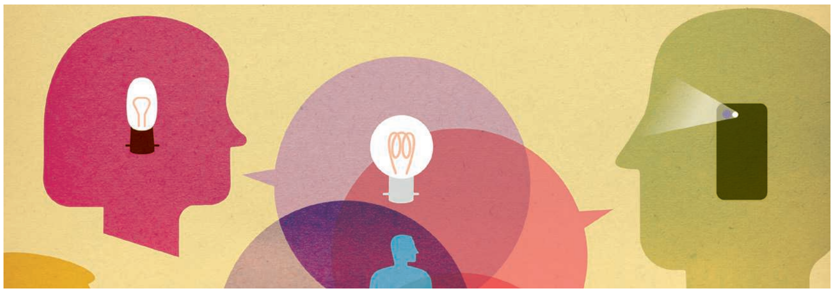
Lessons Learned From a Learning Program for Software Architects

Frances Paulisch, Matthias Backert, and Thomas Blum,
Siemens Healthineers

// *About 15 years ago, a role-based learning program aimed primarily at software architects was established and has evolved as a key part of our company's learning landscape. This article shares lessons that can be applied to readers' own learning endeavors.* //



TODAY'S WORLD IS changing quickly in multiple dimensions: technology is becoming a key part of our world, systems are becoming increasingly complex, and the environment is becoming more volatile and uncertain. At the same time, the quality attributes (also known as the "nonfunctional requirements") are growing, including attributes of the system like safety, security, availability, and performance but also development attributes such as maintainability and testability. Software professionals working on creating such systems not only face these challenges but also must be able to grasp and apply new concepts quickly. This also includes new technological approaches, processes, and business models such as continuous delivery and DevOps, ecosystems, microservices, machine learning, etc., but also how to apply these to the business situation and domain and in the context of long-living systems.

Especially with so many things changing, we find it important to have a strong focus on the architecture of the system and, consequently, the education and training of the architects. Because architecture is such a cross-cutting concern, this allows us to have the most business impact, avoiding the biggest potential problems and enabling, where feasible, a fast pace of change. The architecture focus is also necessary because design and architecture are known as "wicked problems,"[1] which implies that there is no single right solution, but one must consider different stakeholder perspectives and evaluate tradeoffs. Therefore, establishing a way of thinking about problems and learning to consider the alternatives and tradeoffs is an extremely useful approach, especially in the area of architecture.

Many of the large complex systems that are developed today are developed by teams including software engineers, product managers, and many other roles. Long past are the days of an individual developer focused almost exclusively on technology aspects. Alistair Cockburn in his book *Agile Software Development: The Cooperative Game*[2] and Kevlin Henney in his work "What Do You Mean?"[3] state clearly how

Digital Object Identifier 10.1109/MS.2023.3311267
Date of current version: 1 December 2023

very important communication, collaboration, and knowledge acquisition are. The lack of these frequently contributes to project failures. Therefore, these topics play an important role in our architecture-based learning program. The best technical approach cannot be successfully implemented if the architect is unable to convey the underlying ideas and concepts clearly to teammates and other stakeholders.

The History of Our Learning Program for Software Architects

In 2006 a cross-company core team at Siemens (including the healthcare part of the company) started working on this topic and soon realized that for the mentioned reasons a standard off-the-shelf training would not be sufficient for our purposes. Similar to how we develop complex products, we considered our business case, our stakeholders, and the nonfunctional requirements of our "learning program" product, and established a team with good communication and social skills to pilot and grow the learning program. Through a series of interviews and other means, we gathered information on the root causes underlying insufficient attention to software architecture, the characteristics of software architects who have been successful on multiple complex projects in the past, and the key success factors and pitfalls for software architects in the context of our high-quality, often safety-critical, complex cyberphysical systems.

Our initial focus was on the software architects working for products and programs where the impact of software architecture was the highest. These "senior software architects" (SSWAs), as we call them, contribute to products and programs involving systems of systems, product lines, ecosystems, etc. A (normal) "software architect" (SWA) is typically the responsible architect of a single complex product. After starting with the SSWA program, we later rolled out the SWA program. Subsequently, we applied a similar approach to roll out programs for test architects (TeAs), and system architects (SyAs) as well as for collaborating roles like product managers. In a previous paper,[4] we describe the set of architecture programs and how we used techniques such as commonality and variability analysis to determine which approaches are common across all the four programs and which are role specific.

After Siemens Healthineers became a separate company, most aspects of the SWA learning program were retained and further enhanced. Siemens Healthineers, who had been one of the drivers from the beginning, established some additional aspects, for example, ensuring that our trainers are experienced architects with excellent communication skills and the integration of the learning program with our personnel processes and career paths.

Related Work

Several publications, e.g., Galster and Angelov,[5] make the challenges of teaching software architecture clear. Lago and van Vliet, in their pioneering work on teaching software architecture in an academic setting, also refer to software architecture as being a wicked problem[6] partly due to the fact that there is no single best solution and that one must address the tradeoffs. de Boer et al.[7] similarly argue that since such wicked problems are not addressed well in more traditional active lecturer–passive student scenarios, a collaborative learning approach for them is more effective. More recently, van Deursen et al.[8] also describe a collaborative approach based on open source projects that enables real-world experience with both the necessary technical and social skills. We agree with the pedagogical approaches described by Jeff Offutt[9] that for software engineering we need to teach in a way that encourages "divergent thinking," i.e., that there is inherently no single right solution and that we should encourage collaborative learning as these approaches are needed by software professionals in the industry. A recent overview of a large number of software architecture education activities is given by Oliviera et al.[10]

In 2010, shortly after our learning program was started, there was a conference panel discussion including three company-specific software architecture certification programs (from Boeing, Raytheon, and Siemens) and two public programs (from IASA and the Software Engineering Institute) where the various approaches were compared and contrasted.[11] The closest broadly available practitioner training on software architecture we could find was the offering of the Software Engineering Institute.[12] However, we were particularly interested in an approach that included a broad set of topics also outside of software architecture, for example, including business topics, as well as a strong focus on communication and social skills. Furthermore, it was also important for the architects to be able to work on their own projects. For these reasons we decided to "grow our own" learning program for software architects.

Software Architecture Learning Program

In this section, we share the main aspects of our software architecture learning program at Siemens Healthineers.

The aspects that differentiate our program from other programs described in the literature are that our architecture-based learning program

- is focused on professional practitioners in a real-world industrial setting applying their learnings in their current projects
- is aligned to the needs in our context characterized by high-quality cyberphysical systems with a broad range of quality attributes (performance, security, safety, reliability…) and very long life spans
- is embedded in a company context (e.g., personnel departments, career paths, certification process, and visibility in the company)
- has a strong focus on the active network of the company-internal community (architects, trainers, assessors…)
- is established as the architecture training for the whole company (all business units and global)
- has a long (over 15 years) history in industrial practice.

Despite these points that make our program unique, we are confident that many of the insights we share in this section could apply also for other practitioner-oriented software engineering learning programs.

Role-Oriented Competence Management

We take a very systematic approach to learning. We invest effort into the detailed role description of an architect, and this includes which competences they need at which level of expertise. As shown in Figure 1, in the software architecture learning programs for SSWAs and SWAs, there are a number of competence areas, and for each it is clear whether a basic (knows about), advanced (can apply), or expert (drives) ability is needed in each topic. This allows us to address a broad set of topics (at least at a basic level) but also to drill down to hands-on activities for some, e.g., expert-level topics.

Timeless Problem-Solving Approach

Furthermore, we focus on a level and style of learning that transcends,

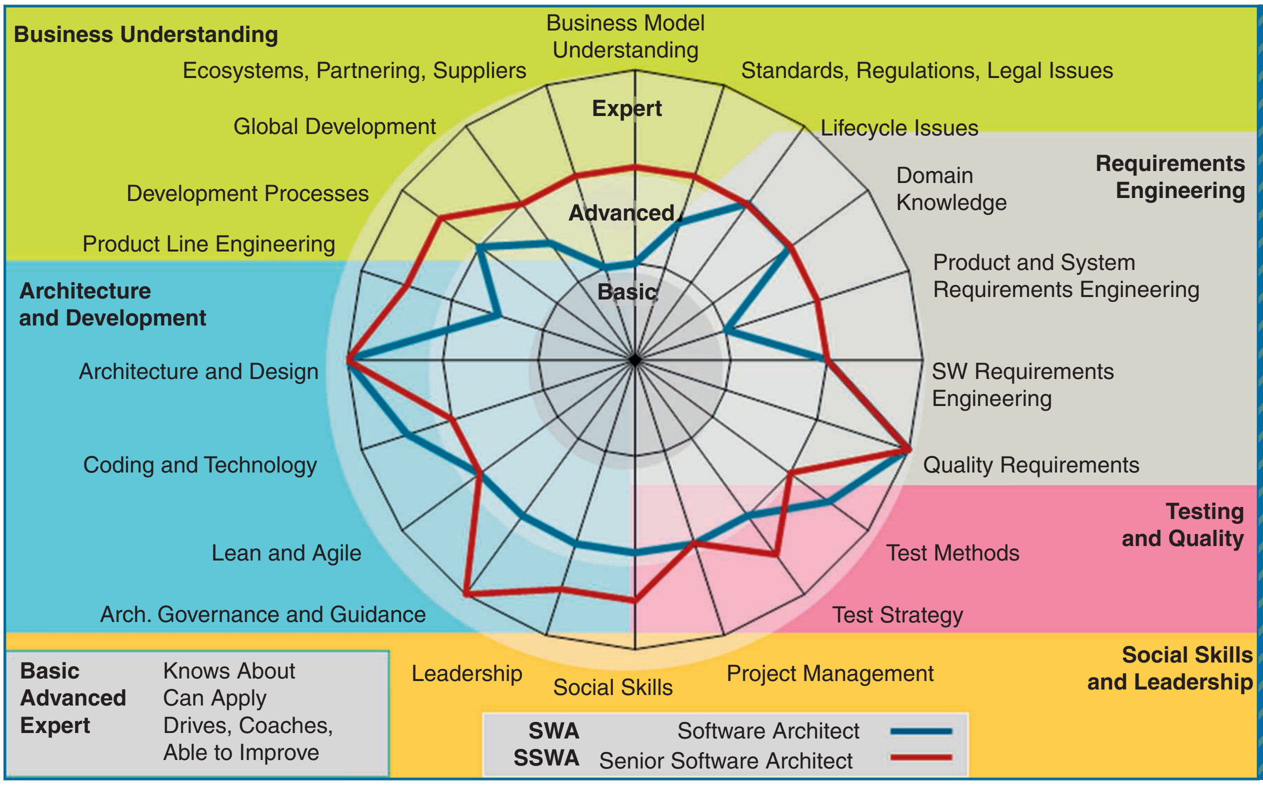


FIGURE 1. The mapping of depth to competences for SSWA and SWA.

for the most part, the detailed technology aspects. The content is very hands-on but at a level of abstraction that could apply to many different technologies. The learning program hones the ability on how to think about problems and evaluate the tradeoffs appropriately and always in the given context. For deeper insights into various technical solutions, we recommend separate modules or refer to external content. To bring new content into the programs, we correspondingly move content that is otherwise common knowledge or can be learned in other ways out of the program. This approach naturally allows us to be able to keep the "essence" more stable over the years but also gradually evolve the content.

Hands-On on Own Real Projects

One of the main reasons we decided to build our own program was that the external training material we saw typically used generic examples to show various techniques, and the participants often worked on those generic examples. For us, with our very complex, cyberphysical systems, it was very important that the participants work hands-on with their concrete projects. Only individuals who serve as the main software architect of a real, current, and adequately complex project are eligible to join the program. This ensures that all participants in the runs have a similar level of challenges. When the participants get "homework," such as an essay on how they handled tradeoffs between performance and cybersecurity, they do this based on their own project, which makes it very hands-on and real. Learning is typically more effective when one actually applies the knowledge in a project rather than just hearing about it in a lecture, and this is another reason that the direct application in the participant's own projects is so important.

Furthermore, this close connection to the project is very important also for the acceptance of the significant time the architects spend in this learning program. If they were "only" doing training on some abstract, generic "toy" example, they would not have a positive business impact on the project during the training. However, with our hands-on approach and always connecting back to the participants' real work, they see immediately how they could apply what they are learning.

Holistic Set of Topics, Not Only Architecture

The content of the program, by design, goes well beyond learning about software architecture. As shown in Figure 2, architecture is one of the four main topics; the other three are business understanding, requirements engineering, and testing/quality. Across all topics and with a focus of about 40% of the time is the topic of "social skills."

Especially SSWAs learn to work not only in the system but also on a broader organization-wide view on the system.[13] They thereby improve processes, remove organizational obstacles, and encourage new business models, for example, by introducing DevOps or ecosystems.

Communication Ability Is Key

Our strong focus on the ability of software architects to communicate well is due to the modern understanding of an architect's role as a central person who is not only a technical expert but also a skilled communicator. Architects are often a kind of "living bridge": a technical communication hub to other roles to also ensure a common understanding. We often refer to the "driving triumvirate" (see Figure 3), which includes the architect (representing the interests of R&D), the product (life cycle) manager (representing the product requirements), and the R&D project

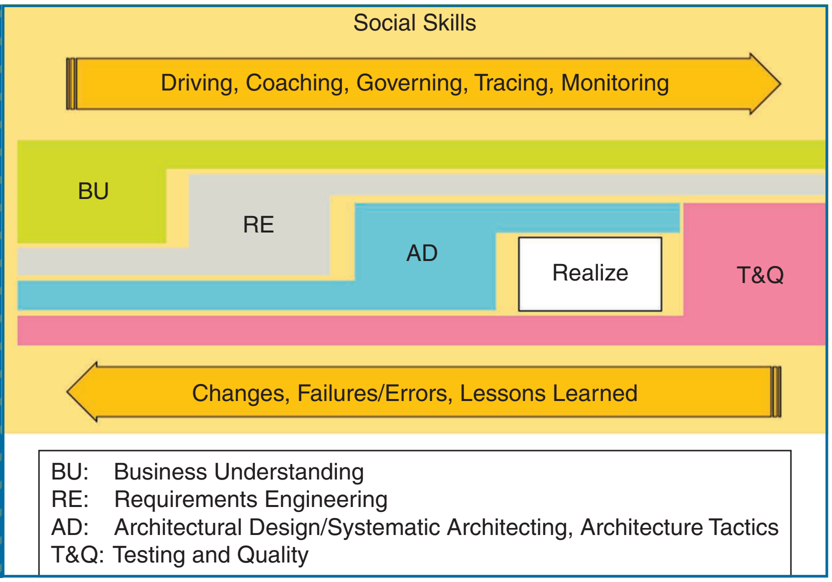

Social Skills
Driving, Coaching, Governing, Tracing, Monitoring
BU
RE
AD
Realize
T&Q
Changes, Failures/Errors, Lessons Learned
BU: Business Understanding
RE: Requirements Engineering
AD: Architectural Design/Systematic Architecting, Architecture Tactics
T&Q: Testing and Quality

FIGURE 2. The four plus one topic areas of the learning program.

manager/product owner (representing the realization of this product). Furthermore, in many cases, architects need to lead the team even if they do not have the authority to actually tell other persons what to do, so their skills in communicating and convincing are particularly important. Moreover, the "how" is as important as the "what." An architect could be absolutely right in their view, but they must be able to consider the perspective of the persons they are communicating with and be able to convey the message in a way that the other person can understand and subsequently find an agreement. In fact, the skill of achieving appropriate compromises is particularly valuable and is nurtured by practice on concrete examples throughout the learning program.

Agile Approach

Salza et al.[14] share many approaches for applying agile approaches in education and training. We apply agility in our product development, and so it also comes naturally to us to apply it also in our learning approaches. By applying an "agile learning model," we prod the participant to uncover their blind spots and close gaps and coach them not to fall back into old patterns and behaviors. Following self-organized learning, the participants are responsible for their learning outcomes, and they have a high degree of freedom regarding the format (how) and time (when) for learning. Here the agile "pull" principle is very helpful. The student pulls the learning resource or person that helps most in his/her situation. At least 30% of the content is adapted to the current needs of the group, e.g., by inviting additional experts on a topic. On our way to becoming a "learning organization," every expert shall be able to transfer his/her knowledge. So participants also learn to teach each other and their organizations also in innovative formats such as BarCamps.

Highly Experienced Trainers

The persons attending these trainings are already very experienced software architects and typically have a strong self-esteem. If the architecture part of the training was led by a communicator who is skilled but who lacks practical architectural experience, the participants would not be so receptive to the trainer's advice. This is one of the reasons we find it essential to have highly experienced hands-on architects from real projects as the trainers: this makes the message they convey, and the additional real stories they can tell about their real-life experience, much more vivid and convincing to the participants.

Sustainable Architect Community and Mentoring

We actively encourage the participants of the learning program to "give back" to the architect community, e.g., especially suitable persons who have gone through the SSWA program are often asked to be architect trainers for the SWA program. Within their own business lines, these architects are explicitly expected to mentor junior architects to help groom them to become candidates for the learning program. Furthermore, we have a number of cross-company events focused on architecture and a regular company-wide meeting of software architects at Siemens Healthineers. We explicitly invest effort in networking activities to continue to grow and promote the community throughout the company. This community of is often asked for their perspective on important cross-company topics related to software or to conduct an architecture assessment on an important project.

Alignment With the Personnel Department

At our company the program is also well aligned within the career landscape for software professionals. It is a normal part of the regular career review process to identify and nominate potential candidates for the learning program. In addition to

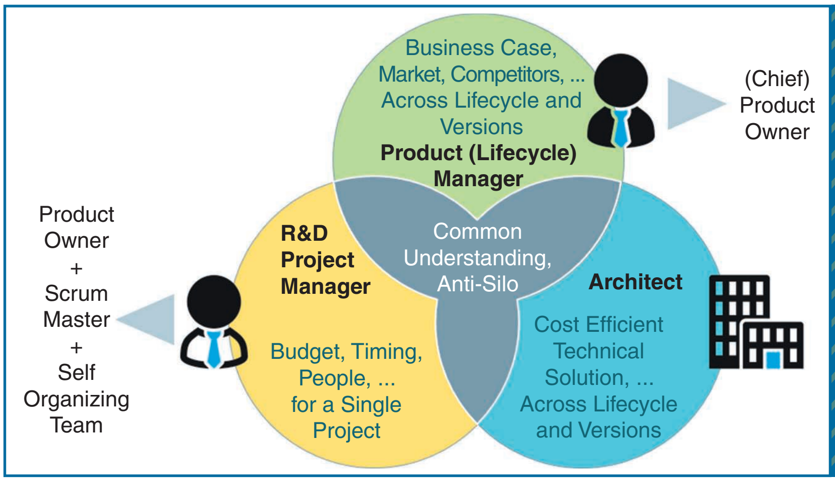


FIGURE 3. The "driving triumvirate" of the R&D project manager/product owner, the product (life cycle) manager, and the architect.

the learning program itself, there is an associated certification process through which the architects are evaluated in an interview setting on theory, practice, and social skills to be formally recognized as "certified" architects. Furthermore, establishing the certified architect as a typical case for an "expert career" helps keep the experts in the company.

It was very important to us to really be able to systematically evaluate that the candidate has successfully applied the knowledge gained in the learning program.

There is an explicit nomination process for the learning programs, with the most stringent prerequisites for the SSWA program. For the SSWA it requires, for example, a recommendation letter, typically from the head of the relevant business unit, and an official application including information on the project of which the candidate is the architect. There is an entry interview, typically with an experienced SSWA and a software manager, and all of these together set a high bar to enter the program. This helps to ensure the needed commitment to go through the program, as the participation in the program and the associated homework can easily take 20% of the architect's time. A further benefit of this filtering is that it helps that all accepted participants are experienced and active architects with high qualifications. It is quite often the case that candidates for the program are excluded as they are not yet sufficiently qualified or they are not the responsible architect of an appropriate project.

Deciding whether to have a formal "certification" for this learning program was one of the most important decisions we made when initiating the program. As you can imagine, it is quite challenging to set up a certification process that is effective and fair and can stand the test of time. This is probably also the most effort-intensive part of the program for the part of the organization that provides the certification.

It was very important to us to really be able to systematically evaluate that the candidate has successfully applied the knowledge gained in the learning program. Other external certifications on other topics are often based on multiple choice questions. We instead have a certification process much more focused on demonstrating to the assessor that the candidate knows how to apply the knowledge in practice on their project. For the SSWA the participants must pass a set of "knowledge" gates in various topic areas during the program itself. In the knowledge gates they have to show how they applied their knowledge in their own projects. Finally, close to the end of the program, they must also pass a "capability" gate where they have a structured interview with a set of assessors including an SSWA, a software manager, and a soft skills/psychology expert. Typically, after passing the certification, the organization ensures that the additional skills learned in the program are taken full advantage of when determining future goals and responsibilities. For example, it is common that the software architecture of our important products is driven by certified SSWAs. Further information on the certification topic for our learning program is provided by Paulisch and Zimmerer.[15]

Global Reach

At Siemens Healthineers we have about 40 certified SSWAs and 160 certified SWAs, from a pool of ca. 5,000 software engineers. Although the certified architects represent only a small fraction of the entire population, their influence is very strong throughout the company and across the globe, not only within their respective business lines but also as a common voice across the company when discussing our software strategy.

We offer the SSWA learning program only in one location approximately once a year (typically ca. 12 participants). The SWA program is currently offered in India, Germany, and the United States. The other two programs, SyA and TeA, are offered only in one location.

Online Only Versus Online Virtual Versus in Person

For our global setup, there have always been discussions about how much of the learning program content we handle through online means and how much in person. Especially since COVID-19, we had no choice but to go online, but we are now increasingly back to a mix of formats. We strive to ensure that at least one of the typically three training sessions is in person, and in some cases, all of them are in person. Certainly, online virtual instructor-led formats are much more effective than online only. The trainers need to know how to teach differently in these three kinds of formats. Especially in the online virtual setup, one must explicitly do a number of activities (joint virtual cooking, for example) to get to know (and trust) each other.

Be especially wary of your training departments telling you "online only is much better." In our experience, one must differentiate. For some topics, self-paced learning using online-only formats and web-based training are very suitable. Furthermore, technical or pure "knowledge" topics can often be effectively done online. However, if your training aims to change the way of thinking, to change the culture of your organization, to change the "mindset," then this is best done through collaborative and in-person training. If in-person training is not possible, then at least online virtual instructor-led training is more effective than online only. It may seem, initially, that that in-person approach is the most expensive approach. However, especially in a fast-changing world, such an investment in learning pays off. This is especially true of trainings for the software architect roles as this role has a huge impact on the suitability and success of the resulting product.

ABOUT THE AUTHORS

FRANCES PAULISCH is the head of the Software Initiative at Siemens Healthineers, 91301 Forchheim, Germany. Her research focuses on coordinating strategic topics related to software across the company including training and the sharing of best practices on a broad range of topics. Paulisch received a Ph.D. in software engineering from the University of Karlsruhe. Contact her at frances.paulisch@siemens-healthineers.com.

MATTHIAS BACKERT is globally responsible for the learning programs in the area of software at Siemens Healthineers, 91058 Erlangen, Germany. He is one of the architects and trainers of the Software Curriculum portfolio and responsible especially for agile, requirements engineering and social skills topics. Contact him at matthias.backert@siemens-healthineers.com.

THOMAS BLUM is the lead software architect in the area of magnetic resonance devices at Siemens Healthineers, 91052 Erlangen, Germany, He is a certified senior software architect (SSWA) and the main architect trainer for the SSWA learning program. Contact him at thomas.blum@siemens-healthineers.com.

Experiment With New Ideas

We regularly experiment trying out new approaches, and if they work out, then we add them into the program (and face the tough decision of what to take out instead). Often, we suggest a run of the SSWA program to organize a company-wide event. As described by Backert et al.,[16] this has often recently taken the form of a "hackathon" typically on new upcoming technologies and trends like applying artificial intelligence and machine learning. This is a great way to get broad involvement from enthusiastic software engineers throughout the company and to have a little competition further liven the discussion. Such events were not only fun but had an extremely high "learning to time" ratio. Most recently, the run organized a BarCamp in India, which included about 70 software engineers in India and about 70 further online around the world. They addressed a broad range of topics, and the essence was captured and shared across the company.

The software architect learning programs have an over 15-year history, which in the days of software is a very long time. As such, one can learn from how those foundations were built, for example, applying approaches that we use for our medical devices with long life spans and also to our training-related "products" such as this set of learning programs. Similar to how our products are very diverse but also have many elements in common, our set of ca. 250 architects are also strong individuals but, through the learning program, now share a common language and a more systematic way to think about challenges.

We can only report here qualitative results about the impact. In a company-internal survey asking architects who had completed the program at least a year ago, 85% stated that they had a "more" or "much more" structured way of working. The managers of the architects often report that the architects come out of the program with a noticeably different mindset and also see that the program has added value in the daily business. Both the architects and their managers say that they benefit from the established cross-company network of architects from different domains but with similar problem-solving techniques.

For readers who would consider to establish or further grow their training programs in the software area, we hope that we have provided useful insights on our experience that can also be helpful to your endeavors. The biggest challenge is to be willing to make the necessary investment in time and resources to initiate and continuously evolve a learning program. We wholeheartedly believe that the investment in the training of our personnel is of extreme importance and especially so in these fast-changing times.

References

1. H. W. J. Rittel and M. M. Webber, "Dilemmas in a general theory of planning," *Policy Sci.*, vol. 4, no. 2, pp. 155–169, Jun. 1973, doi: 10.1007/bf01405730.
2. A. Cockburn, *Agile Software Development: The Cooperative Game*, 2nd ed. Reading, MA, USA: Addison-Wesley, 2016.
3. K. Henney. *What Do You Mean.* (2019). Kevlin Henney. [Online Video]. Available: https://www.youtube.com/watch?v=ndnvOElnyUg
4. M. Backert, T. Blum, R. Kreuter, F. Paulisch, and P. Zimmerer, "Software curriculum @ Siemens – The architecture of a training program for architects," in *Proc. IEEE 32nd Conf. Softw. Eng. Educ. Training (CSEE&T)*, Nov. 2020, pp. 1–6, doi: 10.1109/CSEET49119.2020.9206182.
5. M. Galster and S. Angelov, "What makes teaching software architecture difficult?" in *Proc. 38th Int. Conf. Softw. Eng. Companion*, May 2016, pp. 356–359, doi: 10.1145/2889160.2889187.
6. P. Lago and H. van Vliet, "Teaching a course on software architecture," in *Proc. 18th Conf. Softw. Eng. Educ. Training (CSEET)*, Apr. 2005, pp. 35–42, doi: 10.1109/cseet.2005.33.
7. R. C. de Boer, R. Farenhorst, and H. van Vliet, "A community of learners approach to software architecture education," in *Proc. 22nd Conf. Softw. Eng. Educ. Training*, 2009, pp. 190–197, doi: 10.1109/cseet.2009.10.
8. A. van Deursen et al., "A collaborative approach to teaching software architecture," *Proc. ACM SIGCSE Tech. Symp. Comput. Sci. Educ.*, Mar. 2017, pp. 591–596, doi: 10.1145/3017680.3017737.
9. J. Offutt, "Putting the engineering into software engineering education," *IEEE Softw.*, vol. 30, no. 1, p. 96, Jan./Feb. 2013, doi: 10.1109/MS.2013.12.
10. B. R. N. Oliviera, L. Garcés, K. T. Lyra, D. S. Santos, S. Isotani, and E. Y. Nakagawa, "An overview of software architecture education," in *Proc. Anais Do XXV Congresso Ibero-Americano em Engenharia de Softw. (CIbSE)*, Jun. 2022, pp. 76–90, doi: 10.5753/cibse.2022.20964.
11. P. Clements, "Certified software architects," *IEEE Softw.*, vol. 27, no. 6, pp. 6–8, Nov. 2010, doi: 10.1109/ms.2010.137.
12. "Education and outreach," Softw. Eng. Inst., Pittsburgh, PA, USA. Accessed: Jun. 3, 2023. [Online]. Available: https://www.sei.cmu.edu/education-outreach/
13. C. McGoff, *The Primes: How Any Group Can Solve Any Problem.* Hoboken, NJ, USA: Wiley, 2012.
14. P. Salza, P. Musmarra, and F. Ferrucci, "Agile methodologies in education: A review," in *Agile Lean Concepts Teaching Learning*, D. Parsons and K. MacCallum, Eds. Singapore: Springer, Oct. 2018, pp. 25–45.
15. F. Paulisch and P. Zimmerer, "A role-based qualification and certification program for software architects," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng.*, May 2010, vol. 2, pp. 21–27, doi: 10.1145/1810295.1810300.
16. M. Backert, F. K. Jeberla, S. Kumar, and F. Paulisch, "Software engineering learning landscape: An experience report from Siemens Healthineers," in *Proc. 4th Int. Workshop Softw. Eng. Educ. Next Gener.*, May 2022, pp. 43–50, doi: 10.1145/3528231.3528356.