Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 827–833
DOI: 10.15439/2014F257
ACSIS, Vol. 2

Experience with Real-Life Students' Projects

Jaroslav Král
Masaryk University in Brno,
Botanická 68a, 602 00 Brno, Czech Republic
Email: kral@fi.muni.cz
and
Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
Email: kral@sisal.mff.cuni.cz

Michal Žemlička
Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
Email: zemlicka@sisal.mff.cuni.cz
and
University of Finance and Administration
Estonska 500, 101 00 Praha 10, Czech Republic
Email: michal.zemlicka@post.cz

***Abstract*—Student software projects are often focused on training coding skills and on model-driven software system design. The projects rarely develop skills needed for the proper formulation of system visions and requirements specifications. To solve this issue the projects must deal with real-life software projects issues. The projects should solve main commercial aspects of real-life – they must include looking for project topics in practice and there should be possible to communicate and collaborate with future project users. Successful projects should be rewarded (optimally paid) by the users like other commercial products. We discuss here the quite successful experience with a "prototype" implementation of the concept.**

I. Introduction

THE SOFTWARE development is a risky process. The proportions of failed and challenged projects is about 1/4 and 1/2 respectively [1], [2], [3]. It is known that it is due to poor quality of project visions (aims) and project requirements specification. Late stages of software development processes are seldom the source of the issues.

The development issues are besides the management failures caused by underestimation of the complexity of the vision and specification stages.

A detailed analysis of the problem indicates that the development team members are in fact not aware enough of the software engineering aspects of the projects, i.e. that the development of software systems is a technical (engineering) problem. It holds, surprisingly enough, not only for users (project sponsors and project stakeholders inclusive) but also for IT experts taking part in the projects. They all must be aware that a seemingly simple requirement need not be implemented and used due the technical aspect easily.

Our experience from academia as well as form industry shows the reasons for failures and challenges in software projects are often caused by IT experts unable to effectively and properly take part in project vision and requirements specification.

Young people are often excellent in coding. It especially holds for young people active in theoretical disciplines. They know very well that their coding skill is difficult to overcome. This fact is overemphasized. Soft software development skills are difficult to develop. It holds especially for the people trained in hard skills needed for academic or scientific career. They are excellent coders but usually not excellent in design and especially in requirements specification. They are often proud on their intellectual abilities and scorn the users being unable to write and test programs. They underestimate or disregard the importance and the complexity of real-life problems. They are unable to take part in such agile software development processes when they must take users as partners. They even do not admit that they should take part in reqirements specification. They are unable to tune specifications in cooperation with users. We will discuss our experience with student projects aimed to solve the above issues at IT side. The projects were a part of postgradual studies.

The issues can be solved if IT students take part in team projects covering very early stages of software development – marketing, vision formulation, and requirement specification. The students should cooperate with prospective users of the developed systems. It implies that the students project can fail. We know that real-life projects fail frequently.

We will discuss here the structure of student projects we have used to solve the issue and present experience with them.

The paper is structured as follows: First, we discuss the problems discussed above in details. Then we describe two variants of the projects intended to train analytical skills. Finally, we summarize the experiences gained in the projects.

II. The Soft Facts on Project Failures and Challenges

The proportion of failed projects is remarkably stable, the proportion of failed projects remains very high for decades – see the surveys Why Projects Fail [4] or the Chaos Report [3]. The reasons are not solely at project stakeholder and project management side.

Our experience indicates that the issue is a bit more difficult and complex. People involved in software system development, maintenance, and use are not aware enough that software systems are complex technical entities requiring engineering attitudes. All developers, IT experts inclusive, are not ready to apply software engineering knowledge, attitudes and processes. IT experts are then not able to convince users

978-83-60810-58-3/$25.00 © 2014, IEEE

and managers that the application of software engineering is necessary and that requirements specification is no straightforward matter.

We must conclude that the project failures are in this sense due to IT experts more frequently than it is presented in project failure analyses. It is quite difficult to change this attitude as developers, especially coders, dislike developing and applying needed skills. Model-driven architecture and related attitudes are helpful partly only.

Our experience with the students' projects indicates that our students like and are able to propose an elegant solving of technical problems. They are often not interested in seemingly boring problems of real life. They, for example, are able to design a very complex SQL statement but they are not aware that a small modification of it could be substantially more usable in real-life projects.

We also experienced that people tending to be managers underestimate the fact that a more detailed technical knowledge could be very useful for them too. It concerns especially the cases when managers make decisions having important technical consequences.

III. Education and Analytical Skills

We can conclude that the education and mastering of soft knowledge and skills necessary for analytical tasks in software development is of growing importance but it does not meet needs. IT students and even some of their teachers do not understand the importance of the issue. They often consider analytical documents to be something like gossips whereas the only valuable thing is a big piece of (good) code. This attitude is supported by the fact that young people are often very good coders.

The poor understanding of the importance of the early stages of software development is the crucial reason of lower quality of analysis oriented student knowledge. The issue is further strengthened by the fact that the needed development skills are not trained enough. It leads to the lack of good project management and to poor results of agile methods of software development.

We discuss here possible ways of solving this issue. The training of necessary social skills is a key issue here. The training can be realized only if the requirements specification and, if possible, the vision and market analysis are trained in students' team projects. The goals of the projects should be the development of small information systems or the development of their autonomous parts.

The projects should be designed so that they meet the needs of real-world people and in collaboration with them, i.e. they should be small commercial projects. The development process should have as much common as possible with standard development of information systems in small software firms – market analysis, business, and technical risks inclusive.

We present below the main principles of the implementation of the above requirements and the experiences with the projects at two Czech universities. We further discuss issues of our attitude to be still solved. The most important problem of the projects is the trend to the fragmentation and extreme specialization of scientific knowledge and of education processes, and a growing gap between hard a soft knowledge of our alumni. They are not trained to be good project managers as recomended, e.g., by Ebert [5].

IV. Information Systems and Professional Knowledge

The basic goal of the postgradual study of IT experts is the education of professsionals being able to propose, develop, maintain, and execute software, usually information systems. The development of such systems are enhanced variants of classical waterfall model [6] consisting of:

1) Analysis done in cooperation of the system developers with future system users: by formulation of visions (answering the question "why?" and outline of the basic requirements) and by requirements specification,
2) design,
3) coding,
4) testing:
   - unit testing,
   - integration testing,
   - function testing,
   - system testing;
5) integration,
6) deployment.

The scheme can be enhanced and modified in various ways, see iterative development (agile development, scrum [7], XP [8]) and incremental development (various SOA methodologies [9], [10], [11] inclusive). The variants of development can be viewed as repeated and integrated waterfalls.

Knowledge and skills needed in the initial stages of software development are often missing. It is known [3], [12] that the user involvement and management issues are crucial for the success of information systems development. The faults in these stages of the development can cause more than 80% costs spent to remove development errors in not failed projects. Almost half of software projects significantly overrun expenses and terms.

The overall losses caused by the errors are significantly higher. For example conceptual errors are the main reason of most project failures. According to [3], [13] it happens to about $1/4$ of projects. These misconceptions usually stay behind poor maintainability of software systems and behind their early or permanent obsolescence [14].

There are yet other snags: The conceptual errors can hiddenly lead to a user discomfort or can even reduce their long-term wellness feeling and even their health. It reduces productivity and worsens social climate in firms. It destroys firm culture and threatens its future.

Coding and testing are usually without substantial issues. The education of the skills needed for coding and testing is therefore successful.

It is the reason for looking for new methodologies and development paradigms like the concept of SOA ecosystem by

OASIS. The stages 1, 2, and partially 5–7 require according [15], [1], [3] and the experience of software developers:

- collaboration of developers with users,
- support by management,
- requirements specifications specific abilities of the involved team,
- etc.

It requires a collection of soft skills and, what is more important, the skills must be trained.

The students should take part in the early development stages of projects designed so that they are "approximations" of real-world projects usually for small to medium enterprises (SME). The knowledge (often even sole information) collected in books need not be sufficient. The needed knowledge should be broad open and flexible to be usable for the specification of user needs.

Some services in SOA can be developed (may be as prototypes) as students projects. It is also possible to use sprint subprojects of SCRUM or XP methodology. It is a crucial proposition as real world projects must be able to bring some real world effects. They should incorporate the vision and even some marketing. It is a very hard task in academia environment. We must cope with the fact that such projects can face real-world risks. The existence of the risks contradicts the concepts and organizational principles of the IT education. It, together with the trends in the academic research, leads to the overspecialization and fragmentation of student knowledge. It further tends to grow contempt of social and generally soft knowledge at the students' side. The study plans must cope with the fact that some of our projects may fail.

V. Basic Requirements on the Real-World-Like Projects

The analysis of the problems of our alumni is strongly restricted by data privacy protection. The data collection requires explicit approval of the students with the data collection and use. Even in the cases when alumni are successfully contacted, that their approval will not cause selective/choice effects. Based on the facts and our experience with students projects there can be made the following conclusions:

The students overestimate the importance coding skills. They are partially right as they are in such skills peerless. One perfect coder (people from the top 5% quantile) can replace some 20–30 average coders ([16], [17]). It leads to some conceit and underestimation of soft and social skills, to unacceptable team attitudes and to some negative aspects of hacking – see the Corncob Antipattern [18].

It is more important that such individuals strengthen the feeling of exclusivity and underestimate the necessity of good relations with users and ability to understand their knowledge and needs. It is not easy to communicate with them what is strengthened by the fact that they underestimate knowledge outside IT. Persuasion that it is neither good nor simple is usually a hopeless task. The only way is to create situation showing explicitly that they are not right.

Let us note that social and organizational skills are getting obsolete significantly slower than the knowledge and skills necessary for coding where within 5 years about one half of them get obsolete. Partial solution of this issue is possible if project aims are properly selected. We have tried two main variants of the projects:

1) Projects having features typical for SME/SMB projects. They train especially the skills of vision setting and requirement specification.
2) Projects solving partial tasks in software development in and for large enterprises. Such projects improve skills necessary for cooperation in such environments and enable understanding the environment and its processes.

VI. Involvement of students in commercial projects of large software vendors

Large vendors, e.g. IBM, have started a close collaboration with some Czech universities. The collaboration is implemented as the engagement of selected students in development teams of the vendors. The students often take part in the development of information system of the commercial partners as assistant analysts.

The students successively take up relevant roles in analysis and negotiation with a vendor client. He/she is induced to use vendor policies, processes, and tools. The students often take also part in design, especially in model building. In the cases when the students prove themselves useful they can be rewarded and have quite often opportunity to become gradually the employees of the firm.

The firms hire in some sense the students. It follows that it is applicable for some students only. The results and experiences with the concept are surprisingly good. The students are often warmly accepted by the enterprise team members. They are pleased to collaborate with the students.

The development of excellence centers supported e.g. by Europe grants enables further enhancement and a broader use of this concept. It simplifies the involvement of start-up firms and open new possibilities for master and dissertation projects.

The main advantage of these forms of studies is the training of analytical and design skills in the real world environment (ecosystem). The main disadvantage is that the students do not come in contact with the formulation of visions (aims) and marketing.

VII. Small to medium business oriented projects

Small-to-medium enterprises (SME) are a frequent domain where our alumni get good jobs. Most of our alumni get job in SME. SME need people having skills enabling to take part in all phases of software life cycle starting with market analysis, looking for a client, vision statement, negotiation, business agreements, and requirements specification.

These skills must be trained. We have trained the necessary skills in projects designed and implemented in the following way:

1) The projects are implemented as team seminary works being an optional part of a postgradual study. There are usually about 20 seminary participants.
2) The participants are required to propose themes of possible projects. The themes should have the potential to be commercially successful. The student must therefore, using their contacts market analysis formulate project proposal containing project name, the specification of possible user(s), and a (abstract of) vision. It must be based on the analysis of a real-word organization (partner).
3) The proposals are presented and defended.
4) The most promising proposals are chosen and the participants form teams having three to seven members.
5) Each team choose tools (e.g. an information system) supporting its work. An open source/free project support systems are preferred. The team members present their professional profiles, CVs and experiences.
6) The project name and abbreviated name is fixed. A project logo is welcome. The visions are refined and included into project home page. Achieved results are presented and defended using data projectors. The presentation should include the testable project effects.
7) The teams develop requirements specifications and system models. The involvement of the partner experts or intended end users is welcome.
   a) The teams can use open source software or free systems.
   b) The integration of the system into the partner ERP is highly appreciated.
   c) The team must do market analysis, especially the detection and study od the properties of similar systems.
8) There are at least two progress reports for each project. They have the form of reviews in the software engineering sense.
9) The final project defense at the end of seminary consists of:
   a) Presentation of the proposed system capabilities in the form understandable for users.
   b) Review of the technical properties a concise description of development processes, prototypes, and models.
   c) Interesting experiences.

The projects cover the software development stages at least up to design. There should be a well-founded hope that the project development could be continued and implemented on commerce basis (payment inclusive).

VIII. Schedule of the Seminary

A. Initial Meeting

- The teacher specifies the aims (vision) of the seminar. He/she shows that the main challenge of software project, especially of the project at SME, is the poor formulation of aims/visions and requirement specification of software project. The challenges are strengthened by the fact that the aims and requirements specifications are seemingly easy to understand and therefore easy. It is not the case but to understand the problem the students must take part in real-life projects needed collaboration with people from practice (future system users).
- A seminary support system specified (offered) rules of seminary students communication are specified in cooperation with the lecturer and the student.
- The overall schedule is specified see the points below.

B. Looking for Project Topics and Possible Business Partners

- Looking for Project Topics and Possible Business Partners
- Parallel activities:
  - Students present themselves, their experience, knowledge, and skills. Aims:
  - training the skills applicable during job seeking and for taking part in seminary project

C. Presentations and Competitive Choice of Project Proposals

- Choice of project leaders. The leaders are usually the students bringing the successful project proposals. They usually serve as project contact people. Involvement of the teacher in the choice is possible but not preferred.
- The leaders choice team members.
- The roles in the teams are tentatively defined.

D. Aims and Main Specifications

The aims and main specifications should be formulated in cooperation with business people – tentative users of the system.

E. Organizational Data

- Project identification;
- Project topic;
- Team structure and team roles – minimal team size is 3, maximal 7 members;
- Project supporting system (open source);
- The chosen development tools;
- Semiformal risk analysis.

F. Initial Outputs of the System Development

- the system full and short names;
- the specification of business partner;
- tentative vision.

G. First Presentation of the Project

First presentation of the projects (after a month) is in the form of a review:

- Presentation of project homepage.
- Main outputs.
- Control of the contents and documents in project support system, especially the team meeting reports, reports from negotiation with partners, and other project reports.

H. Closing Customer-Oriented Presentation

- presentation of project home page;
- presentation of goals/visions and final capabilities;
- possible use of prototypes.

I. Closing Technical Presentation

- Project schedule;
- Derived technical models (diagrams);
- Experience, positive and negative aspects.

J. Final Session

- Evaluation of the projects and students activities;
- Informal evaluation of individual students;
- Common personal $+$ and $-$.

The structure should be accommodated to the expertise of the students. Sometimes happens that there are students being professional software development team or software company leaders. In such cases the other students can take significant advantage of it.

IX. Issues Common for all Team Student Projects

Some students' projects can be done individually but in a team it is possible to collect some additional experience.

A. The Team

The students should build the teams independently. The students should fulfill their other study duties in parallel to the project. The team therefore faces the risk that some team members could be busy with other study duties or that their study can be for other reasons terminated. It can lead to one of the following situations:

- The students can close the ranks and start to help each other also in other (not project related) duties to keep the team at full strength.
- The students start to protect themselves by decomposing the project into relatively autonomous parts (or even by choosing projects that are naturally divided in this way) implementable by individuals. It allows them to show that they fulfilled their partial duties. Sometimes the finished components could provide at least basic functionality of the desired system.

We succeeded in some projects to set up the policy that every team member got assigned some development labor (preferably several tasks) in both primary and secondary role in pair programming. The role of a buddy (secondary developer in the pair) means that the person is informed what the primary developer does and why it does and that it could be asked to consult some issues or even to help. Such teams were quite reliable even when a team member was for some (even longer) time out of service (due other school duties or an illness).

A creation of such pairs requires some minimal tolerance and mental compatibility of the involved people. It is moreover advantageous, is a person has different shadows for his/her different tasks or if it plays a shadow of someone else than he/she is shadowed. If someone get into troubles, the shadow could take his/her work over and can delegate some of the shadow's work to his/her own shadows. The load for individual people than could grow by a part only what gives better chance that the person will be able to finish the work without induced troubles.

It has also approved itself if the project has been designed to change its extent (it is, there was a kernel that must be finished and multiple extensions that were optional – developed when nothing bad happened). Such projects could be finished successfully, reliably, and without fatal rushing.

Creation of a good team and tuning the extent of the project is a nontrivial task. It shows that it could take even several weaks yet before the project officially starts.

Side Effects

Working on a project may bring further benefits for the students:

- a deeper familiarization with the topic handled by the project;
- mastering tools for team work;
- act and negotiate with people (it is necessary to get on with other team members as potential team break could usually harm all its members);
- often also the discovery that a teacher can be true even if the student is thinking that he/she know it better (and then it shows that the hint given by the teacher and rejected by the student(s) could save a lot of work and avoid a lot of troubles).

B. Experience with the Projects

We practiced "real-life" student projects for eleven years at two Czech universities in two different cities (Prague and Brno). There were 4–6 projects a year at each university. There were interesting trends. The students in Prague tended to prefer coding. It caused the falling interest of the students in taking part in the projects. There are no active projects of this type in Prague now. There are less visible reasons for it. For example, a limited number of small enterprises developing information systems in Prague and a strong preference of theoretical computer science. Last but not least, it is possible to finish the study in other study branches with less effort and risk.

The trend in Brno has been towards a broad use of free or open software systems (project management tools, modeling tools, documentation tools) used to support the vision, requirements specification, and the team organization. It substantially enhanced the quality of the project processes. There were no project failures provided that their initial defences were successful. The reason was that the project vision must be successfully defended at the beginning of the project. An unsuccessful defense implied an immediate project cancellation. It was rather an exception.

The best projects in Brno were further positively influenced by the following aspects:

- There are many medium-sized firms in Brno employing the students.

- Some students have their own firms.
- There is a well-working collaboration of big firms (e.g. IBM) with Brno universities.
- SOA and Scrum methodologies can provide enough small-to-medium real-life projects.
- Some projects (usually 1–2 a year) were not only accepted and partly paid by the software firms but they were enhanced, commercially finished, and used for several years.

C. Students Projects and Curricula

The students' projects introduce many issues. They are of organizational, financial, and juridical nature. Let us mention some of them:

- Inclusion of project supervision into teachers load (good project supervision takes significantly more time than what corresponds to assigned/scheduled hours).
- Taking time complexity of the project into curricula design (to avoid time coexistence of critical parts of the project with other crucial study duties).
- Maintenance of successful projects. Good projects should be continued after the project evaluation. The issue here is that the evaluation is long term and laborious and the students that developed the application or system have also other study duties. It can also happen that they leave the university. Such projects could be used to train maintenance. There is a risk that an improper maintenance could break the well-working system.
- Turning the project into its commercial phase (in the case of its success) – requires having a procedure making from study result a commercial project (it includes transfer of the rights between the university and the company that will care about the project further). Some universities have this procedure stable and simple, for other universities it is a very hard (if not impossible) task.
- Not every teacher/lecturer is able to properly supervise students projects. It is necessary to find them and to prepare them. It requires some skills and experience that not every professor must have.

X. Conclusions

Majority of our student projects were quite successful. Their participants were as a rule able to find viable topics, contact people from firms, form student teams, find and apply team work supporting software, develop needed diagrams, and present the results. Some projects led to successful commercial products. It was probably partly due to the fact, that almost all the students were part-time employees of software firms. As a valuable byproduct the students discussed their experiences from the firms where they were employed.

The main contribution of the seminaries is a better understanding of the importance of a proper combination of soft and hard knowledge by all students, not only by the ones taking part at the seminaries. The information is spred spontaneously via social networks It is appreciated ex post by our alumni being in practice for several years.

Our seminary model has been successfully applied at Faculty of Informatics, Masaryk University Brno and partly at Faculty Mathematics and Physics, Charles University Prague.

We can conclude that the seminaries described above were successful. The weak point is that their concept is difficult to be applied massively. An implicit precondition of project success is that some of the seminary participants are quite excellent programmers and that some of the students have moreover a broader knowledge and social skills. It follows that they were good in technical abilities and STEM (science, technology, engineering, and mathematics) knowledge. It is a challenge as the STEM education becomes less popular and often not properly taught. It moreover can negatively influence soft knowledge needed in the project as it must take into account some aspects of STEM knowledge. The deteriation of STEM education is a threat for the quality of coding and for the education of coders. They moreover tend not to work together with users and to use user-oriented knowledge domain.

XI. Future research

The present curricula induce the fragmentation of education processes. It is then very difficult to organize long lasting student projects as well as scientific projects. There is a stronger challenge. Current principles of evaluation of universities and their professors based on impact factors prefer very strong specialization of research, narrow knowledge areas, and short term projects. It handicaps multidomain knowledge and skills needed in system analysis and requirements specification.

It is opened how to combine the training of technical skills needed for SME with the training of the skills for large software vendors. It main issue is the difference in enterprise culture, resources, and needs. It follows tthat many aspects of our projects must still be tuned. The idea is, however, crucial as otherwise there were be lack of good project managers having economic, social, as well as technical knowledge and skills. There is a danger that otherwise our alumni will become laborers (line workers) at Scrum-based duplicate production.

We intend to develop methods for development of quite complex systems using multiple student projects. We believe that it is possible if a specific SOA architecture [11], [19] is used. We intent to use open source and free software and combine it with commercial commodity software offered by large software vendors (e.g. Microsoft Excel, Microsoft PowerPoint, or OpenOffice Calc). We will attempt to apply the experience of small Czech software firms here.

References

[1] Standish Group, "The chaos report," 1994, [Online:] http://www.ics-support.com/download/StandishGroup_CHAOSReport.pdf; accessed 2014-02-28.

[2] ——, "Chaos: A recipe for success," 1999, [Online:] https://www4.informatik.tu-muenchen.de/lehre/vorlesungen/vse/WS2004/1999_Standish_Chaos.pdf; accessed 2014-02-28.

[3] ——, "Chaos manifesto 2013: Thing big, act small," 2013, [Online:] http://versionone.com/assets/img/files/ChaosManifesto2013.pdf; accessed 2014-02-28. [Online]. Available: http://versionone.com/assets/img/files/ChaosManifesto2013.pdf

[4] Calleam Consulting, "Why projects fail," 2014, [Online:] http://calleam.com/WTPF/?page_id=1445; accessed 2014-02-28. [Online]. Available: http://calleam.com/WTPF/?page_id=1445
[5] C. Ebert, "Software product management," *Software, IEEE*, vol. 31, no. 3, pp. 21–24, May 2014, DOI: 10.1109/MS.2014.72.
[6] W. W. Royce, "Managing the development of large software systems," in *IEEE WESCON Proceedings*. Institute of Electrical and Electronics Engineers, Aug. 1970, pp. 328–338.
[7] M. Cohn, *Succeeding With Agile: Software Development Using Scrum*. Addison-Wesley Professional, 2009.
[8] K. Beck, *Extreme Programming Explained: Embrace Change*. Boston: Addison Wesley, 1999.
[9] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "Reference model for service-oriented architecture 1.0, OASIS standard, 12 October 2006," 2006. [Online]. Available: http://docs.oasis-open.org/soa-rm/v1.0/
[10] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, 2005.
[11] J. Král and M. Žemlička, "Implementation of business processes in service-oriented systems," in *2005 IEEE International Conference on Services Computing (SCC 2005)*, vol. 2. IEEE Computer Society, 2005, pp. 115–122, DOI: 10.1109/SCC.2005.58.
[12] J. Martin, *An Information Systems Manifesto*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984.
[13] M. Levinson, "Recession causes rising IT project failure rates," *CIO Magazine*, Jun. 2009. [Online]. Available: http://www.cio.com/article/495306/Recession_Causes_Rising_IT_Project_Failure_Rates_
[14] P. Armour, "The reorg cycle," *Communications of the ACM*, vol. 46, pp. 19–22, Feb. 2003, DOI: 10.1145/606272.606288.
[15] I. Sommerville, *Software Engineering*, 9th ed. Pearson Education, Apr. 2010.
[16] G. M. Weinberg, *The Psychology of Computer Programming*. New York: Van Nostrand, 1971.
[17] B. W. Boehm, "Software engineering economics," 1981.
[18] W. J. Brown, R. C. Malveau, H. W. S. McCormick, III, and T. J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. New York: John Wiley & Sons, 1998.
[19] Open Group, "Open Group standard SOA reference architecture," Nov. 2011. [Online]. Available: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12490