural Patterns

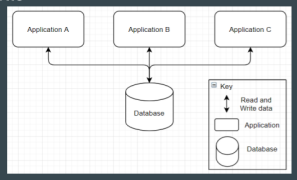
- Layered pattern
- Shared-data pattern
- Client-server pattern
- Multi-tier pattern
- Competence center and platform



Strictly Layered pattern

Examples of Architectural Patterns

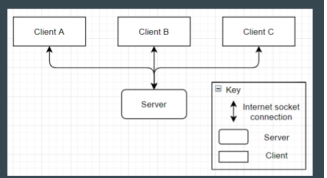
- Layered pattern
- Shared-data pattern
- Client-server pattern
- Multi-tier pattern
- Competence center and platform



Shared-data pattern with three Applications

Examples of Architectural Patterns

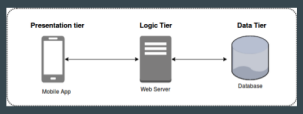
- Layered pattern
- Shared-data pattern
- Client-server pattern
- Multi-tier pattern
- Competence center and platform



Client-server pattern with three clients and one server

Examples of Architectural Patterns

- Layered pattern
- Shared-data pattern
- Client-server pattern
- Multi-tier pattern
- Competence center and platform



Simple three tier structure.
Source: https://dzone.com/articles/serverless-multi-tier-architecture-on-aws

Examples of Architectural Patterns

- Layered pattern
- Shared-data pattern
- Client-server pattern
- Multi-tier pattern
- Competence center and platform

More Architectural Patterns

- Broker Pattern
- Model View Controller (MVC) Pattern
- Pipe-and-Filter Pattern
- Peer-to-Peer Pattern
- Service-Oriented Architecture Pattern
- Publish-Subscribe Pattern
- Map-Reduce Pattern

Take a look at Chapter 13 of the book Software Architecture in Practice by L. Bass which discusses architectural patterns deeper.

Chapter 1.4

1.4 What Makes a "Good" Architecture?

•••

Based on Chapter 1.4 of the book Software Architecture in Practice by L. Bass et al.

It Depends On The Purpose

- An architecture is not inherently good or bad
- We can evaluate the fit of an architecture based on the quality attributes

Process Recommendations for Good Architecture

- Architecture is made by single or small group of architects
- Architecture is based on a prioritized list of quality attributes
- Architecture is documented using views
- Architecture is evaluated for the prioritized list of quality attributes
- Architecture lends itself for incremental implementation

Process Recommendations for Good Architecture

- Architecture is made by single or small group of architects
- Architecture is based on a prioritized list of quality attributes
- Architecture is documented using views
- Architecture is evaluated for the prioritized list of quality attributes
- Architecture lends itself for incremental implementation

Process Recommendations for Good Architecture

- Architecture is made by single or small group of architects
- Architecture is based on a prioritized list of quality attributes
- Architecture is documented using views
- Architecture is evaluated for the prioritized list of quality attributes
- Architecture lends itself for incremental implementation

Process Recommendations for Good Architecture

- Architecture is made by single or small group of architects
- Architecture is based on a prioritized list of quality attributes
- Architecture is documented using views
- Architecture is evaluated for the prioritized list of quality attributes
- Architecture lends itself for incremental implementation

Process Recommendations for Good Architecture

- Architecture is made by single or small group of architects
- Architecture is based on a prioritized list of quality attributes
- Architecture is documented using views
- Architecture is evaluated for the prioritized list of quality attributes
- Architecture lends itself for incremental implementation

Process Recommendations for Good Architecture

- Architecture is made by single or small group of architects
- Architecture is based on a prioritized list of quality attributes
- Architecture is documented using views
- Architecture is evaluated for the prioritized list of quality attributes
- Architecture lends itself for incremental implementation

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

Product Recommendations for Good Architecture

- The architecture should have well defined modules
- Quality attributes should be achieved using well-known architectural patterns
- The architecture should not depend on a particular version of a commercial product
- Modules that produce and consume data should be separated
- Don't expect a one to one correspondence between modules and components
- Every process should be written so that its assignment to a specific processor can be easily changed
- The architecture features a small number of ways for components to interact
- The architecture should contain a specific set of resource contention areas

End Chapter 1

- Make the exercises
- Use the book Software Architecture in Practice for extra information

Appendix B HTML, CSS and JavaScript Code

HTML

Index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/style.css">
</head>
<body>

<header>
	<a href="index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
	<div class="container">
		<h1>Introduction</h1>
		<p>
			In this educational tool one can improve their knowledge and skills on software architecture.
			This tool was created based on the book <a href="http://jz81.github.io/course/sa/Software%20Architecture%20in%20Practice%20(3rd).pdf">Software Architecture in Practice by L. Bass</a>. It does not cover all the chapters presented in the book. The emphasis of this tool will be on the documentation and communication about software architecture in general. Therefore this tool is great for people who want to understand and discuss the design choices in software architecture that are being made or become better at explaining their own architectural designs.
		</p>
		<p>
			This educational tool was made by Arno de Vries as part of the Bachelor Graduation Project at the University of Twente.
		</p>
		<div><img src="images/software architecture illustration.jpg" alt="Software Architecture Illustration"></div>
		<p>Image from freepik.com <a href="https://www.freepik.com/free-vector/software-engineer-concept-illustration_9936439.htm#page=2&query=software%20engineer&position=0">link</a></p>
		<div class="button-container">
			<a href="Chapter 1/chapter1.html" title="Chapter 1 What is Software Architecture?"><span class="next-button">Start Chapter 1 ></span></a>
		</div>
```

```
	</div>
</main>
<footer>
	Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

Index-old.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/style.css">
</head>
<body>

<header>
	Educational Tool on Software Architecture

</header>
<main>
	<div class="container">
	<section class="introduction">
		<div class="introduction-text">
			<h1>Introduction</h1>
			<p>
				In this educational tool one can improve their knowledge and skills on software architecture.
				This tool was created based on the book <a href="http://jz81.github.io/course/sa/Software%20Architecture%20in%20Practice%20(3rd).pdf">Software Architecture in Practice by L. Bass</a>. It does not cover all the chapters presented in the book. The emphasis of this tool will be on the documentation and communication about software architecture in general. Therefore this tool is great for people who want to understand and discuss the design choices in software architecture that are being made or become better at explaining their own architectural designs.
			</p>
			<p>
				This educational tool was made by Arno de Vries as part of the Bachelor Graduation Project at the University of Twente.
			</p>
		</div>
```

```
		<div class="introduction-image">
			<img src="images/software architecture illustration.jpg" alt="Software
Architecture Illustration">
		</div>
	</section>
	<section class="chapter1">
		<div class="chapter-intro">
			<h1>Ch. 1 What is Software Architecture?</h1>
			<p>
				To learn skills in and knowledge about software architecture it is
important to first understand what software architecture is. This chapter will
				cover why there is a need for software architecture.
			</p>
		</div>
		<div class="chapter-section">
			<h2>1.1 What software archicture is and what it is not</h2>
			<p>To learn skills and knowledge about software architecture it is
important to understand what software architecture is and what it is not. In this
			first video a definition and its assumptions will be discussed.</p>
			<div class="video-container"><iframe
src="https://www.youtube.com/embed/EIMdCqeqbCg"></iframe></div>
			<h3>1. Anser True or False to the following statements on software
architecture. This question is about chapter 1.1</h3>
				<p>a. Different types of software structures together form
the architecture of a system.</p>
				<p>b. A software architecture is a very detailed description
of a system including internal implementation details of elements in a system.</p>
				<p>c. Every software system has a software architecture.</p>
				<p>d. The behaviour of elements in software is also part of
the architecture.</p>
				<p>e. Picking a well known architecture at random and
building the system from that architecture is an effective way to develop a new system.</p>
				<h3>2. Decide based on the diagram shown if it is displaying a
Module, Component-and-connector or Allocation structure. This question is about chapter 1.1</h3>
				<p>a:</p>
				<img src="images/Exercises2a.png" alt="Exercises diagram
2a">
				<p>b:</p>
				<img src="images/Exercises2b.png" alt="Exercises diagram
2b">
				<p>c:</p>
				<img src="images/Exercises2c.png" alt="Exercises diagram
2c">
		</div>
		<div class="chapter-section">
```

<h2>1.2 Architectural Structures and Views</h2>
It is important to distinguish between the structure and the view. In this video a definition for both will be discussed. Besides that the purpose
of these different types of structures will be explained.
<iframe src="https://www.youtube.com/embed/MxqFYFRf5fc"></iframe>
<img src="images/SoftwareStructuresTable.PNG" alt="Useful Software Structures">
Source: Table 1.1 in the book Software Architecture in Practice by L. Bass, Useful Architectural Structures
<h3>3. Decide if the shown trait belongs to either a structure or a view. This question is about chapter 1.2</h3>
a. A diagram that represents a certain relation between architectural elements in a system.
b. A set of elements in a system.
c. A file with code that is responsible for the user interface.
<h3>4. Decide based on the question about a system which structure contains the answer. This question is about chapter 1.2</h3>
a. What is the assignment of each software element to development teams?
b. How does data process through the system?
c. What other software elements is a module allowed to use?

<h2>1.3 Architectural Patterns</h2>
Architectural patterns form an important aspect of software architecture. In this video the concept architectural pattern as well as several examples will be discussed.
<iframe src="https://www.youtube.com/embed/8Qmq8nmUUeI"></iframe>
<h3>5. A diagram example of an architectural pattern is shown. Decide what type of pattern it is(module, C&C or allocation) and the specific pattern name. This question is about chapter 1.3</h3>
a:
<img src="images/Exercises5a.png" alt="Exercises diagram 5a">
b:
<img src="images/Exercises5b.png" alt="Exercises diagram 5b">
c:
<img src="images/Exercises5c.png" alt="Exercises diagram 5c">

```
<div class="chapter-section">
	<h2>1.4 What makes a "Good" Architecture?</h2>
	<p>In this last video of chapter 1 we try to give an answer to the question what makes a good architecture. Several recommendations are given. </p>
	<div class="video-container"><iframe src="https://www.youtube.com/embed/5JOxq0WA_-o"></iframe></div>
	<h3>6. Different Recommendations are shown on screen. Determine if it’s a good or bad recommendation. This question is about chapter 1.4</h3>
		<p>The design of an architecture should be done in a big group to obtain as many ideas as possible.</p>
</div>
<div class="chapter-section">
	<h2>Answers</h2>
	<button onclick="showAnswers()">Show Anwers</button>
	<ol id="answers">
		<h3>1. True or false statements</h3>
			<p>a. True. An architecture of a system can be seen as a set of structures that together form the architecture of the system. With these structures one can reason about different aspects of the system.</p>
			<p>b. False. An architecture is an abstraction of a system. Architecture is about the public side of elements in a system. To help understand how elements interact. Internal implementation details are not architectural as they don’t tell anything about the interaction with other elements in the system.</p>
			<p>c. True. Though for some systems the architecture is very simple, they all have elements and relations among them that form the architecture. It is not always known though what the architecture looks like. Therefore it is important to document the architecture of the system for future references.</p>
			<p>d. True. It is actually a really important aspect of the architecture. One could imagine how a system would work based on a box and line drawing identifying the elements in the system, however it is better to document this and make it clear so people understand the behaviour of the elements in the system.</p>
			<p>e. False. Not all architectures are good architectures. It really depends on the needs of the system if an architecture is good or not. Therefore it is important to have good architectural design.</p>
		<h3>2. Structure types</h3>
			<p>a. Module structure as this diagram tells you something about the different implementation units and how they relate to each other on a building structure. Source: Figure 13.5 from Software Architecture in Practice by L. Bass, layered design with segmented layers.</p>
			<p>b. Component-and-connector structure as this diagram describes how different components are connected to each other. Here they can either read, write or read & write data to the database or headless program. Source Figure 13.13 from Software Architecture in Practice by L. Bass, The shared-data diagram of an enterprise access management system.</p>
```

c. Allocation structure. This diagram describes how different hardware components are responsible for the execution of certain elements within the system. Source: AWS Serverless Multi-Tier Architectures with Amazon API Gateway and AWS Lambda

<h3>3. Structure or View statements</h3>

a. View. A representation of a certain element in a diagram for example is called the view.

b. Structure. The actual elements in the system is the structure.

c. Structure. Actual code is part of the architectural elements in a system. Therefore this belongs more to the structure than to the view. The fact that it is responsible for the user interface does not change that.

<h3>4. Question to structure type connection</h3>

a. Allocation structure. Organizational questions about which team is responsible for which software elements is part of the allocation structures.

b. Component-and-connector structure. This question is about a runtime component and specifies how the data processes through a system focussing on the relation between elements when it comes to data. Therefore a component-and-connector structure is most suitable to answer this question.

c. Module structure. One might think that since this question address the module and what elements it may use that it therefore is answered by a component-and-connector structure. But here it is not about the actual relation they have, just that they have a relation. Therefore a use structure answers this question which is a module type structure.

<h3>5. Architectural patterns</h3>

a. Multi-tier pattern. Different computing components are grouped together in different execution structures. Source: Figure 13.15 in the book Software Architecture in Practice by L. Bass, A multi-tier view of the consumer website java EE application, which is part of the Adventure Builder System

b. Client-server pattern. Source: Figure 13.9 in the book Software Architecture in Practice by L. Bass, The client-server architecture of an ATM banking system.

c. Layer pattern. Source: Figure 13.2 in the book Software Architecture in Practice by L. Bass, A simple layer diagram, with a simple key answering the uses question.

<h3>6. Good or bad recommendation</h3>

Bad. You want to minimize the amount of people responsible for the architecture. This way you keep technical consisteny and conceptual integrity throughout the design.

</section>

<!-- unfinished

```
<section class="chapter2">
	<div class="chapter-intro">
		<h1>Ch. 2 Why should you learn Software Architecture?</h1>
		<p>

		</p>
	</div>
	<div class="chapter-section">
		<h2>1.1 Enhancing Communication among Stakeholders</h2>
		<p></p>
	</div>
	<div class="chapter-section">
		<h2>1.2 Other Important reasons for good Software Architecture</h2>
		<p></p>
	</div>
	<div class="chapter-section">
		<h2>1.3 Exercises</h2>
		<p></p>
	</div>
</section>
<section class="chapter3">
	<div class="chapter-intro">
		<h1>Ch. 3 The Different Contexts of Software Architecture</h1>
		<p>

		</p>
	</div>
	<div class="chapter-section">
		<h2>1.1 Architecture in a Technical Context</h2>
		<p></p>
	</div>
	<div class="chapter-section">
		<h2>1.2 Architecture in a Project Life-Cycle Context</h2>
		<p></p>
	</div>
	<div class="chapter-section">
		<h2>1.3 Architecture in a Business Context</h2>
		<p></p>
	</div>
	<div class="chapter-section">
		<h2>1.4 Architecture in a Professional Context</h2>
		<p></p>
	</div>
	<div class="chapter-section">
		<h2>1.5 Exercises</h2>
```

```
                </div>
        </section>
        -->
        </div>
</main>
<footer>
        Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>

<script>
function showAnswers() {
  var x = document.getElementById("answers");
  if (x.style.display === "none") {
    x.style.display = "block";
  } else {
    x.style.display = "none";
  }
}
</script>
</body>
</html>
```

End-tool.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/style.css">
</head>
<body>

<header>
        <a href="index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
        <div class="container">
                <div class="chapter-section">
                        <h1>End of the Educational Tool</h1>
                        <p>
```

```
                        This is the end of the educational tool for now. This first version
only covers one chapter. The plan however is to add more chapters to this tool and make it an
online resource that covers all the basics of software architecture.
                    </p>
                    <p>
                        Thank you for using the tool. Hopefully you learned something
                    </p>
                </div>
                <div class="button-container">
                    <a href="Chapter 1/chapter1-4.html" title='1.4 What makes a "Good"
Architecture?'><span class="back-button">< Chapter 1.4</span></a>
                    <a href="index.html" title="Chapter 1.1 What software architecture is and
what it is not"><span class="next-button">Main ></span></a>
                </div>
        </div>
</main>
<footer>
        Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

Chapter1.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="../css/style.css">
</head>
<body>

<header>
        <a href="../index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
        <div class="container">
                <div class="chapter-section">
                    <h1>Chapter 1 What is Software Architecture?</h1>
                    <p>
                        To learn skills in and knowledge about software architecture it is
important to first understand what software architecture is. This chapter will
                        cover why there is a need for software architecture.
                    </p>
                    <p>This chapter consists of the following subsections</p>
                    <ul>
                        <li><a href="chapter1-1.html">1.1 What Software Architecture Is and
What It Is Not</a></li>
                        <li><a href="chapter1-2.html">1.2 Architectural Structures and
Views</a></li>
```

```
                        <li><a href="chapter1-3.html">1.3 Architectural Patterns</a></li>
                        <li><a href="chapter1-4.html">1.4 What Makes A "Good"
Architecture?</a></li>
                </ul>
        </div>
        <div class="button-container">
                <a href="../index.html" title="Main Page"><span class="back-button">< Main
Page</span></a>
                <a href="chapter1-1.html" title="Chapter 1.1 What software architecture is
and what it is not"><span class="next-button">Chapter 1.1 ></span></a>
        </div>
    </div>
</main>
<footer>
    Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

Chapter1-1.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Educational Tool on Software Architecture</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="../css/style.css">
    <script defer src="../javascript/script1-1.js"></script>
</head>
<body>

<header>
    <a href="../index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
    <div class="container">
        <div class="chapter-section">
            <h1>Chapter 1.1 What software architecture is and what it is not</h1>
            <p>To learn skills and knowledge about software architecture it is
important to understand what software architecture is and what it is not. In this
            first video a definition and its assumptions will be discussed.</p>
            <div class="video-container"><iframe
src="https://www.youtube.com/embed/EIMdCqeqbCg"></iframe></div>
        </div>
        <div class="chapter-section">
            <h2>Summary</h2>
            <ul>
                <li>A definition for software architecture as presented by L.Bass
in Software Architecture in Practice is as follows: <br>
                    <i>The software architecture of a system is the set of
structures needed to reason about the system, which comprise software elements, relations among
them, and properties of both.</i></li>
                <li>There are three types of architectural structures that can be
identified and play a role in the design, documentation and analysis of architectures. These are
module, component&connector and allocation structures. (Chapter 1.2 will go more into depth about
these structures)</li>
                <ul>
                    <li>Module structures tell something about the different
implementation units within a system.</li>
```

```
                                        <li>Component&Connector structures describe how elements
interact with each other at runtime.</li>
                                        <li>Allocation structures are about how software is embedded
in the systems environment.</li>
                                </ul>
                        </ul>
                </div>
                <div class="chapter-section">
                        <h2>Exercises</h2>
                        <h3>1. Anser True or False to the following statements on software
architecture.</h3>
                        <form name="exercise1">
                                <p>a. Different types of software structures together form the
architecture of a system.</p>
                                <label class="btn">
                                        <input type="radio" name="question1a" value="True">
                                        <span id="1a-true-btn">True</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1a" value="False">
                                        <span id="1a-false-btn">False</span>
                                </label>
                                <p class=hidden-answer1><i>True. An architecture of a system can be
seen as a set of structures that together form the architecture of the system. With these
structures one can reason about different aspects of the system.</i></p>
                                <p>b. A software architecture is a very detailed description of a
system including internal implementation details of elements in a system.</p>
                                <label class="btn">
                                        <input type="radio" name="question1b" value="True">
                                        <span id="1b-true-btn">True</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1b" value="False">
                                        <span id="1b-false-btn">False</span>
                                </label>
                                <p class=hidden-answer1><i>False. An architecture is an abstraction
of a system. Architecture is about the public side of elements in a system. To help understand
how elements interact. Internal implementation details are not architectural as they don’t tell
anything about the interaction with other elements in the system.</i></p>
                                <p>c. Every software system has a software architecture.</p>
                                <label class="btn">
                                        <input type="radio" name="question1c" value="True">
                                        <span id="1c-true-btn">True</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1c" value="False">
                                        <span id="1c-false-btn">False</span>
                                </label>
                                <p class=hidden-answer1><i>True. Though for some systems the
architecture is very simple, they all have elements and relations among them that form the
architecture. It is not always known though what the architecture looks like. Therefore it is
important to document the architecture of the system for future references.</i></p>
                                <p>d. The behaviour of elements in software is also part of the
architecture.</p>
                                <label class="btn">
                                        <input type="radio" name="question1d" value="True">
                                        <span id="1d-true-btn">True</span>
                                </label>
```

```
                                <label class="btn">
                                        <input type="radio" name="question1d" value="False">
                                        <span id="1d-false-btn">False</span>
                                </label>
                                <p class=hidden-answer1><i>True. It is actually a really important
aspect of the architecture. One could imagine how a system would work based on a box and line
drawing identifying the elements in the system, however it is better to document this and make it
clear so people understand the behaviour of the elements in the system.</i></p>
                                <p>e. Picking a well known architecture at random and building the
system from that architecture is an effective way to develop a new system.</p>
                                <label class="btn">
                                        <input type="radio" name="question1e" value="True">
                                        <span id="1e-true-btn">True</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1e" value="False">
                                        <span id="1e-false-btn">False</span>
                                </label>
                                <p class=hidden-answer1><i>False. Not all architectures are good
architectures. It really depends on the needs of the system if an architecture is good or not.
Therefore it is important to have good architectural design.</i></p>
                        </form>
                        <label class="btn">
                                <input type="button" name="" value="Check Answers"
onclick="checkAnswers1()">
                                <span>Check Answers</span>
                        </label>

                        <h3>2. Decide based on the diagram shown if it is displaying a Module,
Component-and-connector or Allocation structure.</h3>
                        <form name="exercise2">
                                <p>a. diagram source: Software Architecture in Practice by L. Bass
et al.</p>
                                <img src="../images/Exercises2a.png" alt="Exercises diagram 2a">
                                <label class="btn">
                                        <input type="radio" name="question2a" value="module">
                                        <span id="2a-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="compcon">
                                        <span id="2a-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="allocation">
                                        <span id="2a-allocation-btn">Allocation Structure</span>
                                </label>
                                <p class=hidden-answer2><i>Module structure as this diagram tells
you something about the different implementation units and how they relate to each other on a
building structure. Source: Figure 13.5 from Software Architecture in Practice by L. Bass,
layered design with segmented layers.</i></p>
                                <p>b. diagram source: Software Architecture in Practice by L. Bass
et al.</p>
                                <img src="../images/Exercises2b.png" alt="Exercises diagram 2b">
                                <label class="btn">
                                        <input type="radio" name="question2b" value="module">
                                        <span id="2b-module-btn">Module Structure</span>
                                </label>
```

```
                                <label class="btn">
                                        <input type="radio" name="question2b" value="compcon">
                                        <span id="2b-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2b" value="allocation">
                                        <span id="2b-allocation-btn">Allocation Structure</span>
                                </label>
                                <p class=hidden-answer2><i>Component-and-connector structure as
this diagram describes how different components are connected to each other. Here they can either
read, write or read & write data to the database or headless program. Source Figure 13.13 from
Software Architecture in Practice by L. Bass, The shared-data diagram of an enterprise access
management system.</i></p>
                                <p>c: diagram source: docs.aws.amazon.com (full link in answer)</p>
                                <img src="../images/Exercises2c.png" alt="Exercises diagram 2c">
                                <label class="btn">
                                        <input type="radio" name="question2c" value="module">
                                        <span id="2c-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="compcon">
                                        <span id="2c-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="allocation">
                                        <span id="2c-allocation-btn">Allocation Structure</span>
                                </label>
                                <p class=hidden-answer2><i>Allocation structure. This diagram
describes how different hardware components are responsible for the execution of certain elements
within the system. Source: <a
href="https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gate
way-lambda/three-tier-architecture-overview.html">AWS Serverless Multi-Tier Architectures with
Amazon API Gateway and AWS Lambda</a></i></p>
                        </form>
                        <label class="btn">
                                <input type="button" name="" value="Check Answers"
onclick="checkAnswers2()">
                                <span>Check Answers</span>
                        </label>
                </div>
                <div class="button-container">
                        <a href="chapter1.html" title="Chapter 1 Main Page"><span
class="back-button">< Chapter 1 Main</span></a>
                        <a href="chapter1-2.html" title="Chapter 1.2 Architectural structures and
views"><span class="next-button">Chapter 1.2 ></span></a>
                </div>
        </div>
</main>
<footer>
        Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

Chapter1-2.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="../css/style.css">
<script defer src="../javascript/script1-2.js"></script>
</head>
<body>

<header>
        <a href="../index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
        <div class="container">
                <div class="chapter-section">
                        <h1>1.2 Architectural Structures and Views</h1>
                        <p>It is important to distinguish between the structure and the view. In
this video a definition for both will be discussed. Besides that the purpose of these different
types of structures will be explained.</p>
                        <div class="video-container"><iframe
src="https://www.youtube.com/embed/MxqFYFRf5fc"></iframe></div>
                        <img src="../images/SoftwareStructuresTable.PNG" alt="Useful Software
Structures">
                        <p>Source: Table 1.1 in the book Software Architecture in Practice by L.
Bass, Useful Architectural Structures</p>
                </div>
                <div class="chapter-section">
                        <h2>Summary</h2>
                        <ul>
                                <li>Understanding the difference between a structures and views is
important when discussing software architecture. Definitions are as follows: <br>
                                        <i>A view is a representation of a coherent set of
architectural elements, as written by and read by system stakeholders. It consists of a
representation of a set of elements and the relations among them.</i> <br>
                                        <i>A structure is the set of elements itself, as they exist
in software or hardware.</i></li>
                                <li>Module structures can help answer questions about how the
system is structured statically. Describing the tasks each module is responsible for.Useful
module structures are for example the decomposition, uses, layer or class structure.</li>
                                <li>Component-and-Connector structures define how elements in the
system behave and interact with each other at runtime. These structures can help answer questions
about the software's performance, security, availability and more. Useful C&C structures are for
example the service and concurrency structure.</li>
                                <li>Allocation structures embody information about the relation of
the system to its non software environment. These structures help us answer question about the
external environment of the software. Useful allocation structures are for example the
deployment, implementation and work assignment structure.</li>
                                <li>Elements of different structures in a system can often
overlap.</li>
                        </ul>
                </div>
                <div class="chapter-section">
                        <h2>Exercises</h2>
                        <h3>1. Decide if the shown trait belongs to either a structure or a
view.</h3>
                        <form name="exercise1">
```

```
                                <p>a. A diagram that represents a certain relation between
architectural elements in a system.</p>
                                <label class="btn">
                                        <input type="radio" name="question1a" value="Structure">
                                        <span id="1a-structure-btn">Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1a" value="View">
                                        <span id="1a-view-btn">View</span>
                                </label>
                                <p class=hidden-answer1><i>View. A representation of a certain
element in a diagram for example is called the view.</i></p>
                                <p>b. A set of elements in a system.</p>
                                <label class="btn">
                                        <input type="radio" name="question1b" value="Structure">
                                        <span id="1b-structure-btn">Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1b" value="View">
                                        <span id="1b-view-btn">View</span>
                                </label>
                                <p class=hidden-answer1><i>Structure. The actual elements in the
system is the structure.</i></p>
                                <p>c. A file with code that is responsible for the user
interface.</p>
                                <label class="btn">
                                        <input type="radio" name="question1c" value="Structure">
                                        <span id="1c-structure-btn">Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1c" value="View">
                                        <span id="1c-view-btn">View</span>
                                </label>
                                <p class=hidden-answer1><i>Structure. Actual code is part of the
architectural elements in a system. Therefore this belongs more to the structure than to the
view. The fact that it is responsible for the user interface does not change that.</i></p>
                        </form>
                        <label class="btn">
                                <input type="button" name="" value="Check Answers"
onclick="checkAnswers1()">
                                <span>Check Answers</span>
                        </label>
                        <h3>2. Decide based on the question about a system which structure contains
the answer.</h3>
                        <form name="exercise2">
                                <p>a. What is the assignment of each software element to
development teams?</p>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="module">
                                        <span id="2a-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="compcon">
                                        <span id="2a-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="allocation">
```

```
                                        <span id="2a-allocation-btn">Allocation Structure</span>
                                </label>
                                <p class=hidden-answer2><i>Allocation structure. Organizational
questions about which team is responsible for which software elements is part of the allocation
structures.</i></p>
                                <p>b. How does data process through the system?</p>
                                <label class="btn">
                                        <input type="radio" name="question2b" value="module">
                                        <span id="2b-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2b" value="compcon">
                                        <span id="2b-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2b" value="allocation">
                                        <span id="2b-allocation-btn">Allocation Structure</span>
                                </label>
                                <p class=hidden-answer2><i>Component-and-connector structure. This
question is about a runtime component and specifies how the data processes through a system
focussing on the relation between elements when it comes to data. Therefore a
component-and-connector structure is most suitable to answer this question.</i></p>
                                <p>c. What other software elements is a module allowed to use?</p>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="module">
                                        <span id="2c-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="compcon">
                                        <span id="2c-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="allocation">
                                        <span id="2c-allocation-btn">Allocation Structure</span>
                                </label>
                                <p class=hidden-answer2><i>Module structure. One might think that
since this question address the module and what elements it may use that it therefore is answered
by a component-and-connector structure. But here it is not about the actual relation they have,
just that they have a relation. Therefore a use structure answers this question which is a module
type structure.</i></p>
                        </form>
                        <label class="btn">
                                <input type="button" name="" value="Check Answers"
onclick="checkAnswers2()">
                                <span>Check Answers</span>
                        </label>

                </div>
                <div class="button-container">
                        <a href="chapter1-1.html" title="Chapter 1.1 What software architecture is
and what it is not"><span class="back-button">< Chapter 1.1</span></a>
                        <a href="chapter1-3.html" title="Chapter 1.3 Architectural Patterns"><span
class="next-button">Chapter 1.3 ></span></a>
                </div>
        </div>
</main>
```

```
<footer>
	Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

Chapter1-3.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="../css/style.css">
<script defer src="../javascript/script1-3.js"></script>
</head>
<body>

<header>
	<a href="../index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
	<div class="container">
		<div class="chapter-section">
			<h1>1.3 Architectural Patterns</h1>
			<p>Architectural patterns form an important aspect of software
architecture. In this video the concept architectural pattern as well as several examples will be
discussed.</p>
			<div class="video-container"><iframe
src="https://www.youtube.com/embed/dkgJORpLDwk"></iframe></div>
		</div>
		<div class="chapter-section">
			<h2>Summary</h2>
			<ul>
				<li>Useful compositions of architectural elements are called
architectural patterns. A definition by <a
href="https://en.wikipedia.org/wiki/Architectural_pattern">Wikipedia (link)</a> is as follows:
<br>
					<i>An architectural pattern is a general, reusable solution
to a commonly occurring problem in software architecture within a given context.</i>
				</li>
				<li>Examples of architectural patterns are the following:
					<ul>
						<li>Layered pattern</li>
						<li>Shared-data pattern</li>
						<li>Client-server pattern</li>
						<li>Multi-tier pattern</li>
						<li>Competence center and platform</li>
					</ul>
				</li>
			</ul>
		</div>
		<div class="chapter-section">
			<h2>Exercises</h2>
			<h3>1. A diagram example of an architectural pattern is shown. Decide what
type of pattern it is(module, C&C or allocation) and the specific pattern name.</h3>
			<form name="exercise1">
				<p>a. diagram source: Software in Practice by L.Bass</p>
```

```
                                <img src="../images/Exercises5a.png" alt="Exercises diagram 5a">
                                <label class="btn">
                                        <input type="radio" name="question2a" value="module">
                                        <span id="2a-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="compcon">
                                        <span id="2a-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2a" value="allocation">
                                        <span id="2a-allocation-btn">Allocation Structure</span>
                                </label>
                                <label class="text-input">
                                        <span>Specific pattern name</span>
                                        <input type="text-input">
                                </label>
                                <p class=hidden-answer1><i>Multi-tier pattern. Different computing
components are grouped together in different execution structures. Source: Figure 13.15 in the
book Software Architecture in Practice by L. Bass, A multi-tier view of the consumer website java
EE application, which is part of the Adventure Builder System</i></p>
                                <p>b. diagram source: Software in Practice by L.Bass</p>
                                <img src="../images/Exercises5b.png" alt="Exercises diagram 5b">
                                <label class="btn">
                                        <input type="radio" name="question2b" value="module">
                                        <span id="2b-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2b" value="compcon">
                                        <span id="2b-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2b" value="allocation">
                                        <span id="2b-allocation-btn">Allocation Structure</span>
                                </label>
                                <label class="text-input">
                                        <span>Specific pattern name</span>
                                        <input type="text-input">
                                </label>
                                <p class=hidden-answer1><i>Client-server pattern. Source: Figure
13.9 in the book Software Architecture in Practice by L. Bass, The client-server architecture of
an ATM banking system.</i></p>
                                <p>c. diagram source: Software in Practice by L.Bass</p>
                                <img src="../images/Exercises5c.png" alt="Exercises diagram 5c">
                                <label class="btn">
                                        <input type="radio" name="question2c" value="module">
                                        <span id="2c-module-btn">Module Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="compcon">
                                        <span id="2c-compcon-btn">Component-and-Connector
Structure</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question2c" value="allocation">
                                        <span id="2c-allocation-btn">Allocation Structure</span>
```

```
                                </label>
                                <label class="text-input">
                                        <span>Specific pattern name</span>
                                        <input type="text-input">
                                </label>
                                <p class=hidden-answer1><i>Layer pattern. Source: Figure 13.2 in
the book Software Architecture in Practice by L. Bass, A simple layer diagram, with a simple key
answering the uses question.</i></p>
                        </form>
                        <label class="btn">
                                <input type="button" name="" value="Check Answers"
onclick="checkAnswers1()">
                                <span>Check Answers</span>
                        </label>
                </div>
                <div class="button-container">
                        <a href="chapter1-2.html" title="Chapter 1.2 Architectural structures and
views"><span class="back-button">< Chapter 1.2</span></a>
                        <a href="chapter1-4.html" title='1.4 What makes a "Good"
Architecture?'><span class="next-button">Chapter 1.4 ></span></a>
                </div>
        </div>
</main>
<footer>
        Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

Chapter1-4.html

```
<!DOCTYPE html>
<html>
<head>
<title>Educational Tool on Software Architecture</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="../css/style.css">
<script defer src="../javascript/script1-4.js"></script>
</head>
<body>

<header>
        <a href="../index.html" title="Home Page">Educational Tool on Software Architecture</a>
</header>
<main>
        <div class="container">
                <div class="chapter-section">
                        <h1>1.4 What makes a "Good" Architecture?</h1>
                        <p>In this last video of chapter 1 we try to give an answer to the question
what makes a good architecture. Several recommendations are given.</p>
                        <div class="video-container"><iframe
src="https://www.youtube.com/embed/5JOxq0WA_-o"></iframe></div>
                </div>
                <div class="chapter-section">
                        <h2>Summary</h2>
                        <ul>
```

```
                        <li>Deciding if an architecture is good or bad depends on the
purpose. The fit of an architecture can be evaluated based on the quality attributes of the
system.</li>
                        <li>There are however recommendations that can be made to strive
for good architecture. they can be split between process and product recommedations.
                                <ul>
                                        <li>Process recommendations are concerned about the
process of desiging the architecture.</li>
                                        <li>Product recommendations focus on what traits the
actual product should have.</li>
                                </ul>
                        </li>
                        <li>A list of all the recommendations can be found in the video
above or in the book Software Architecture in Practice by L. Bass.</li>
                </ul>
        </div>
        <div class="chapter-section">
                <h2>Exercises</h2>
                <h3>1. Different Recommendations are shown on screen. Determine if it is a
good or bad recommendation.</h3>
                <form name="exercise1">
                        <p>a. The design of an architecture should be done in a big group
to obtain as many ideas as possible.</p>
                        <label class="btn">
                                <input type="radio" name="question1a" value="Good">
                                <span id="1a-good-btn">Good</span>
                        </label>
                        <label class="btn">
                                <input type="radio" name="question1a" value="Bad">
                                <span id="1a-bad-btn">Bad</span>
                        </label>
                        <p class=hidden-answer1><i>You want to minimize the amount of
people responsible for the architecture. This way you keep technical consisteny and conceptual
integrity throughout the design.</i></p>
                        <p>b. Actual implementation of the software architecture should
ideally be done at once.</p>
                        <label class="btn">
                                <input type="radio" name="question1b" value="Good">
                                <span id="1b-good-btn">Good</span>
                        </label>
                        <label class="btn">
                                <input type="radio" name="question1b" value="Bad">
                                <span id="1b-bad-btn">Bad</span>
                        </label>
                        <p class=hidden-answer1><i>You actually want the architecture to
lend itself for incremental implementation. This way you can find potential problems
earlier.</i></p>
                        <p>c. Modules that produce and consume data should be
seperated.</p>
                        <label class="btn">
                                <input type="radio" name="question1c" value="Good">
                                <span id="1c-good-btn">Good</span>
                        </label>
                        <label class="btn">
                                <input type="radio" name="question1c" value="Bad">
                                <span id="1c-bad-btn">Bad</span>
                        </label>
```

```
                                <p class=hidden-answer1><i>Often the production or consumption of
data changes. When these activities are seperated it will be easier to modify.</i></p>
                                <p>d. An architecture should have a large number of ways for
components to interact.</p>
                                <label class="btn">
                                        <input type="radio" name="question1d" value="Good">
                                        <span id="1d-good-btn">Good</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1d" value="Bad">
                                        <span id="1d-bad-btn">Bad</span>
                                </label>
                                <p class=hidden-answer1><i>This is not ideal as it increases the
complexity and therefore makes the implementation more difficult. For easier modifiablity it is
better to reduce the number of different interactions between modules.</i></p>
                                <p>e. Use well-known archtectural patterns to achieve the quality
attributes.</p>
                                <label class="btn">
                                        <input type="radio" name="question1e" value="Good">
                                        <span id="1e-good-btn">Good</span>
                                </label>
                                <label class="btn">
                                        <input type="radio" name="question1e" value="Bad">
                                        <span id="1e-bad-btn">Bad</span>
                                </label>
                                <p class=hidden-answer1><i>These proven patterns give good
structure to your system and will also allow for easier understandability. Make sure to pick a
pattern that fits the quality attributes.</i></p>
                        </form>
                        <label class="btn">
                                <input type="button" name="" value="Check Answers"
onclick="checkAnswers1()">
                                <span>Check Answers</span>
                        </label>
                </div>
                <div class="button-container">
                        <a href="chapter1-3.html" title="Chapter 1.3 Architectural Patterns"><span
class="back-button">< Chapter 1.3</span></a>
                        <a href="../end-tool.html" title="End page of the educational tool"><span
class="next-button">Next ></span></a>
                </div>
        </div>
</main>
<footer>
        Educational Tool by Arno de Vries as part of the Bachelor Graduation Project at the
University of Twente. Contact a.j.devries@student.utwente.nl
</footer>
</body>
</html>
```

CSS

Style.css

```
* {
        font-family: sans-serif;
        box-sizing: border-box;
        margin: 0;
        padding: 0;
```

```
        color: #10443e;
}

:root {
        --light-grey: #def7ee;
}

header {
        text-align: center;
        font-weight: 700;
        font-size: 48px;
        padding: 50px 0px;
        background-color: var(--light-grey);
}

a {
        text-decoration: none;
}

a:hover {
        color: #48a219;
}

main {
        background-color: white;
        display: flex;
        justify-content: center;
        min-height: 60vh;
}
h1 {
        font-size: 32px;
}

h1, h2 {
        text-transform: uppercase;
}

p {
        padding: 10px 0;
}

ul {
        margin-left: 20px;
}

iframe {
        display: block;
        margin: 0 auto;
        width: 800px;
        height: 500px;
}
footer{
        padding: 100px;
        background-color: #696969;
        color: white;
}

img {
```

```
        max-width: 100%;
        padding-top: 20px;
        display: block;
}

h3 {
        font-size: 20px;
}

button {
        border-radius: 0;
        padding: 10px 20px;
}

#answers {
        display: none;
}

.container {
        max-width: 1000px;
        width: 100%;
        padding: 20px;
}

.chapter1, .chapter3{
        background-color: var(--light-grey);
}

.chapter-section {
        padding: 50px 0;
}

.video-container {
        position: relative;
        padding-bottom: 56.25%;
        height: 0;
}

.video-container iframe {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
}

/* Setting the label as a inline block element */
.btn {
        display: inline-block;
        margin: 20px 10px;
}

/* Hiding the original radio button */
.btn input{
        position: absolute;
        opacity: 0;
        cursor: pointer;
}
```

```
/* The span is the actual button that you can select */
.btn span{
	padding: 10px 20px;
	cursor: pointer;
	font-size: 18px;
	background-color: white;
	border: 3px solid #10443e;
	border-radius: 5px;
}

/* Hover animation when mouse is over button */
.btn span:hover {
	background-color: lightgrey;
	border-color: #48a219;
}

/* Change the background and border color when button is selected */
.btn input:checked ~ span{
	background-color: #10443e;
	border-color: #10443e;
	border-width: 5px;
	color: white;
}

.text-input {
	display: block;
	margin: 10px;
}

/* setting up the next and back buttons*/
.next-button, .back-button{
	padding: 10px 20px;
	cursor: pointer;
	font-size: 18px;
	background-color: white;
	border: 3px solid #10443e;
	border-radius: 5px;
	display: inline-block;
}

/* hover animation when mouse is over the button */
.next-button:hover, .back-button:hover {
	background-color: lightgrey;
	border-color: #48a219;
}

/* next button should move to the right */
.next-button {
	float: right;
}

/* back button should move to the left */
.back-button {
	float: left;
}

/* a button container because the buttons were not showing up correctly */
```

```
.button-container {
	padding: 10px;
	height: 80px;
}

.hidden-answer1, .hidden-answer2{
	display: none;
	margin-left: 10px;
}
```

JavaScript

script1-1.js

```
function checkAnswers1 () {
	var q1a = document.exercise1.question1a.value;
	var q1b = document.exercise1.question1b.value;
	var q1c = document.exercise1.question1c.value;
	var q1d = document.exercise1.question1d.value;
	var q1e = document.exercise1.question1e.value;


	if (q1a=="True"){
		document.getElementById("1a-true-btn").style.color = "#48a219";
	}else{
		document.getElementById("1a-false-btn").style.color = "#FF7F7F";
		document.getElementById("1a-true-btn").style.color = "#48a219";
	}
	if (q1b=="False"){
		document.getElementById("1b-false-btn").style.color = "#48a219";
	}else{
		document.getElementById("1b-true-btn").style.color = "#FF7F7F";
		document.getElementById("1b-false-btn").style.color = "#48a219";
	}

	if (q1c=="True"){
		document.getElementById("1c-true-btn").style.color = "#48a219";
	}else{
		document.getElementById("1c-false-btn").style.color = "#FF7F7F";
		document.getElementById("1c-true-btn").style.color = "#48a219";
	}

	if (q1d=="True"){
		document.getElementById("1d-true-btn").style.color = "#48a219";
	}else{
		document.getElementById("1d-false-btn").style.color = "#FF7F7F";
		document.getElementById("1d-true-btn").style.color = "#48a219";
	}

	if (q1e=="False"){
		document.getElementById("1e-false-btn").style.color = "#48a219";
	}else{
		document.getElementById("1e-true-btn").style.color = "#FF7F7F";
		document.getElementById("1e-false-btn").style.color = "#48a219";
	}

	var answers = document.getElementsByClassName('hidden-answer1');
	for (var i = 0; i < answers.length; i++) {
		answers[i].style.display = 'block';
```

```
        }
}

function checkAnswers2 (){

        var q2a = document.exercise2.question2a.value;
        var q2b = document.exercise2.question2b.value;
        var q2c = document.exercise2.question2c.value;

        if (q2a=="module"){
                document.getElementById("2a-module-btn").style.color = "#48a219";
        }else{
                document.getElementById("2a-compcon-btn").style.color = "#FF7F7F";
                document.getElementById("2a-allocation-btn").style.color = "#FF7F7F";
                document.getElementById("2a-module-btn").style.color = "#48a219";
        }

        if (q2b=="compcon"){
                document.getElementById("2b-compcon-btn").style.color = "#48a219";
        }else{
                document.getElementById("2b-module-btn").style.color = "#FF7F7F";
                document.getElementById("2b-allocation-btn").style.color = "#FF7F7F";
                document.getElementById("2b-compcon-btn").style.color = "#48a219";
        }

        if (q2c=="allocation"){
                document.getElementById("2c-allocation-btn").style.color = "#48a219";
        }else{
                document.getElementById("2c-compcon-btn").style.color = "#FF7F7F";
                document.getElementById("2c-module-btn").style.color = "#FF7F7F";
                document.getElementById("2c-allocation-btn").style.color = "#48a219";
        }

        var answers = document.getElementsByClassName('hidden-answer2');
        for (var i = 0; i < answers.length; i++) {
                answers[i].style.display = 'block';
        }

}
```

Script1-2.js

```
function checkAnswers1 () {
        var q1a = document.exercise1.question1a.value;
        var q1b = document.exercise1.question1b.value;
        var q1c = document.exercise1.question1c.value;


        if (q1a=="View"){
                document.getElementById("1a-view-btn").style.color = "#48a219";
        }else{
                document.getElementById("1a-structure-btn").style.color = "#FF7F7F";
                document.getElementById("1a-view-btn").style.color = "#48a219";
        }

        if (q1b=="Structure"){
                document.getElementById("1b-structure-btn").style.color = "#48a219";
        }else{
                document.getElementById("1b-view-btn").style.color = "#FF7F7F";
```

```
                document.getElementById("1b-structure-btn").style.color = "#48a219";
        }

        if (q1c=="Structure"){
                document.getElementById("1c-structure-btn").style.color = "#48a219";
        }else{
                document.getElementById("1c-view-btn").style.color = "#FF7F7F";
                document.getElementById("1c-structure-btn").style.color = "#48a219";
        }


        var answers = document.getElementsByClassName('hidden-answer1');
        for (var i = 0; i < answers.length; i++) {
                answers[i].style.display = 'block';
        }
}

function checkAnswers2 (){

        var q2a = document.exercise2.question2a.value;
        var q2b = document.exercise2.question2b.value;
        var q2c = document.exercise2.question2c.value;

        if (q2a=="allocation"){
                document.getElementById("2a-allocation-btn").style.color = "#48a219";
        }else{
                document.getElementById("2a-compcon-btn").style.color = "#FF7F7F";
                document.getElementById("2a-module-btn").style.color = "#FF7F7F";
                document.getElementById("2a-allocation-btn").style.color = "#48a219";
        }

        if (q2b=="compcon"){
                document.getElementById("2b-compcon-btn").style.color = "#48a219";
        }else{
                document.getElementById("2b-module-btn").style.color = "#FF7F7F";
                document.getElementById("2b-allocation-btn").style.color = "#FF7F7F";
                document.getElementById("2b-compcon-btn").style.color = "#48a219";
        }

        if (q2c=="module"){
                document.getElementById("2c-module-btn").style.color = "#48a219";
        }else{
                document.getElementById("2c-compcon-btn").style.color = "#FF7F7F";
                document.getElementById("2c-allocation-btn").style.color = "#FF7F7F";
                document.getElementById("2c-module-btn").style.color = "#48a219";
        }

        var answers = document.getElementsByClassName('hidden-answer2');
        for (var i = 0; i < answers.length; i++) {
                answers[i].style.display = 'block';
        }

}
```

Script1-3.js

```
function checkAnswers1 (){
```

```
	var q2a = document.exercise1.question2a.value;
	var q2b = document.exercise1.question2b.value;
	var q2c = document.exercise1.question2c.value;

	if (q2a=="allocation"){
		document.getElementById("2a-allocation-btn").style.color = "#48a219";
	}else{
		document.getElementById("2a-compcon-btn").style.color = "#FF7F7F";
		document.getElementById("2a-module-btn").style.color = "#FF7F7F";
		document.getElementById("2a-allocation-btn").style.color = "#48a219";
	}

	if (q2b=="compcon"){
		document.getElementById("2b-compcon-btn").style.color = "#48a219";
	}else{
		document.getElementById("2b-module-btn").style.color = "#FF7F7F";
		document.getElementById("2b-allocation-btn").style.color = "#FF7F7F";
		document.getElementById("2b-compcon-btn").style.color = "#48a219";
	}

	if (q2c=="module"){
		document.getElementById("2c-module-btn").style.color = "#48a219";
	}else{
		document.getElementById("2c-compcon-btn").style.color = "#FF7F7F";
		document.getElementById("2c-allocation-btn").style.color = "#FF7F7F";
		document.getElementById("2c-module-btn").style.color = "#48a219";
	}

	var answers = document.getElementsByClassName('hidden-answer1');
	for (var i = 0; i < answers.length; i++) {
		answers[i].style.display = 'block';
	}

}
```

Script1-4.js

```
function checkAnswers1 () {
	var q1a = document.exercise1.question1a.value;
	var q1b = document.exercise1.question1b.value;
	var q1c = document.exercise1.question1c.value;
	var q1d = document.exercise1.question1d.value;
	var q1e = document.exercise1.question1e.value;


	if (q1a=="Bad"){
		document.getElementById("1a-bad-btn").style.color = "#48a219";
	}else{
		document.getElementById("1a-good-btn").style.color = "#FF7F7F";
		document.getElementById("1a-bad-btn").style.color = "#48a219";
	}
	if (q1b=="Bad"){
		document.getElementById("1b-bad-btn").style.color = "#48a219";
	}else{
		document.getElementById("1b-good-btn").style.color = "#FF7F7F";
		document.getElementById("1b-bad-btn").style.color = "#48a219";
	}

	if (q1c=="Good"){
```

```
                document.getElementById("1c-good-btn").style.color = "#48a219";
        }else{
                document.getElementById("1c-bad-btn").style.color = "#FF7F7F";
                document.getElementById("1c-good-btn").style.color = "#48a219";
        }

        if (q1d=="Bad"){
                document.getElementById("1d-bad-btn").style.color = "#48a219";
        }else{
                document.getElementById("1d-good-btn").style.color = "#FF7F7F";
                document.getElementById("1d-bad-btn").style.color = "#48a219";
        }

        if (q1e=="Good"){
                document.getElementById("1e-good-btn").style.color = "#48a219";
        }else{
                document.getElementById("1e-bad-btn").style.color = "#FF7F7F";
                document.getElementById("1e-good-btn").style.color = "#48a219";
        }



        var answers = document.getElementsByClassName('hidden-answer1');
        for (var i = 0; i < answers.length; i++) {
                answers[i].style.display = 'block';
        }
}
```

Appendix C Survey I

Survey on online educational tool on software architecture

* Required

Information and Consent

You have been invited to participate in a survey on an online educational tool on software architecture. The aim of this tool is to improve the participants knowledge and skills on software architecture. This tool is part of the Bachelor Graduation Project by Arno de Vries at the University of Twente.

In this survey you are asked to give feedback on the online educational tool based on your experience with the tool. The online educational tool is the tool that was developed for this graduation project. It can be found at http://edu-tool.arno-de-vries.nl/.

Your participation in this study is entirely voluntary and you can withdraw at any time.

This survey does not collect email address or other personal information. Answers to this questionnaire may be shared with other researchers for future research studies that may be similar to this study or may be completely different. The information shared with other researchers will not include any information that can directly identify you. Researchers will not contact you for additional permission to use this information. The data from this survey will be temporarily stored on a University of Twente Google Drive account. When the thesis is published, the results will be published in this document as an appendix. After publication the data will be removed from the University of Twente Google Drive account.

For question regarding this research please contact Arno de Vries (a.j.devries@student.utwente.nl) or the supervisor Ansgar Fehnker (ansgar.fehnker@utwente.nl)

If you have questions about your rights as a research participant or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee, Faculty of EEMCS, by email ethics-comm-ewi@utwente.nl.

1. I understand the information provided above and consent to its conditions for participation. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

Steps of this Survey	To participate in this survey I would like ask you to execute the following steps. 1. Go to the following website that hosts the educational tool: http://edu-tool.arno-de-vries.nl/ 2. Take a look at the videos and exercises from chapter 1. 3. Go back to this Google Form to give feedback on the educational tool.

Usability of the Tool

2. The tool is easy to use *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

3. It is clear how to use the tool *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

4. Using the tool frustrated me *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. What did you like most about the tool? *

6. What did you like the least about the tool? *

7. Any comments on the usability of the tool

Design of the Tool

8. The design of the tool is attractive *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. The tool is engaging *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. The tool is interesting *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. Any comments on the design of the tool

Effectiveness of the Tool

12. The tool is very useful *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

13. I understand the contents of the videos *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

14. This tool can help me learn more about software architecture *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

15. The learning objectives of the tool are clear *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

16. This tool will achieve the proposed learning objectives *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

17. Any comments on the effectiveness of the tool

General questions

18. Where do you think this tool could be useful within the Creative Technology curriculum? *

19. If you have any general comments on the tool or this research project you can place them here.

20. Which study program are you in? *

Mark only one oval.

- ◯ Bachelor Creative Technology
- ◯ Bachelor Technical Computer Science
- ◯ Master Interaction Technology
- ◯ Master Computer Science
- ◯ Other: _____

21. Which year did you start your bachelor program? *

Mark only one oval.

- 2016
- 2017
- 2018
- 2019
- 2020
- 2021
- Other: _____

Appendix D Survey I Results

Usability of the Tool

Question	Resp. 1	Resp. 2	Resp. 3	Resp. 4	Resp. 5
The tool is easy to use	5	3	4	4	4
It is clear how to use the tool	5	4	5	4	5
Using the tool frustrated me	3	2	1	1	3

What did you like most about the tool?

Resp. 1	Nice integration of YouTube videos
Resp. 2	Everything is in one place.
Resp. 3	I like it that there are separate videos instead of one long video
Resp. 4	easy to use at your own speed
Resp. 5	The usability was good. It is self explanatory

What did you like the least about the tool?

Resp. 1	The tool doesn't use the entire screen
Resp. 2	I would have liked to have an option to see what was presented in the youtube videos textually. This would make sure I could read through what was said without having to find the sepecific part of the video that describes what I am looking for.
Resp. 3	That all the information is on one page. It would be better if it was separated on different pages (each category/video on a different page)
Resp. 4	graphics of the questions could make it easier to read
Resp. 5	The design could be more inviting/visually appealing

Any comments on the usability of the tool

Resp. 1	Maybe sections on different pages would be useful to decrease cognitive overload (don't know if that's the right name for it). Furthermore, I personally prefer answers right after each question instead of at the end
Resp. 2	I expected to be able to fill in answers and afterwards be shown if my answers are correct, instead of an option to just show the answers.
Resp. 3	Instead of the "Show answers" button you could make a form in which people can fill in the answers on the questions. And then a "Submit" button at the end and show how many answers the user filled in correctly. Overall, the usability of the tool is good!

Resp. 4	nope
Resp. 5	Would be nice if less scrolling was necessary. For example by letting the exercise answers appear next to, or close to the exercises

Design of the Tool

Question	Resp. 1	Resp. 2	Resp. 3	Resp. 4	Resp. 5
The design of the tool is attractive	4	4	3	3	2
The tool is engaging	3	2	4	4	2
The tool is interesting	2	4	3	5	2

Any comments on the design of the tool

Resp. 1	The show answers button looks a bit outdated
Resp. 2	I would have preferred to have the videos and assignments follow each other. So: video - assignment - video - assignment, etc. At the end of the fourth video, I had already forgotten what was said in the first video.
Resp. 3	
Resp. 4	again the questions look like a large part of text, could be made more appealing
Resp. 5	

Effectiveness of the Tool

Question	Resp. 1	Resp. 2	Resp. 3	Resp. 4	Resp. 5
The tool is very useful	4	3	4	5	3
I understand the contents of the videos	3	4	5	5	2
This tool can help me learn more about software architecture	5	4	5	5	4
The learning objectives of the tool are clear	5	3	4	5	3
This tool will achieve the proposed learning objectives	4	3	4	5	4

Any comments on the effectiveness of the tool

Resp. 1	
Resp. 2	For me it was unclear who the target audience is of this tool and what knowledge is needed beforehand. If the tool is designed to learn software architecture without any knowledge about the topic beforehand, for example for students or employees, I think it is a bit overwhelming

	since there any many terms involved and some knowledge of software systems is required. However, if you already have some knowledge this tool would be effective to quickly learn some things about software architecture.
Resp. 3	
Resp. 4	
Resp. 5	

Where do you think this tool could be useful within the Creative Technology curriculum?

Resp. 1	Theoretical courses like computer science
Resp. 2	Around module 6. Then they have some programming knowledge already, which I feel is necesarry.
Resp. 3	In the first year (module 2,3 or 4)
Resp. 4	in the first year when we start learning about writing software
Resp. 5	Getting understanding of the underlying principles in programming and computer science

If you have any general comments on the tool or this research project you can place them here.

Resp. 1	
Resp. 2	
Resp. 3	
Resp. 4	
Resp. 5	

Which study program are you in?

Resp. 1	Bachelor Creative Technology
Resp. 2	Bachelor Creative Technology
Resp. 3	Bachelor Creative Technology
Resp. 4	Bachelor Creative Technology
Resp. 5	Bachelor Creative Technology

Which year did you start your bachelor program?

Resp. 1	2018
Resp. 2	2017

Resp. 3	2018
Resp. 4	2017
Resp. 5	2017

Appendix E Survey II

Survey on online educational tool on software architecture

* Required

Information and Consent

You have been invited to participate in a survey on an online educational tool on software architecture. The aim of this tool is to improve the participants knowledge and skills on software architecture. This tool is part of the Bachelor Graduation Project by Arno de Vries at the University of Twente.

In this survey you are asked to give feedback on the online educational tool based on your experience with the tool. The online educational tool is the tool that was developed for this graduation project. It can be found at http://edu-tool.arno-de-vries.nl/.

Your participation in this study is entirely voluntary and you can withdraw at any time.

This survey does not collect email address or other personal information. Answers to this questionnaire may be shared with other researchers for future research studies that may be similar to this study or may be completely different. The information shared with other researchers will not include any information that can directly identify you. Researchers will not contact you for additional permission to use this information. The data from this survey will be temporarily stored on a University of Twente Google Drive account. When the thesis is published, the results will be published in this document as an appendix. After publication the data will be removed from the University of Twente Google Drive account.

For question regarding this research please contact Arno de Vries (a.j.devries@student.utwente.nl) or the supervisor Ansgar Fehnker (ansgar.fehnker@utwente.nl)

If you have questions about your rights as a research participant or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee, Faculty of EEMCS, by email ethics-comm-ewi@utwente.nl.

1. I understand the information provided above and consent to its conditions for participation. *

   Mark only one oval.

   - ◯ Yes
   - ◯ No

Steps of this Survey

To participate in this survey I would like ask you to execute the following steps.

1. Go to the following website that hosts the educational tool: http://edu-tool.arno-de-vries.nl/
2. Take a look at the videos and exercises from chapter 1.
3. Go back to this Google Form to give feedback on the educational tool.

After these steps the survey is done. So now you can start by going to the educational tool.

Usability of the Tool

2. The tool is easy to use *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

3. The interactive component of the exercises was intuitive *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

4. It is clear how to use the tool *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. Using the tool frustrated me *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

6. What did you like most about the tool? *

7. What did you like the least about the tool? *

8. Any comments on the usability of the tool

Design of the Tool

9. The design of the tool is attractive *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. The tool is engaging *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. The tool is interesting *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

12. The design of the exercises were attractive *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

13. Any comments on the design of the tool

Effectiveness of the Tool

14. The tool is very useful *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

15. I understand the contents of the videos *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

16. This tool can help me learn more about software architecture *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

17. The learning objectives of the tool are clear *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

18. This tool will achieve the proposed learning objectives *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree						Strongly agree

19. The exercises helped me become more engaged in the tool *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree						Strongly agree

20. Any comments on the effectiveness of the tool

General questions

21. Where do you think this tool could be useful within the Creative Technology curriculum?

22. If you have any general comments on the tool or this research project you can place them here.

23. Which study program are you in? *

Mark only one oval.

- ◯ Bachelor Creative Technology
- ◯ Bachelor Technical Computer Science
- ◯ Master Interaction Technology
- ◯ Master Computer Science
- ◯ Other: _____

24. Which year did you start your bachelor program? *

Mark only one oval.

- ◯ 2016
- ◯ 2017
- ◯ 2018
- ◯ 2019
- ◯ 2020
- ◯ 2021
- ◯ Other: _____

Appendix F Survey II Results

Usability of the Tool

Question	Resp. 1	Resp. 2	Resp. 3	Resp. 4	Resp. 5
The tool is easy to use	5	5	4	4	5
The interactive component of the exercises was intuitive	5	5	4	3	5
It is clear how to use the tool	5	4	5	4	5
Using the tool frustrated me	1	1	2	4	1

What did you like most about the tool?

Resp. 1	It's simple and clear
Resp. 2	Very easy to use and pretty fun. Also a nice look
Resp. 3	I like the exercises next to the information and good video with visuals
Resp. 4	Quick checking of your knowledge with the quiz
Resp. 5	it makes the infromation easy to understand and gives great opportunity to test the knowledge you gained

What did you like the least about the tool?

Resp. 1	It could be more aesthetically pleasing
Resp. 2	When i clicked on a true/false i expected an immediate reaction on whether my answer is correct
Resp. 3	There were some buttons overlapping and the ‘check answers’ button could have another color so it is clear that it confirms the answers
Resp. 4	The design, it looks a bit dull
Resp. 5	you dont know how much chapters are still to come so maybe adding a diplay of your process would encourage the user to finish the chapters

Any comments on the usability of the tool

Resp. 1	
Resp. 2	Typing error in: "Anser True or False to the following statements on software architecture."
Resp. 3	Maybe adding colors or icons? And perhaps making some keywords in the text bold so the student recognizes these
Resp. 4	it would help if it were more attractive to use
Resp. 5	nope

Design of the Tool

Question	Resp. 1	Resp. 2	Resp. 3	Resp. 4	Resp. 5
The design of the tool is attractive	2	5	4	1	4
The tool is engaging	5	4	4	2	5
The tool is interesting	4	4	3	3	4
The design of the exercises were attractive	4	5	4	2	5

Any comments on the design of the tool

Resp. 1	
Resp. 2	
Resp. 3	
Resp. 4	looks boring, not sparking interest
Resp. 5	no

Effectiveness of the Tool

Question	Resp. 1	Resp. 2	Resp. 3	Resp. 4	Resp. 5
The tool is very useful	5	5	4	3	5
I understand the contents of the videos	4	4	3	2	5
This tool can help me learn more about software architecture	5	5	4	4	5
The learning objectives of the tool are clear	5	5	4	3	5
This tool will achieve the proposed learning objectives	5	5	3	3	5
The exercises helped me become more engaged in the tool	5	5	4	4	4

Any comments on the effectiveness of the tool

Resp. 1	
Resp. 2	
Resp. 3	
Resp. 4	
Resp. 5	

Where do you think this tool could be useful within the Creative Technology curriculum?

Resp. 1	In courses such as algorithms, where there is a lot of theory
Resp. 2	For more theory focused courses like computer science
Resp. 3	During the first year to get to know where students struggle, and during mod6 to help with the tutorials
Resp. 4	at the start of introducing programming
Resp. 5	when you start to learn about software architecture, it gives a great foundation and can be reused to recap in later parts of the curriculum

If you have any general comments on the tool or this research project you can place them here.

Resp. 1	
Resp. 2	
Resp. 3	
Resp. 4	
Resp. 5	

Which study program are you in?

Resp. 1	Bachelor Creative Technology
Resp. 2	Bachelor Creative Technology
Resp. 3	Bachelor Creative Technology
Resp. 4	Bachelor Creative Technology
Resp. 5	Bachelor Creative Technology

Which year did you start your bachelor program?

Resp. 1	2018
Resp. 2	2018
Resp. 3	2016
Resp. 4	2016
Resp. 5	2017

References

[1] Bass, L., Clements, P. and Kazman, R., 2012. Software Architecture in Practice, Third Edition. 3rd ed. Pearson Education Inc., p.3.

[2] UNESCO. 2021. Open Educational Resources (OER). [online] Available at: <https://en.unesco.org/themes/building-knowledge-societies/oer> [Accessed 18 March 2021].

[3] Youtube.com. n.d. Khan Academy. [online] Available at: <https://www.youtube.com/channel/UC4a-Gbdw7vOaccHmFo40b9g> [Accessed 18 March 2021].

[4] Schell, J., n.d. The art of game design. 2008.

[5] "Bachelor EER | Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)", Universiteit Twente, 2021. [Online]. Available: https://www.utwente.nl/en/eemcs/education/rules-guidelines/eer-b/. [Accessed: 18- Mar- 2021].

[6] "Education and Examination Regulations Bachelor of Science Creative Technology," *utwente.nl*. [Online]. Available: https://www.utwente.nl/en/eemcs/education/rules-guidelines/eer-b/eer-b2020/ewi-bachelor-crea-oer-2020.pdf.

[7] A. Mader and W. Eggink, "A Design Process for Creative Technology," *Proceedings of the International Conference on Engineering and Product Design Education,* 2014

[8] M. Galster and S. Angelov, "What makes teaching software architecture difficult?," *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016.

[9] A. van Deursen, M. Aniche, J. Aue, R. Slag, M. de Jong, A. Nederlof and E. Bouwers, "A Collaborative Approach to Teaching Software Architecture," *Proceedings of the Conference on Integrating Technology into Computer Science Education,* 2017

[10] C. R. Rupakheti and S. V. Chenoweth, "Teaching Software Architecture to Undergraduate Students: An Experience Report," *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015.

[11] O. E. Lieh and Y. Irawan, "Teaching Adult Learners on Software Architecture Design Skills," *2018 IEEE Frontiers in Education Conference (FIE)*, 2018.

[12] S. Brown, *The C4 model for visualising software architecture*. [Online]. Available: https://c4model.com/. [Accessed: 18-Apr-2021].

[13] P. J. Guo, J. Kim, and R. Rubin, "How video production affects student engagement," *Proceedings of the first ACM conference on Learning @ scale conference*, 2014.

[14] J. Schell, *The art of game design: a book of lenses*. Boca Raton: CRC Press, 2020.

[15] M. Lavi, “The Complete Guide to Becoming a Software Architect,” *Udemy*. [Online]. Available: https://www.udemy.com/course/the-complete-guide-to-becoming-a-software-architect/?utm_source=adwords&utm_medium=udemyads&utm_campaign=LongTail-New_la.EN_cc.ROWMTA-B&utm_content=deal4584&utm_term=_._ag_101378276820_._ad_438174747322_._kw__._de_c_._dm__._pl__._ti_dsa-1007766171032_._li_1010605_._pd__._&matchtype=b&gclid=Cj0KCQjw6-SDBhCMARIsAGbI7UhKOStfI4QRofBZorTiXD9-Lc7qsIbjTTFx9wg2rJtPxB-kALEU-RkaAvxqEALw_wcB. [Accessed: 16-Apr-2021].
[16] C. Simmons, “Developer to Architect,” *Pluralsight*, 23-Oct-2013. [Online]. Available: https://www.pluralsight.com/courses/developer-to-architect?aid=7010a000002BWqGAAW&promo=&utm_source=non_branded&utm_medium=digital_paid_search_google&utm_campaign=EMEA_Dynamic&utm_content=&gclid=Cj0KCQjw6-SDBhCMARIsAGbI7UgVFAgVWgeqbgcZadNN_sT6BwDRkSanC-Ak_QYh_wL3Fyw5DfOU_aYaAhYoEALw_wcB. [Accessed: 16-Apr-2021].
[17] K. Wong, “Software Architecture,” *Coursera*. [Online]. Available: https://www.coursera.org/learn/software-architecture?ranMID=40328&ranEAID=JVFxdTr9V80&ranSiteID=JVFxdTr9V80-0BGMb8oP_twoE5AnYJ7ceA&siteID=JVFxdTr9V80-0BGMb8oP_twoE5AnYJ7ceA&utm_content=10&utm_medium=partners&utm_source=linkshare&utm_campaign=JVFxdTr9V80. [Accessed: 16-Apr-2021].
[18] M. Farragher, “Become a Solution Architect: Architecture Course,” *Udemy*. [Online]. Available: https://www.udemy.com/course/how-to-become-an-outstanding-solution-architect/?LSNPUBID=JVFxdTr9V80&ranEAID=JVFxdTr9V80&ranMID=39197&ranSiteID=JVFxdTr9V80-_Aw5RzmuQshcFHYc8rYsTQ&utm_medium=udemyads&utm_source=aff-campaign. [Accessed: 16-Apr-2021].
[19] J. Christopher Jones, *Design methods.* Wiley-Interscience, London, 1970.
[20] En.wikipedia.org. 2021. MoSCoW method - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/MoSCoW_method> [Accessed 2 August 2021].
[20] “Mobile vs. desktop usage in 2020,” Perficient, Inc. [Online]. Available: https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage. [Accessed: 02-Aug-2021].
[21] “meCUE 2.0,” Evaluation of interactive products meCUE 2.0 questionnaire. [Online]. Available: https://mecue.de/Homepage%20Content/english/meCUE_PV.pdf. [Accessed: 02-Aug-2021].