Paper ID #36526

Schedule Risk and PERT in Undergraduate Capstone Projects

Michael Van Hilst

Dr. Van Hilst is an Associate Professor of Software Engineering at Embry-Riddle University in Prescott, Arizona. Prior to that he taught at Nova Southeastern University and Florida Atlantic University. Dr. Van Hilst entered academia after an extensive career in industry. He worked for 10 years at the Harvard-Smithsonian Center for Astrophysics where, as senior architect, he worked on NASA's Einstein, Hubble, and Chandra space telescopes. He also worked at IBM Research, for the French CNRS, and at HP Labs, where he was a member of the Software Technology Lab. While at FAU, he consulted for Motorola, over many years, on improving their development processes. Dr. Van Hilst has two bachelors and a masters degree from MIT. He did his PhD at the University of Washington under David Notkin.

Reginald Paul Parker (Professor)

Schedule Risk and PERT in Undergraduate Capstone Projects

Introduction

The senior capstone project is a rite of passage for students of engineering. It serves as an introduction to "real world" project experience before they go out into the real world. It is a time for students to learn nuggets of wisdom like, "Spec sheets always lie," and "There is no such thing as a problem, only challenges."

A well planned and executed student project should be finished, tested, and working at the end of the semester, by the expected date of completion. But, as any faculty member will tell you, things do not always go according to plan. The shortfalls are often attributed to unforeseen circumstances, personal failures, or simply running out of time.

Risks and the management of risks are part of the capstone experience. But traditional approaches to risk management do not address the most common kind of risk found in student capstone projects, namely, risk to the schedule.

In this paper, we propose an approach to schedule risk management, based on industry practices, designed to address schedule risks in student capstone projects, such as optimistic estimates, knowledge gaps, and unexpected difficulty. This work is part of an ongoing effort to improve both the learning experience, and outcomes, of capstone projects in our university.

In the next section we review the existing literature on risks in capstone projects, both risks that occur, and practices for risk management. The section on PERT starts with a comparison of the origins of PERT and CPM, to illuminate the issues we raise and the gap we seek to address. The remainder of the PERT section describes the original PERT practices specifically designed for projects with low experience and high uncertainty. The following section, on Risk Reduction describes 6 different measures that can be taken to address risk and uncertainty when the risks are identified on a per-task basis. The next 5 sections consider risk management alternatives found in Spiral, Agile, Knowledge Factory (Lean), FMEA, and JAD practices from industry. The section with our Discussion, borrows from all 7 sources of scheduling and risk management practices to propose an alternative and more suitable set of practices specifically to address risks to the schedule in student capstone projects. The final section contains the Conclusion.

Current Literature

In 2014, Vanhanen and Lehtinen [1] surveyed papers on capstone projects searching for discussions of risks. Out of 67 papers discussing capstone projects and the issue of risk, only two looked at the risks that actually occurred (Ahtee [2] and Koolmanojwang [3]).

Ahtee and Poranen [2] surveyed the final reports from 76 student capstone projects in a Computer Science department and counted the number of times different kinds of risk appeared in the report. Scheduling problems were the most common. "Projects were not able to follow their initial schedule, and they had to change the schedule unexpectedly," often by reducing in scope. The second most cited risk involve using new tools and languages, where either through learning difficulties or actual experience, the team found themselves less productive than they had hoped. The third most common risk was hardware that failed or did not meet expectations.

Koolmanojwong and Boehm [3] discuss risks and risk management in a graduate level Software Engineering project course. Design issues, technical knowledge gaps, COTS issues, and time and budget constraints had the highest frequency of occurrence.

Vanhanen and Lehtinen [1] studied 11 capstone projects to understand the types of problems that occurred. The top failures were that the teams fell short of their goals (both in features and quality), communications broke down, and students didn't "take responsibility." In the discussion, the authors referred to poor quality estimation, high learning needs, and poor motivation. Student motivation becomes a problem when heroic effort is the only option for success.

More recently, Makiaho and Poranen [4] compared the risks identified up front by students in 7 capstone projects, with the risks that actually happened. The students had been advised to follow the practices defined in Sommerville [5] and the PMBoK [6]. All 7 teams identified extrinsic events up front, like sickness and members quitting. In the final reporting, 5 of the 7 teams experienced running out time, of which 4 viewed it as unforeseen.

Capstone projects have fixed schedules, limited resources, and fixed personnel. A better strategy for capstones should address their biggest sources of risk: knowledge gaps, unrealistic schedules, and tools and components that do not meet expectations.

In Traditional Project Management (TPM) scheduling starts with breaking the project into smaller tasks in a Work Breakdown Structure (WBS), defining dependencies between tasks, and estimating the expected duration for each task. Risks are treated separate from schedule development, in a way that leads students to focus on extrinsic events. Sommerville [5], Wysocki [7], Hoffman [8], PMBoK [6], PRINCE2 [9], INCOSE [10], and even SEI [11], all give advice similar to that shown in Figure 1. While there is sometimes discussion of reducing or avoiding risk, few specifics are given. The focus is detection and response.

1. Identify the risks.
2. Assess the risks in terms of probability and impact.
3. Prepare plans to respond to risks should they happen.
4. Monitor to detect risk occurrences.
5. Apply the planned mitigation strategy in response to occurrences (with possible follow-up analysis).

Fig. 1. Commonly Suggested Steps for Risk Management

Alternatives to TPM have been put forward, notably Boehm's Spiral Model and Agile practices. A Spiral project [12][13] is divided into three or more cycles, where the first one or two cycles are designed to reduce and, ideally, eliminate uncertainty and risk. As presented, choosing what to address in the first cycle, and how (typically through prototyping), depends heavily on expertise and experience.

Agile methodologies [14][15][16][17] have their own practices for dealing with uncertainty and change. But are plan-driven, which means that the rate at which work is done, is driven by requirements rather than the schedule. Given that capstone projects must be completed on a fixed date, this can be a challenge. Even with sprints, the capstone project must have some idea of the overall schedule to set the scope and anticipate needs.

In our search for strategies to address the schedule risk in capstone projects, we also looked two older practices, FMEA and PERT. Both have a compelling focus: address the risks early rather than wait for them to happen. Failure Mode and Effects Analysis has been around since 1949 (MIL-STD-1629). Originally applied to military contracts and NASA, it has recently been adopted in both the automotive (SAE J1739)[18] and aviation industries (ANSI/AIAA S 102.2.4). Its key feature is that risks are identified for each element in a project breakdown, rather than for the project as a whole.

PERT was developed in the mid 1950's, about the same time as TPM. In fact, WBS is a PERT practice. But PERT was developed specifically for projects with large knowledge gaps, schedule uncertainty, and components that had never been tried before.

In the remainder of this paper, we will look more deeply at all four of these practices, starting with PERT, and present a proposal for combining elements of each to create a more realistic risk management strategy to address the schedule risks in undergraduate capstone projects.

PERT

CPM and PERT were both developed for project scheduling in the late 1950's. The Critical Path Method (CPM) was developed by DuPont for maintenance and construction projects on chemical plants. They had an experienced construction force of 3000 who built plants all the time. To them, "any construction activity had an optimal, or normal, way to be performed – a preferred method of given crew, duration, cost, etc." [19]. They knew how long every activity would take. The only risks were from extrinsic events, which could not be controlled.

Program Evaluation Review Technique (PERT) [20] was developed by Booz, Allen, Hamilton for the US Navy Special Projects Office to build Polaris missile submarines. No one had ever built a submarine to launch ballistic missiles, let alone, launched a missile from underwater. For many of the activities, they did not, and could not, know how long it would take. PERT scheduling was later used by NASA for the Apollo moon project.

Student capstone projects have more in common with building the first Polaris submarine, with a team of engineers short on experience, than they have with building the 150th DuPont chemical plant, using the same crew that built the 149th. Traditional Project Management adopted names and notations from PERT. But the risk management and scheduling practices follow CPM.

In PERT planning, risk exposure is assessed on a per-task basis. Task durations are estimated using three-point estimation: most likely (M), optimistic (O), and pessimistic (P). Even when you cannot reliably assign a "normal" duration, you can provide reasonable bounds and an estimate that accounts for risk.

The PMBoK includes the original PERT calculations, (O+4M+P)/6 for the weighted mean and (P-O)/6 for standard deviation. Most likely means the time with the highest probability – the mode. They asked experts, "under normal circumstances, how long do you think it should take?" They assume a beta curve, where bad luck has a longer tail, and chose one that offered the above simple approximations for mean and standard deviation. (See Figure 2.) Mean is a better choice than mode for scheduling. The means are used to calculate critical path, while the standard deviations give the probability of project completion by any given day.

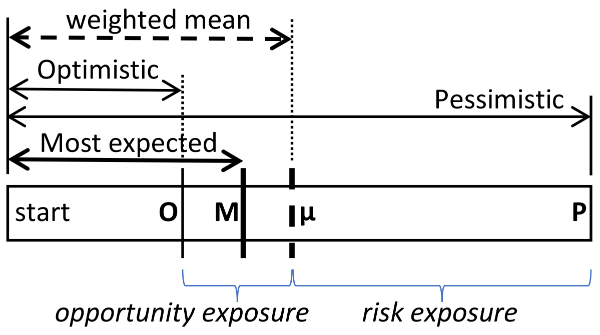

Fig. 2: Task duration based on optimistic (O), most expected (M), and pessimistic (P)

The probability of the calculated mean is 50%. For O and P, different sources suggest using 1%, 5%, or 10%. Too close to the extremes invites things like miracles and asteroid collision. The original PERT Phase 1 report [20] simply said, "barring acts of God." For students, who are not experts and have no experience of "normal", it might be better to ask directly for the mean, rather than the mode. The question is then posed as a better's wager. What value would you choose for the expected, such that, if repeated, 50% of the times it takes more and 50% of the times it takes less?

In risk management, exposure is defined as the cost times the probability. In a traditional risk meeting, putting a value to risk probability and costs is difficult. For schedule risk, PERT simplifies that task. The probabilities are the same in all tasks, while the cost is time, which is given. For whichever estimate is used, the risk exposure is P-E, while the opportunity exposure is E-O. P-O can be used directly to rank risks, i.e., in a tornado diagram. (See Figure 3.)

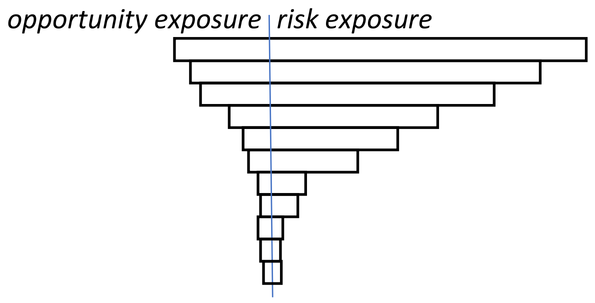

Fig. 3. Tornado Diagram showing tasks ordered by exposures.

Where in the schedule an exposure occurs also makes a difference. If a risk occurs early in the schedule, there is more time available in the schedule with which to compensate. Leaving a risk exposure late in the schedule makes it far more likely to have an unrecoverable negative impact. When working on the Chandra Space Telescope at Marshall Space Flight Center in the mid 1980's (the same office that ran Apollo, 15 years earlier), the first author was told, "move your exposures to the front."

The PERT Phase 1 Report [20] suggested laying the schedule out from back to front based on predecessor-successor relationships, summing the time and variance to completion along the way. Then, going front to back, choose which tasks to schedule first, based on the earliest start date and largest variance to completion. The result was a schedule that moved exposures to the front.

The PERT method of estimation makes it easier for students to identify exposures for every task, and then, using whatever scheduling method they choose, moving the exposures to the front. In the next section we describe an expanded collection of risk reduction activities, many of them only possible when risk assessments are made at the task level.

Task Based Risk Reduction

Once the development risks are localized in specific tasks, a variety of measures, some of which would not otherwise be available, all become available to reduce the risk and increase the likelihood of project success.

Proactive scheduling: The risky task can be scheduled earlier in the project such that should it cost extra time, adjustments can still be made. The original PERT did this by prioritizing paths with high-risk tasks. In Agile practice, a task can be moved forward before some things it depends on, by splitting it into two tasks, the part that can be done now (possibly with temporary scaffolding), and the rest to be completed later. The later task is classified as "technical debt."

Substitution: The risky task could be replaced with a different task involving less risk. As an example, if the task specifies a tool or language which no member of the team has ever used, then perhaps the task could be redefined to use a more familiar, though perhaps suboptimal, choice. A similar strategy is Boehm's Simplifiers and Complicators (S&C) tool [12].

Parallel development: If the task involves a feature that is high value, but also high risk, it could be run in parallel with a task to create a similar feature, but with lower value and less risk. At the end of a fixed period (e.g., one or two iterations of either an agile or spiral process) a decision must be made about which version of the feature to continue. Toyota used this technique to good effect when they developed the Prius. In the beginning, they made plans for both new technology regenerative brakes and traditional drum brakes, without being committed to either. Part way into the project, the regenerative breaks proved viable, and the uncertainty around that work was removed [21].

Spike: To reduce the uncertainty, design a task to specifically address the missing knowledge as quickly as possible. If you do not know how to do something, such that that part of the project is not ready to be worked on, create a different task to create and test the missing knowledge. On the Chandra Telescope, when an inexperienced vendor won the contract for an important piece of equipment to be delivered late in the project, we ordered a do-nothing item from the same vendor to be delivered early, with features likely to cause the kinds of problems we were worried about. Sure enough, there were many things they had not known about space hardware which they then learned, and the uncertainty appearing late in the schedule was greatly reduced. Boehm [22][13] talks about putting this kind of work in a prototype. In Agile terminology, Kent Beck coined the name "spike" – a simple task aimed at answering a question or gathering information. Spikes are also described in Leffingwell [15]: "Spikes, another invention of XP, are a special type of story used to drive out risk and uncertainty in a user story or other project facet," and in Al Hashimi [23].

Outside help: If there are others with experience and expertise for the activity in question, it may be cost effective to hire a consultant, contract it out, or simply ask. For students, knowing what questions to ask, and understanding the relationship between an issue and their schedule, should motivate them to seek timely advice.

Item Elimination (scope reduction): The item, and its associated tasks, could be removed. A decision could be made that the feature involved was not sufficiently important to be worth the risk to the project. If scope must be reduced, best do it early.

Where the risk involves the acquisition or use of a third-party component or tool, the strategy should be to order it early enough to gain experience and validate, with the option to go with plan B. For hardware, the strategy might involve ordering two of them in case one is damaged while learning its proper use and limitations. In either case, the testing should be early enough to obtain a replacement, or implement plan B.

Spiral

Boehm [12] and others have suggested a risk-driven iterative Spiral development process for capstone projects. In a Spiral process, the complete development lifecycle is repeated several times, starting with simulations, prototypes, or parts of the system, with the goal of addressing risks and uncertainty early on. The first cycle addresses feasibility questions, followed by one or two cycles in which the remaining risks and uncertainty are removed, leading ultimately to finish the product in a smooth flow. The literature is a little vague about how to choose the activities for a given iteration. "Relying on risk-assessment expertise, the spiral model places a great deal of reliance on the ability of software developers to identify and manage sources of project risk" [22]. PERT's task-based risk assessment fits this model well, by contributing to early and specific risk identification, and offering a richer set of strategies to address them within the schedule. Progress between iterations can be measured by the reduction in exposures and variance.

Agile

The Agile model consists of a series of timeboxed iterations, called sprints. The initial objective is a "minimum viable product" after which successive sprints add customer value. The tasks, to be addressed in the next iteration, are chosen based on value vs. effort for the current state of development. Each sprint starts with a planning meeting, where a customer representative or product owner chooses the priorities for the next iteration.

Students may see the Agile focus on customer value as justification to do the easy visible stuff, first. That makes sense if requirements change is the biggest driver of risk. But for projects where schedule and technical risks are the prime concern, priorities should be based on impact on the schedule. As a consultant in industry, the first author saw this choice firsthand. On projects where task priority was based on customer value, and not schedule impact, analysis of the event logs showed considerable churn and rework at the end, from things that should have been done much earlier [24].

The task-based assessment of exposure during schedule estimation, described here, makes it feasible to flag tasks with exposures for priority treatment. When a high risk task cannot be done early, the risk part should be addressed as a separate spike [23][14].

When students measure progress in terms of tasks completed, artifacts created, tests passed, or code written, reducing risk by testing prototypes, completing tutorials, and finding workarounds, does not register as progress. Agile practices measure progress not in terms of work done, but in terms of work left to do, using a burndown chart. Students sometimes confuse being busy with making progress. Burn-down charts avoid that. A similar concept addressing remaining effort and ETC (Expected Time to Completion) can be found in the PMBoK.

With PERT scheduling there can be two lines on a burndown chart, one based on expected time to completion and one for remaining uncertainty (variance or exposure). Reduction in variance is measured by both completing tasks with variance and reassessing three-point estimates after acquiring knowledge. Venture capitalists (VC) impose early milestones for the elimination of "important ambiguities or knowledge gaps" [25]. The variance milestones represent greater confidence and fewer risks going forward. Charts, like the cone of uncertainty, show declining uncertainty, though only figuratively (see Figure 4). In our approach, it can be measured.

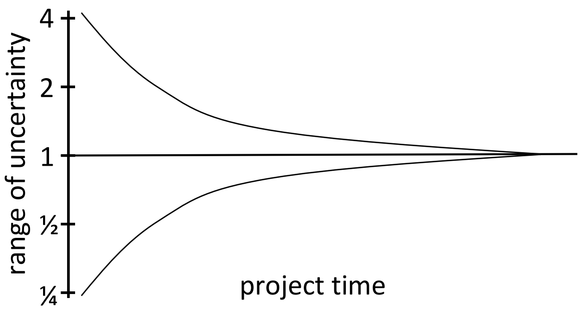

Figure 4. Cone of Uncertainty

Knowledge Factory

Another antidote, to the over-emphasis on artifacts, is called "knowledge view" or "knowledge factory." Patterson [26] recommends viewing product development as an information assembly line. "It is a process for adding value to the information set as rapidly as possible until that information set describes the manufacture, support, and use of a quality new product." Schwaber and Beedle [16] also present a knowledge creation view for Agile. "We acquire knowledge that we eventually capture in the code or in an executable model." Gaining knowledge, even from a failed test, represents progress. Patterson recommends the use of value-stream-mapping techniques (a Lean practice) to optimize the knowledge value-add in every task.

With a knowledge view, development is equivalent to answering a series of questions. Planning is the process of deciding which questions need to be answered and in what order. Code is one form of capturing the answer to a question. So too are design documents, specifications, or evidence from a prototype. Filling a knowledge gap by reading a manual or completing a tutorial also counts. It can be helpful, at the beginning of the capstone project, to ask students to write down the questions that need to be answered for the project to proceed smoothly to completion.

FMEA

FMEA (or Failure Mode and Effects Analysis) is a more rigorous version of risk management than the TPM practices described above. First defined in 1949 (MIL STD 1629) and used by NASA, more recently it has been adapted across much of engineering, e.g. SAE J1739 (surface vehicles) [18], IEC 60812 (electrotechnical), and ANSI/AIAA S102.4-201x (aerospace). FMEA applies risk management to hierarchical decompositions of the system, bottom-up, and throughout the project life cycle. Different forms of decomposition can be used as appropriate (functions, parts, subsystems, tasks, etc.) In its most recent form [18], FMEA applies to both design and process. It also considers interfaces between things; elements that each "work" may not work together, or the hand-off from one to the next may fall short.

FMEA is considered process heavy, because, when done correctly, it can involve a significant amount of documentation. There are specific forms to complete and, because it applies to the many elements in a breakdown, there is more to analyze. But the things that are documented are what you would expect in a risk analysis – the type of failure, its effects and likely causes, severity and likelihood of occurrence, and measures to address it.

The risks, that capstone students experience, are generally process failures. PFMEA [18], is a form of FMEA specifically for processes. PFMEA works with flow models, not unlike the Gantt charts and network diagrams already created in capstone courses. While PFMEA is usually applied to manufacturing, it can be applied to the capstone process using a knowledge factory view of the capstone process, where each task fills a specific knowledge gap, leading to the eventual creation of a proof-of-concept product.

JAD

JAD (Joint Application Development) is a facilitated project planning activity developed at IBM in the late 1970's to rapidly complete the requirements analysis and initial stages of planning. It is used quite widely, including in both Agile methodologies for building systems with rigid time constraints, DSDM [17] and SAFe [15]. In one or more JAD sessions, the major stakeholders gather, figure out, and communicate what to build. It involves using sticky notes and Affinity Diagramming to choose what to do, design how to do it, identify "killer" issues, and modularize the work to create a schedule. JAD meetings promote shared learning and ownership, and are well suited to small groups, making it ideal for developing the plan and schedule in capstones.

Discussion

While students focus on many kinds of risk, the risk they experience at the end of the capstone is not finishing on time, at least not finishing what they had planned to finish. The goal of this work is to revisit the question of capstone project management, by looking, beyond Traditional Project

Management (TPM), for practices and methodologies especially suitable for high risk, high uncertainty engineering projects, with a particular focus on risks to the schedule. As the previous sections show, there is quite a bit from which to draw.

The work is motivated by three keys insights: 1) that when risk is assessed at the lowest task level, significant options to address the risk, beyond "monitor and respond", become available, 2) that the original PERT practices of schedule estimation provide a method of risk assessment (albeit specific to impact on the schedule) that is both easier to apply and more sound, than the traditionally proposed approach to risk analysis and assessment, and 3) the knowledge factory view of development facilitates the application of practices, normally associated with industry and manufacturing, to student capstone projects. Risk management is applied in the same process as schedule development, where it should be.

To give the students a meaningful and productive experience, we propose to apply risk analysis similar to the current engineering practice of FMEA, perhaps even using some of the same forms, but narrowing its application to the immediately applicable, and easier to address, problem of risk to the schedule.

In the past, students made estimates for task duration that were wildly unrealistic, and then pretended they were sound. When their project did not live up to expectations, they often blamed themselves, treating the problem as a shortcoming. In this redefined view of combined schedule estimation and risk management, we hope that students can see their own lack of experience, skill, and knowledge as a normal part of high risk, high uncertainty projects, that needs to be built into the schedule. Once these issues are on the table, estimation can be more realistic. The uncertainty is expressed as bounds which are easier to estimate and less sensitive to error than a single time estimate, while providing valuable information about the corresponding risk. We propose the use of burn-downs, borrowed from Agile and Spiral, to track and manage both progress and risk across the life of the project.

We envision student teams having planning sessions, similar to JAD planning sessions, to address both schedule development and risk planning. We have already conducted such sessions in multi-discipline projects, using sticky notes for tasks, and yarn for the dependencies. The yarn is labeled with the knowledge and artifacts expected to satisfy the dependence. In the time estimation phase, placing of upper (P) and lower (O) bounds allows students to discuss personal limitations (both time and experience), as factors to be addressed. Where warranted, a Delphi method can be incorporated, adding to both the quality and the experience. Unaddressed knowledge gaps should result in new tasks, while tasks with large exposures should be addressed using one or more of the strategies outlined in the section on Task Based Risk Reduction. As in Spiral, and shown in the PMBoK cone of uncertainty, students should seek to drive out the uncertainty and risk in the first third of the schedule.

Our recommendation is that tasks, dependencies, and estimates be recorded in a spreadsheet, with which to compute the additional statistics (i.e. variance and mean), and, from there, loaded into tools, like Microsoft Project, to generate Gantt charts and critical path network diagrams. The probability of success by a given date can also be computed, using the PERT algorithm. In the future, we plan to give the students spreadsheets with pre-loaded columns and formulas, to make such tasks easier. We are also developing checklists to aid in the discussions.

In industry practice, joint planning sessions are often facilitated by staff from the project office. In our case we included students from the project management program of our business school. The combination made it real and proved both popular and highly beneficial for both engineering and business school students. The engineering students were able to share work with the business students, who had more experience with the project management tools and a serious interest in project management, while, for the business students, it gave them real project insight to go with what they had learned in class.

10. Conclusion

In this paper we described the most commonly reported risks in student capstone projects. Most of the reported risk concerned the schedule. We then compared these risks to the discussion of risk management from the major references used to teach students in capstone courses. We describe alternative risk management practices, specific to schedule risk, found in PERT, Spiral, Agile, Knowledge Factory, and FMEA practices. We found that the practice of PERT time estimation offered a comparatively easy way to apply a metric to the schedule risk in each task, while the Knowledge Factory showed how to apply manufacturing practices, like those from Lean and FMEA, to development projects akin to student capstones. Finally, we combined these ideas to show what a suitable, and industry relevant, schedule planning and risk management activity for student capstone projects might look like.

This paper is intended to share our many insights with others involved with student capstone projects, from both academia and industry. We hope it stimulates more discussion.

Bibliography

[1] J. Vanhanen and T.O.A. Lehtinen. “Software engineering problems encountered by capstone project teams.” *International Journal of Engineering Education*, Vol. 30, pp. 1461-1475, 2014.

[2] T. Ahtee and T. Poranen, “Risks in students' software projects,” In Proc. 2009 22nd Conference on Software Engineering Education and Training, 2009, pp. 154-157, doi: 10.1109/CSEET.2009.31.

[3] S. Koolmanojwong and B. Boehm, “A look at software engineering risks in a team project course,” In Proc. 26th International Conference on Software Engineering Education and Training (CSEE&T), 2013, pp. 21-30, doi: 10.1109/CSEET.2013.6595233.

[4] P. Mäkiaho and T. Poranen. “Risks management in software development capstone projects.” In Proc. 19th ACM International Conference on Computer Systems and Technologies, 2018, pp. 160–164, doi: 10.1145/3274005.3274024

[5] I. Sommerville. *Software Engineering*, 7th Edition. Boston, MA: Pearson Addison Wesley. 2004

[6] Project Management Institute. *A Guide to the Project Management Body of Knowledge,* 4th edition. Newtown Square, PA: Project Management Institute. 2008

[7] R. K. Wysocki. *Effective Project Management: Traditional, Agile, Extreme*, 5th edition. Indianapolis, IN: Wiley Publishing, 2009.

[8] H. Hoffman, *The Engineering Capstone Course: Fundamentals for Students and* Instructors. Springer Link, 2014, doi: 10.1007/978-3-319-05897-9.

[9] D. Hindle, *Prince2 Study Guide*, 1st Edition, Alameda, CA: Sybex, 2012

[10] C. Haskins and K. Forsberg, *Systems Engineering Handbook: A guide for system life cycle processes and activities*, Version 3.2.2. San Diego, CA, INCOSE, 2011.

[11] C. Alberts and A. Dorofee. "Risk Management Framework," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Report CMU/SEI-2010-TR-017, 2010.

[12] B. Boehm and D. Port, "Educating software engineering students to manage risk," In Proc. 23rd International Conference on Software Engineering (ICSE), 2001, pp. 591-600, doi: 10.1109/ICSE.2001.919133.

[13] B. Boehm, J. A. Lane, S. Koolmanojwong, and R. Turner. *The Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software*. Boston, MA: Addison-Wesley Professional, 2014

[14] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change*. Boston, MA: Addison Wesley Professional, 2004 [22]((5)) B. W. Boehm, "A spiral model of software development and enhancement," in Computer, vol. 21, no. 5, pp. 61-72, May 1988, doi: 10.1109/2.59.

[15] D. Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, Upper Saddle River, NJ: Addison-Wesley, 2011

[16] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*, 1st edition. Upper Saddle River, NJ: Prentice Hall, 2001.

[17] J. Stapleton and DSDM Consortium. *DSDM: Business Focused Development*, 2nd Edition, Pearson Education, 2003

[18] Society of Automotive Engineering, "Potential Failure Mode and Effects Analysis (FMEA) Including Design FMEA, Supplemental FMEA-MSR, and Process FMEA," SAE J1739_202101, 2021.

[19] J. E. Kelley, M. R. Walker, and J. S. Sayer. "The origins of CPM: a personal history." *PM Network*, Vol. 3, No. 2, pp. 7–22, 1989.

[20] United States. *Program Evaluation Research Task (PERT) Summary Report. Phase 1.* Washington DC: Special Projects Office, Bureau of Naval Weapons, Dept. of the Navy, 1958.

[21] D. K. Sobek, A. C. Ward, and J. K. Liker. “Toyota’s principles of set based concurrent engineering.” *Sloan Management Review*. Vol. 40, No. 2, pp. 67–83. Winter 1999

[22] B. W. Boehm, "A spiral model of software development and enhancement," in Computer, vol. 21, no. 5, pp. 61-72, May 1988, doi: 10.1109/2.59.

[23] H. Al Hashimi, A. Altaleb, and A. Gravell. “An empirical investigation of spikes in Agile software development.” In Proc. ACM European Symposium on Software Engineering (ESSE), 2020, pp. 37–43, doi: 10.1145/3393822.3432342

[24] M. Van Hilst, S. Huang, and H. Lindsay. “Process Analysis of a Waterfall Project Using Repository Data,” *International Journal of Computers and Applications*, Vol. 33, No. 1, pp. 49-56, doi: 10.2316/Journal.202.2011.1.202-2986

[25] M.T. Pich, C.H. Loch, and A. De Meyer, “On uncertainty, ambiguity, and complexity in project management.” *Management Science*, Vol. 48, No. 8, pp. 1008-1023, August 2002

[26] M. Patterson. *Accelerating Innovation: Improving the Process of Product Development*. New York, NY: Van Nostrand Reinhold 1993