Software Curriculum @ Siemens – The architecture of a training program for architects

Matthias Backert
Global Learning Campus
Siemens AG
Erlangen, Germany
matthias.backert@siemens.com

Thomas Blum
Magnetic Resonance
Siemens Healthineers
Erlangen, Germany
thomas.blum@siemens-healthineers.com

Rüdiger Kreuter
Digital Industries
Siemens AG
Munich, Germany
ruediger.kreuter@siemens.com

Frances Paulisch
Development Center
Siemens Healthineers
Forchheim, Germany
frances.paulisch@siemens-healthineers.com

Peter Zimmerer
Corporate Technology
Siemens AG
Munich, Germany
peter.zimmerer@siemens.com

***Abstract*—Siemens established a company-wide role-based qualification and certification curriculum, focusing on the topic of "architecture". Architects play a central role in the product lifecycle of the complex systems that Siemens offers. Since the curriculum's start in 2006 with the "senior software architect" program, we have added programs for software, system, and test architects. This curriculum is meanwhile broadly established throughout Siemens and Siemens Healthineers and other associated Siemens companies. We meanwhile have almost 700 certified architects in almost 20 countries.**

This industrial experience report describes the curriculum briefly but will focus on how the "architecture" of the training program for architects enabled us to evolve and extend the curriculum over time. The evolution covers not only additional roles, but also keeps the technical content up to date and adapts to modern learning formats as we strive to achieve the most impact in the available time. Our experience may be a blueprint for other key role curricula.

Keywords—software engineering education, software architecture training, software architecture, testing, system architecture, digitalization, product line engineering, training evolution

I. INTRODUCTION

As software-based systems become more mission-critical, it becomes increasingly important that the engineers developing these systems are well trained and highly motivated. Due to growing complexity and faster pace-of-change together with high quality attributes, the challenges of developing and maintaining such systems grows similarly.

Siemens' operating companies and strategic companies have together around 25,000 software engineers world-wide working on a diverse set of products including, but not limited to, industrial automation, smart infrastructure, mobility, and medical technology. Starting in 2006, a cross-company team, representing many different parts of the company and knowledge areas, worked together with our company-wide learning organization "Learning Campus" to establish a role-based qualification and certification program for architects, as they are one of the most important roles in research & development [1].

A. Key roles in a product development lifecycle

Architects have a very central role in the product development lifecycle of complex (software-based) systems. They should not only be experts in their field but essential also is their willingness, ability and skill in interacting with many of the other key roles. In some sense, they are a kind of "living bridge", i.e. a technical communication hub, to other roles.

One says a chain is only as strong as its weakest link and so here too the interfaces and "glue" between the roles is essential for a common understanding. In our view it is essential that the 4 different architect roles (including software, system and testing) are stronger and more effective together when they are well-aligned and interlinked, in other words 1+1+1+1 > 4.

In our projects the architect is the connection between management / business and technology ensuring that there is a common understanding across relevant roles. Especially there shall be a close interworking between what we call a "driving triumvirate" (Fig. 1), the architect (representing the interests of R&D), the product lifecycle manager (representing the product requirements), and the R&D project manager (representing the realization of this product) – or comparable roles in an agile environment. Of course, development happens in a test-driven way and with good testing infrastructure in place.

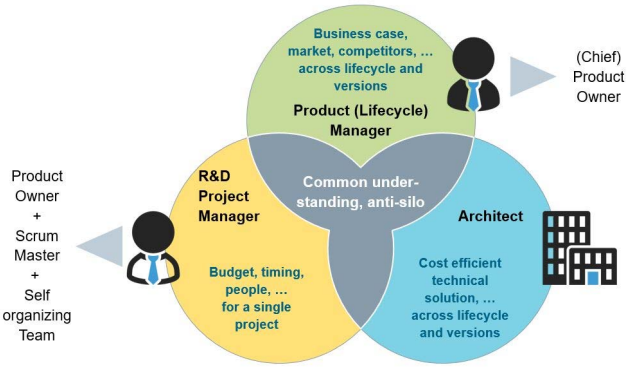

Figure 1: The "driving triumvirate" of R&D project manager, product (lifecycle) manager, and architect

In addition to the architect curriculum there are also other learning programs and trainings for other roles, where the understanding of the curriculum is conveyed so that both sides are aware of the importance of this alignment and have positive experience in working in this manner.

978-1-7281-6807-4/20/$31.00 ©2020 IEEE

B. *Curriculum Pyramid for Architects*

To be able to learn from the feedback and to have the most immediate impact, we started initially with the "senior software architect" (SSWA) role. The "senior" software architect role is for a software architect of a particularly large and complex system or, more typically, a product family or ecosystem. Over the past ca. 15 years we have rolled out qualification and certification programs additionally for team architects or architects of smaller projects, the "software architect" (SWA), for architects of mechatronic systems the "system architect" (SyA), and for the "test architect" (TeA).

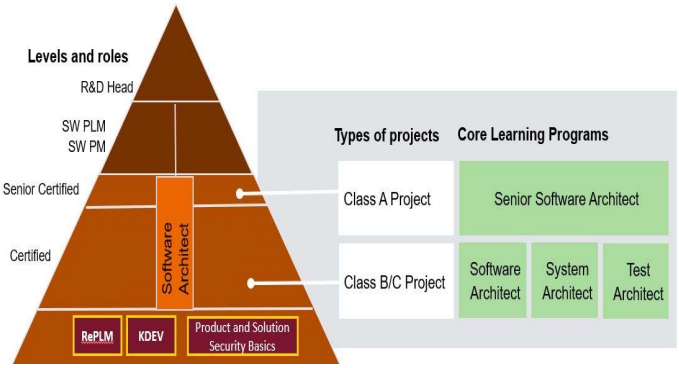

Figure 2: Curriculum Pyramid

Figure 2 shows how these programs are the core of our curriculum pyramid and are integrated with other learning programs and trainings. These four role-based programs will be described in more detail in section II. We meanwhile have almost 700 certified architects (from those four roles) in almost 20 countries.

In our experience "architectural thinking" is the best way to establish the right balance between a solid and modular foundation and having a flexible means to extend and adapt the foundation as needed for a particular case.

When developing the role-based qualification and certification program, this same "architectural thinking" proved valuable. We have a modular approach, put a high emphasis on the interfaces between the modules, have a clear understanding of where we draw the line between what is part of the common foundation and where the "hooks" are to extend and adapt.

Architecture is particularly important as it has a strong impact on the quality attributes of the system (that we often call the "non-functional requirements"). Another reason for our strong focus on architecture is that it forces you to think about what aspects are expensive to change later and to get those established early.

In this paper we will briefly provide an overview of the role-based curriculum (more details are described in [1]), describe the core asset base, the interfaces between these core assets, and how this architect learning program structure enables us to be particularly adaptable for change and provides a lightweight way to be more flexible for unforeseen future needs. We will have an explicit focus on the "what" (e.g. content) and the "how" (e.g. format). Our curriculum has proven to be adaptable on both of those dimensions. We close by summarizing our industrial experience with the program.

II. OVERVIEW OF THE ROLE-BASED CURRICULUM

A. *Content*

As shown in Figure 3 on the basis of the senior software architect (SSWA) program, all of the architect learning programs include a set of in-person workshops with big blocks of time between the workshops where the participant applies what they learned to their own "real-life" projects. Before the first workshop there is a nomination and preparation phase and at the end a final certification assessment and award ceremony.

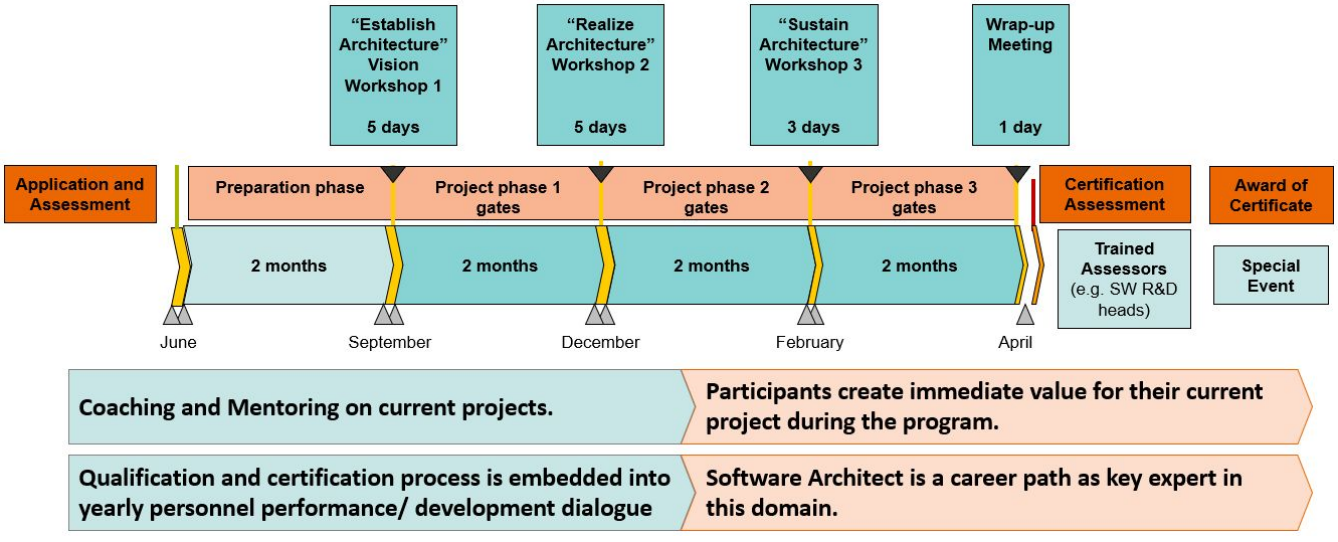

Figure 3: Timing Overview (example from SSWA)

The curriculum addresses different aspects of the 5 main topic areas: business understanding, requirements engineering, architecture & development, testing & quality, and social skills & leadership (see Figure 3). These topics serve as a way of structuring the training blocks within the curriculum. Figure 4 shows how these topic areas integrate related to each other. The additional topic area "Realize", that is e.g. coding, is not in the focus if this curriculum. We have a set of specific trainings for that.

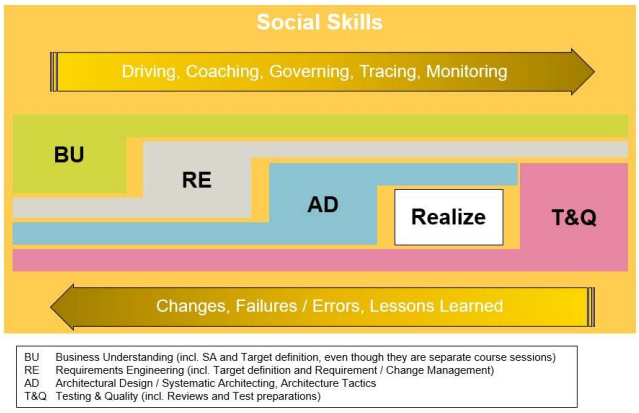

Figure 4: 5+1 topic areas

Figure 5 shows, using testing roles (test architect and test manager), the expected competence levels of these roles for important competence areas. Such competence spiders have been used to select the content for the various learning programs and exist also for the other roles.

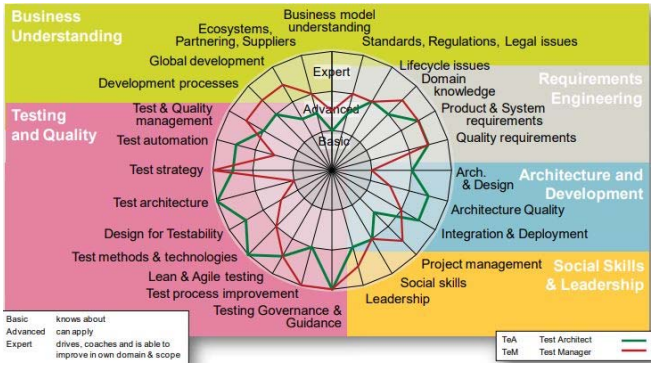

Figure 5: Spider diagram (example from TeA)

The strong emphasis on social skills and leadership for these more technical roles is a particularly strong area of our program as we find this topic to be of great importance. The architects often have the role as a coach and mentor for others and they must communicate well (and be able to listen to and understand the viewpoints and motivations of other roles) as two examples where these skills are valuable. In many of the "war stories" told in the program, project failures are much more often due to non-technical issues, so it is important for the participants to learn to recognize and react to warning signs early.

B. Format

The curriculum has a global mandate. Architects from around the world participate. For all four roles there are runs in Germany. For the SWA role we additionally have runs in the US, India, and China. The global scope of this program, which brings participants from all over the world together and includes also intercultural aspects, helps establish a common understanding for efficient global software engineering. We have extensive experience with geographically distributed software engineering for our products and it is natural that also such a learning "product" is globally distributed as well.

We have explicitly focused on how to ensure the same level of quality across the countries and also how to avoid too much travel (e.g. an alternative approach for certification in the global setup). We regularly exchange trainers i.e. sometimes German trainers do the training in India or China or trainers from India do training in China or Germany. In the cases where we have many different trainers (e.g. SWA) we have regular synchronization meetings amongst them.

We have daily feedback by course participants during the workshops and a summarizing feedback after each workshop. We summarize/discuss that feedback together with our central learning organization "Learning Campus". For all roles except the SWA the certification is done in Germany. For the SWA there is always at least one of the core assessment team members from Germany involved and the participant can stay in his or her country.

Within and beyond the courses we strongly drive and support networking aspects – this includes networking within the roles, across the runs, across the country etc. There is a central official list of all of the certified architects, there is a central wiki page with content especially created for and by the trainers and course members, special in-person networking events (for alumni and active course members), dedicated forums in the company-internal social media channels etc**.** It helps tremendously that the persons have the same common understanding, good social skills, and are eager to help each other.

We have made a very significant investment in the "trainer guideline" (one can view this as a kind of infrastructure to be able to hold the training) as opposed to slides or other materials that convey the content. Typically, the storyline has a set of "learning goals" and detailed information on how to achieve them (which methods, tools, "serious games" etc.). It also describes to what extent we try to convey a particular topic – do we just want to have the participant be aware of a topic, or be able to apply it, or to teach someone else the topic. In the workshops, the participants often explicitly try out new techniques in the trustful environment of the run (i.e. this instance of the training).

We have increasingly incorporated an agile mindset both in the content of the training and also in how the learning is achieved. Agile learning does not mean chaos and ad-hoc learning. Agile learning, like agile, has a long-term goal, a vision. In our case this would be the ability to fulfill a very important and challenging job profile. The trainer is like the product owner for the education – what each individual needs might depend on their experience and ability so different parts would be "pulled" and the trainer oversees and enables that all participants work towards the common vision that the necessary skills are available not only in the awareness of the participant but that the participant has experienced them in practice both in the workshop and in their real projects that they are the architect of.

III. CURRICULUM'S CORE ASSET BASE

When applying an architectural approach such as "product-line engineering" one often refers to a set of common assets (e.g. modules) that can be re-used directly or with modifications. We have explicitly taken this view to analyze, for example, what the commonalities and variabilities are between a software architect and an architect for mechatronic systems. Among the commonalities are to understand the domain, elaborate and maintain the architecture, integration responsibility, documentation and presentation skills, interface to requirement providers, moderation and conflict management, technical mentor and coach, and to drive realization quality.

We describe here many of the the common aspects across the four role-based learning programs.

A. Content

Slides and Trainer Guidelines: Although we do not use many slides for presentation, there are a set of slides that are provided to the participants as additional reference material for off-site learning. Furthermore, there is a large asset base available to the trainers – for example trainer guidelines for organizing activities designed to convey a particular concept. For example, in the system architect program we do a Design Space Exploration (DSE) exercise (a lean method) by producing 24 different paper plane variants with given material and design alternatives to find the best designs regarding cost and flight range

Guiding Principles: We have a set of guiding principles that apply to the whole curriculum:

- Architecture is the key throughout the whole lifecycle as well as across releases.
- Build on existing basis where feasible (from technical and business perspective) and be able to recognize when such reuse is not suitable
- Avoid unnecessary technological platform development by using technical standards and products available on the market
- The product manager must act as the owner of the main requirements
- Pay particular attention to non-functional requirements (NFRs), often overlooked but extremely important
- Be prepared and able to handle changing requirements, but be aware about the risk of late changes
- Synchronize well across the technical disciplines (software, mechanics, electronics, mechatronics, systems engineering)
- Work together truly as a team, avoid "silo" thinking, be willing and able to speak and understand the other roles and disciplines
- Work iteratively, strive to identify and resolve technical and business risks early
- Structure the system to avoid unnecessary complexity, and to actively enable and support multi-site development
- Strive for transparency and base decisions on clear business and/or technical reasons, not political ones
- Do not underestimate the importance of soft skills, these can be particularly important for convincing and motivating

Set of certification topics: There are a set of certification topics and a broad set of example questions that are used by the assessors in the certification assessment of the program. Note that the questions are just the start of the conversation with the assessors and how it is applied to the participant's situation, so the discussion is very individual and a potential reuse of any questions then not quite so critical.

Architects Wiki: The architects wiki is available to all participants of the program, so not restricted only to the trainers. It is a structured area where a wealth of additional information is available and is actively used as a "living" knowledge base.

Having these core assets is not only a big productivity advantage but also helps to ensure that we have a common language and understanding across the various roles of the program and beyond.

B. Format

For all four roles (SSWA, SWA, SyA, TeA) we have a typical reusable pattern of multiple in-person off-site workshops with activities before, after, and between the workshops.

The overall pattern for the curriculum, including in the fullest extent the nomination process, the training itself, and the certification is shown in Figure 3 which shows the SSWA program (that has the fullest extent, with virtual certification gates even between the workshops).

The nomination process: for most of the roles there is a nomination process to ensure that the candidate has the sufficient technical and social skills to likely be able to complete the program; e.g. we expect several years of experience in working as an architect. The candidate must also have current architecture responsibility on a suitable project of organization-wide importance (due to "hands on" aspects). There are many parts of the architect learning program where the participant is required to apply what was learned in the training directly to their "real" project and not a "play" project. In the curriculum all of the projects and examples (a.k.a. "war stories") are based on real project examples, either those of the trainers or those of the participants. This helps not only the learning aspect, but also provides proven immediate benefit and impact for their projects/organizations.

There is a set of assessors from the business units (often managers and past participants of the program) and a process to ensure that there is no conflict-of-interest etc. This includes also more formal aspects like ensuring that the approach is approved by our workers' council etc.

The pre-training meeting: A virtual introductory meeting is held to get to know each other, clarify any open issues before the first in-person workshop starts. Usually, there is some homework assigned here to prepare and work on with their peers during the in-person workshop.

Within the SSWA program the pre-training virtual phase is an important part and runs over six months. Here the participants have explicit homework to do (project presentation, stakeholder presentation, architecture documentation, book presentation etc.) and already some sessions are done on social- and business-related topics.

The virtual preparation starts many months before the first workshop. After a "get to know you" workshop people work together on homework for their projects, do mutual stakeholder presentations, etc.

Participants take responsibility for their career development. In the workshops we uncover personal gaps. To close those gaps, they work on the MyLearningWorld (the modern learning platform for Siemens), pull literature from a Kanban board and present a summary of the content. With that approach we establish a focused collection of reading recommendations for architects.

In a comparable way we introduced such a virtual learning phase before the first workshop for the more recent programs, based on the experience from the SSWA program. This is

another example how experience is spread across the training programs.

The homework: Homework is assigned between the workshops. This either gives a course participant the opportunity to apply what they learned in the recent workshop in the participant's own organization or the homework prepares for the next workshop. Some examples are: create a domain model (SWA, SyA), discuss business case with business manager (SSWA), prepare a design essay (which is used to explain design tactics to SWAs), perform an active design review (SWA), clarify necessary architect's input for testing with the test manager/architect (SyA) etc.

The in-person workshops: These are highly interactive, with a mix of presenting know-how and many related exercises or experience exchanges among the participants. The trainers are highly skilled, with many years professional experience and very familiar with the training's storyline. A typical scenario is that

- the trainer provides a teaser, just enough information to bring all participants on the same track,
- then the participants are split into groups with a task to accomplish like discussing the topic in relation to their environment / daily work, summarizing their results on flip charts
- the groups present their results and the trainer takes care that all relevant topics are covered

The active involvement of the participants help that they learn the material more intensely and the cross-check at the end ensures that the most important aspects are covered.

Joint projects: Often all or most of the participants of a run will do certain bigger task together loosely related to the program. Often a run organizes a company-wide barcamps or a "hackathon" on particular topics to foster the understanding as well as the collaboration between architects within Siemens (for example a recent cross-company hackathon on applying artificial intelligence approaches).

Within the SyA program we regularly conduct hackathons to grow the knowledge base for system related design tactics. Here we collected design tactics for important qualities for the current participants, then split into 3-4 groups per run so that each has one quality to focus on, then each group spends 4-5 hours to create a corresponding design tactic collection, initially on a flip chart or moderation cards and/or via joint editing sessions – we meanwhile have 27 such collections.

Certification: We have a common process for doing the certifications, the actual realization is specific to the roles, e.g. for the SSWA there is an in-depth social skills assessment with psychologists and line managers, but there is a common approach with a pool of assessors who come from the R&D units and can determine the skill level in practice.

IV. CURRICULUM'S EVOLUTION

Starting from a Germany-based learning program the curriculum evolved over time into a globally distributed product family and was subject to continuous improvement.

The development of all 4 learning programs has been driven by a group of relevant experienced domain, discipline and learning experts. After the first program was available, the others were created based on a commonality / variability analysis, that identified intentional commonalities or even reuse of course concepts or content (core assets) and necessary differences. E.g. for the development of the system architect learning program an analysis was done what is the difference between software architects and (mechatronic) system architects. Two significant differences are, for example, from our software architects we expect that they can coach software developers in any aspect of software engineering. From a system architect we cannot expect a corresponding ability to coach the whole topic. Our systems may cover software, mechanic, electric/electronic, pneumatic, hydraulic, bio-chemistry, … there are no super heroes that are experienced experts in all the relevant disciplines. Therefore, system architects need to be able to involve and communicate with experts in all these disciplines. They need to understand terminology, typical issues and needs as well as the different cultures (way of work and thinking). Furthermore, once hardware is involved, there are additional topics to be taken into consideration, including production, logistics, service and maintenance (e.g. spare part concepts), refurbishment, disposal, etc. These additional post-development lifecycle phases will have an influence on system structure, cost consideration, development time, etc. These differences have a significant influence on the selection and development of course content. Nevertheless, there are also many commonalities. All architects need to understand the driving business case(s) and requirements and how to derive an architecture from that. All architects will lead (usually without formal power) teams and need appropriate social skills. In all places we intentionally use common terminology and methods if possible. Experience showed over years that this decreased misunderstanding between software, system and test architects, and vice versa improved communication and interworking between them.

In a "learning organization" every expert is a teacher and is able to convey their knowledge in courses and in the MyLearningWorld at Siemens. In the curriculum the trainers are experts from the business who deal with the content on a daily basis, and therefore they are very aware of major trends and the latest developments of the global software, system, and testing communities.

Participants, being experts in their own field, are encouraged to take over the role of a teacher passing their knowledge to the other participants so that they learn to become an essential part of their learning organization.

Before we invest in a big new content module, e.g. DevOps, Artificial Intelligence, Scaling Agile, with detailed training sessions, we start with a first small step with one session in one program to find out if the new potential "feature" is something which is of interest and relevance to the participants.

With positive feedback we incorporate that new topic into the program as a standard and, if important for other programs in the curriculum as well, this topic will become a reusable core asset. Participants of former runs are kept up to date via barcamp sessions which are organized on a regular basis.

Adding new content implies that older content either is taken out or addressed in some other way. These topics were moved to lower level programs in the curriculum pyramid, to virtual joint pre-workshop sessions the Siemens MyLearningWorld, or, if outdated, removed.

The certification part has stayed more stable, in part as our certification approach is not so oriented on knowledge by answering detailed technical questions, but more on capabilities and the application of knowledge, e.g how would one address a particular topic in their own specific context. Of course, certification content has been kept in-line with the changing course content.

V. EXPERIENCE

We have continuously gathered and learned from feedback or reviews of the training program over the past ca. fifteen years. This includes not only feedback from the participants themselves, but also their managers (typically software R&D heads) and also our human resources departments. Especially the feedback from the architect's managers has been interesting – here two direct quotes: *"I see the SSWA Program as the ideal vehicle for fostering and optimizing SW-Architecture competence in a sustainable way"* and *"The participant as well as our organization have highly benefitted from the SSWA program. During daily work I observe a more consequent and systematic focus as well as an enhanced view on overall system level"*.

Although it probably does not meet the standards of a scientific evaluation, we did conduct also a survey of the architects asking to what extent they saw improvement in terms of time, cost, and quality as well as additional feedback. Although both reduced time and reduced cost were also improved, the biggest benefit was seen actually in the quality of the resulting system. When asked to rate on a 5 point scale from -2 (much less) to +2(much more) structured way of working, 100% of the participants voted for "more" or "much more", 55% even for "much more".

The curriculum is also known also outside our company, e.g. from past SATURN conferences of the Software Engineering Institute (e.g. [2]). Furthermore, when Siemens conducted a benchmark on "software quality" with a set of global companies the Software Curriculum topic was rated as a best practice in this benchmarking.

VI. SUMMARY

The vision we had was that focusing on architecture is an effective way to improve R&D overall and that one can improve that through a holistic architecture-oriented qualification and certification program. After almost fifteen years and growing the program from the initial ideas into this ecosystem of curricula that continues to grow and evolve in a systematic way, we find that this has very clearly been achieved. Through the program we have had a significant positive impact on the products in a broad set of business lines and around the world.

Similar to "real" software architecture, maintaining the Software Curiculum's quality in a continuously changing environment with increasing speed of introduction of new technologies proved to be highly challenging. Following a lean and agile approach with continuous small "experiments" when introducing new training topics (of course supported by the participants feedback), we successfully managed to sustain the Software Curriculum's effectiveness and keep it updated.

Some additional benefits are 1) that the common understanding and language among the architect community helps strengthen the voice of the individual architects, 2) architects know each other and have a trustful person to ask advice of in an early and informal manner, and 3) that through the program we have strengthened the voice of the architects throughout the company.

We are more than ever convinced, that the training format, with a clear commitment to the in-person workshops as only through these can you really change the "mindset" is the right approach so that participants really take responsibility for their projects and are able to work in an agile way.

A key success factor of this curriculum is the growing mix of several approaches. The curriculum started with on-site workshops, reading recommendations and organized networking events. Over time we added the architects wiki as a common knowledge base (with read and write access by all course participants across all disciplines), we steadily add virtual sessions before and during the program partly even inviting alumni (in case of new content), we came up with hackathons to foster the knowledge base and to increase the understanding in innovative technologies.

Virtual sessions as well as the international setup helps to increase cost effectiveness, while keeping the quality high. Besides virtualizing some content allows to invite alumni in sessions with new topics and thus keep participants of former runs up to date. Alumni also have lifetime access to the most current course slides.

The cross-domain participation of course participants fosters experience exchange across domains and business units, spreading successful and harmful experience across the company. Thus, course participants not only learn from the trainers but also to a significant extent from each other, and these learnings are closely related to their daily work.

The fact that the trainers are active architects who work most of their time as architects in real projects rather than being full-time trainers helps to bring real-world experience to the program and makes the knowledge transfer easier.

The integration of the Software Curriculum into our companies' human resources activities is very helpful, both for getting the nominations of the most fitting participants as well as establishing this curriculum as a key part of the technical career path.

The extensive infrastructure for each course, including session slides, trainer guidelines, course material (poster and handout files and material for exercises), as well as documented workflows for the course logistics helps to keep the courses running in case of handovers to new stakeholders.

Finally, as with good architecture, we found it very beneficial to have an early focus on the structure and to explicitly consider how the program can evolve over time to meet the stakeholder needs

We sincerely hope, that by describing our industrial experience with others, we can discuss and share best practices with the international software engineering education and training community.

REFERENCES

[1] F. Paulisch and P. Zimmerer, "A role-based qualification and certification program for software architects: An experience report from Siemens", ACM/IEEE International Conference of Software Engineering, 2010

[2] F. Paulisch and J. Vaupel, "Software Curriculum @ Siemens Healthineers", Linda M. Northrop Software Architecture Award Presentation, May 2019, https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=546741