Towards Automated Software Project Planning

Extending Palladio for the Simulation of Software Processes

Oliver Hummel
Software Design and Quality
Karlsruhe Institute of Technology (KIT)
oliver.hummel@kit.edu

Robert Heinrich
Software Design and Quality
Karlsruhe Institute of Technology (KIT)
robert.heinrich@kit.edu

Abstract: In every non-trivial software project a significant amount of effort must be devoted to project planning and management. However, project managers steering such projects are often still doing this based on relatively vague heuristics and/or their experience. Various studies and numerous project outcomes have continuously been demonstrating that this is not good enough as it frequently leads to budget and schedule overruns or even completely failed projects. In other engineering disciplines it is common, to simulate complex processes in order to mitigate such risks. Since the Palladio approach already provides functionality for architecture simulations and has been successfully extended for the simulation of business processes, it seems feasible to derive, simulate, and even optimize a project plan based on the Palladio model of an architecture. In this position paper, we sketch how Palladio could be used in order to become a useful tool for project managers that may help to avoid common planning pitfalls. Furthermore, we present initial ideas how its existing modeling elements can be mapped on the software development process and which extensions are still necessary.

1 Introduction

Budget and schedule overruns or even completely failed projects in software development are still rather common than an exception. During the so-called "software crisis" in the late 1960s, this was mainly attributed to poor tool support and a lack of knowledge in developing complex software systems. Although, these challenges have clearly been mitigated over the last decades as today sophisticated development environments and other tools have improved software development significantly, the ratio of challenged or failed software projects has not changed much. According to the current Standish Group chaos report [Sta12], only 16 percent of all software projects are successful, roughly 50 percent are "challenged", i.e. either delayed, more expensive, or both and around 30 percent are even prematurely aborted completely. Consequently, renowned researchers such as Robert Glass have proclaimed that today a "software estimation and planning crisis" [Gla06] endangers software projects far more than the previously mentioned lack of technical development skills. Under such circumstances, it is clearly surprising that many project managers still plan software projects with the same primitive techniques that have already been used some thirty years ago. Bar charts or similarly simple notations continue to be the predominant way for scheduling large-scale software projects

Proc. Kieker/Palladio Days 2013, Nov. 27–29, Karlsruhe, Germany
Available online: http://ceur-ws.org/Vol-1083/

(cf. e.g. [Som10]) potentially requiring dozens of person years. Their project plans are often not even based on the simplest armamentarium, such as work breakdown structures or rudimentary effort estimations for the tasks contained, but rather driven by business goals. Although buffers are regularly used to mitigate the risk of failure, usually no what-if analyses are used to simulate and better understand the impact of unforeseen events (such as unexpected dependencies, staff turnover or similar challenges [You03]) or just a sub-optimal progression of projects. Even worse from an engineering perspective is the fact that the current trend towards agile software development [Sch09] has widely led to an ignorance towards upfront project planning and purely relies on effort estimates created "on sight" during the course of the project. Since agile approaches prioritize the requirements of their customers they are usually able to deliver usable and useful software, quickly, though. However, this seems to have created a trend towards developing software systems until the budget is exhausted and not until all requirements are implemented (so-called design to budget). From an engineering standpoint, this is obviously not satisfactory, especially for larger projects, as it introduces a severe risk of an unsatisfying outcome into the development process.

Taking a look over the rim of the software engineering tea cup reveals that other disciplines with similar challenges have been using sophisticated simulation tools for numerous years. They plan, simulate, and optimize not only their products such as airplanes or cars, but also their business and production processes before they implement them in the real world. In software engineering, however, simulation techniques have been used only recently to predict the quality of software products, e.g. with the Palladio approach [BKR09]. Although the idea of simulating (or at least analysing) the software development process itself is also not completely new, existing approaches and tools, mainly presented in the 1990s [KMR99], are rather high-level and treat the development process as black-box. In other words, it is not possible to model the structure of the project team and dependencies between architectural elements, for instance. Moreover, the relationship between software development effort and technical characteristics of the software product, such as its architecture, is usually not considered from a project management perspective.

Business process simulation approaches on the other hand typically allow for the modelling of workflows consisting of activities and dependencies between them. Some of them also allow representing "human resources" involved in the processes while characteristics related to the software product, such as the architecture, can only be represented in software product simulation approaches such as Palladio. The relation between development process and product, however, is completely uncovered in simulations today. We believe that the Palladio tool suite and recent extensions [HHP12] that successfully integrated the simulation of business processes open up a new realm of possibilities: it will allow fine-grained simulations of complex software development projects (see section 2) in the near future. This would not only enable project managers, technical leaders, and other decision makers to carry out what-if analyses of software projects (cf. section 3) before they kick them off, it would also be a valuable tool in software engineering education as it can help to illustrate the difficult trade-offs involved in such decisions. The main contribution of this position paper is presenting this novel application idea for Palladio in section 4 and 4.1. Moreover, a first juxtaposition of modelling elements currently available in Palladio and elements required for development process simulations as well as suggestions for Palladio extensions derived from this comparison (in 4.2 and 4.3) are also presented.

2 Foundations

In this section we prepare the stage for presenting our vision by explaining various foundations of software project planning, software simulation, and activity prediction approaches in three subsections.

2.1 Project Planning Foundations

Software development processes have often been illustrated by using the (over-) simplifying waterfall model [Som10] that basically comprises five consecutive development phases, namely *requirements elicitation*, *architecture and software design*, *programming*, *testing*, and *deployment*. In this model, a new phase can only begin once its predecessor has been completed, which in theory allows a relatively simple upfront project planning based on a reasonable understanding of the project requirements. Under these prerequisites, project managers “merely” need to take three fundamental elements into account for planning: they start by identifying the **artefacts** that must be created to build the system. Once these are identified, the **tasks** required to create them can be derived, estimated in terms of required effort and assigned to developer **roles** respectively the people that fill them in a project.

Practical experience has shown, however, that upfront requirements understanding is seldom complete or unambiguous so that it is impossible to foresee all eventualities that may occur in a large project perhaps running for a couple of years. Even worse, on average at least between one and three percent of all requirements are estimated to change each month of the project duration [Jon08]. Therefore, simply put, more recent agile development approaches (such as Scrum [Sch10]) propose to analyse and implement prioritized requirements one after the other in order to mitigate the risks of the waterfall-like “big-bang approach” just described. Nevertheless, effort estimation [Boe81] and schedule creation [Som10] remain a challenging topic in its own right for both, waterfall as well as iterative projects.

In his seminal book “The Mythical Man-Month” [Bro75], Frederik Brooks has described other important foundations for fine-grained simulations of software development processes that shall be briefly discussed in the following. He recognized that development tasks can typically be instances of three different archetypes: they can either be perfectly partitionable, such as picking cotton; not partitionable at all, such as bearing a child; and partitionable to a certain degree of parallelisation. Once the threshold is reached, the latter archetype tends to take more time the more people are assigned to complete it. This is known as a diseconomy of scale and has also been the basis for Brooks’ famous law: *Adding manpower to a late software project makes it later* [Bro75]. However, as simple as this “law” appears at a first glance, as often it is ignored in practice even by experienced project managers. Software engineering has been relatively ignorant against analysis and simulation of the software development process so far. Apart from a number of relatively high-level approaches developed in the 1990s, described in more detail in the section on related work, there has not been much momentum in this direction.

2.2 Software System and Business Process Simulation Foundations

In general, simulation is a method to evaluate a model numerically, through executing it for a set of inputs in order to see how the output measures evolve [Law06]. It has already been applied in numerous domains to analyse different kinds of models including software architectures, business processes models and even simple software process models. Palladio is a software architecture simulation approach that originally addressed the simulation and analysis of architectural quality characteristics such as performance and reliability. The Palladio Component Model (PCM) [BKR09] allows for the description of component-based software architectures and their hardware environments which form the central input of the simulations. Recently, the original PCM has been extended with concepts for business process modelling and simulation by Heinrich et al. [HHP12] in order to allow the analysis of business processes that are heavily depending on IT systems. Among other concepts, the PCM has therefore been augmented to simulate actions completely performed by human actors.

Moreover, the PCM has been extended with the possibility to represent the organizational environment of a business process so that it can also represent human actors and their properties, such as availability and organizational role. Likewise, the underlying simulation engine EventSim [MeH11] has been extended by a novel scheduling policy reflecting the behaviour of human actors in a simulation. Based on queuing networks [LZG84], the extended simulator is able to predict the utilization of human actors and the execution times of the actions performed by them. This allows the identification of potential bottlenecks and other design flaws in IT-supported business processes.

2.3 KAMP: Karlsruhe Architectural Maintainability Prediction

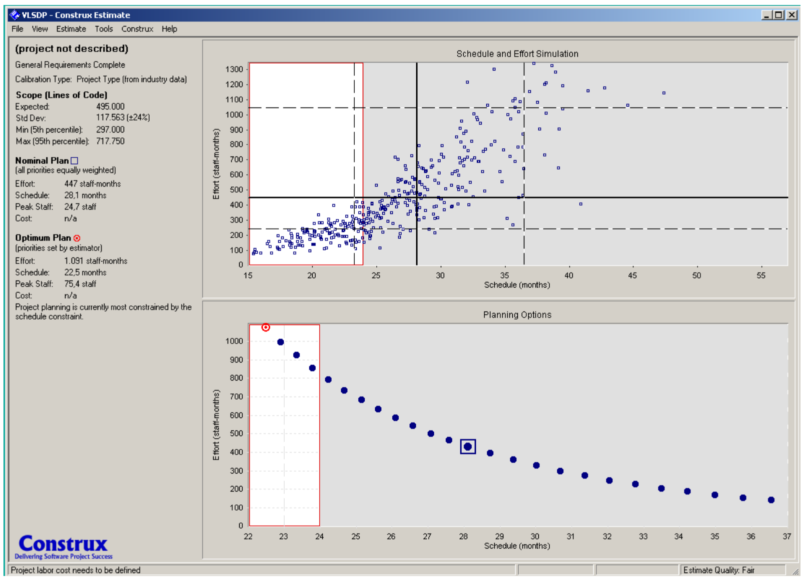
KAMP [StR09] is a quantitative architecture-based prediction method for estimating the effort of changes to a given Palladio model of a software architecture. KAMP for the first time uses concrete architectural models to evaluate software maintainability for potential change requests. Based on these findings, the method estimates change efforts for a semi-automatic derivation of work plans and bottom-up effort estimation. The overall effort of change requests is determined by also taking re-implementation activities as well as re-deployment and upgrade activities into account. Thus, in a nutshell, KAMP supports the identification of a list of activities necessary for software architecture change and is another important building block for a fine-grained simulation of software processes as well as for an automatic derivation of project plans.

3 Related Work and Open Challenges

In this section we give a brief overview of related work that has been targeting the simulation of software processes in recent years and summarize their shortcomings for using them in the fine-granular simulation of software processes. Based on these shortcomings we collect the main open challenges we intend to address with our envisaged Palladio extension presented in section 4.

3.1 Related Work

Mainly influenced by Boehm's seminal work on software estimation, there has been a constant interest in developing tools that support software estimation. Barry Boehm and his group have also packaged the knowledge collected in their COCOMO 81 and COCOMO II methods into easy to use software tools that are freely available on the Web[1]. The former tool is "merely" a simple calculator that derives the estimated work force needed to implement a system of a given size (in lines of code) in a given environment specified by so-called cost drivers. In addition, the latter tool offers some rudimentary simulation capabilities in order to derive a probability distribution for likely personnel efforts. Numerous other tools based on or inspired by Boehm's works are available today. For example, Steve McConnell's company Construx offers a tool called Estimate (see Figure 1) that is able to execute simulations in order to find the most likely outcomes for a given project configuration [Hum11].


VLSDP - Construx Estimate

(project not described)

General Requirements Complete

Calibration Type: Project Type (from industry data)

Scope (Lines of Code)

Expected:	495.000
Std Dev:	117.563 (±24%)
Min (5th percentile):	297.000
Max (95th percentile):	717.750

Nominal Plan □
(all priorities equally weighted)

Effort:	447 staff-months
Schedule:	28,1 months
Peak Staff:	24,7 staff
Cost:	n/a

Optimum Plan ⊙
(priorities set by estimator)

Effort:	1.091 staff-months
Schedule:	22,5 months
Peak Staff:	75,4 staff
Cost:	n/a

Project planning is currently most constrained by the schedule constraint.

Project labor cost needs to be defined

Estimate Quality: Fair

Figure 1: Screenshot of simulated project outcomes in Construx Estimate.

[1] COMOMO 81 calculator: http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/cocomo81.html
COCOMO II calculator: http://csse.usc.edu/tools/COCOMOII.php (both accessed in September 2013)

In other words, the various tools similar to Estimate such as the ones developed by Northrop Grumman [RVM99] or by Abdel Hamid [AbH89] merely aim on creating probability distributions for the most important effort indicators or error distributions of software projects through so-called monte-carlo simulations. In order to achieve this, simulations are executed numerous times with different random values in order to get a better understanding on how they influence the final project outcome. However, beyond some basic cost drivers such as personnel capabilities or performance requirements, it is neither possible to influence the internal workflow nor the dependencies of requirements and architectural components within the simulated software process.

In addition to project planning, the publication of Kellner et al. [KMR99], which is also a good starting point for more related work, identified five further motivations for the use of software process simulation approaches:

- Project planning
- Strategic management
- Process control and operational management
- Process improvement and technology adoption
- Process understanding
- Training and learning

There is also a group of educational software process simulators that can be used to train future project managers. Probably the best known example in this category is SESAM that was developed by the group of Ludewig at the University of Stuttgart [DrL00]. On the one hand, SESAM is a very powerful tool that allows a lot of fine-grained management decisions during the simulated development project. On the other hand, however, it is not usable for extended what-if analyses since it is designed as a game requiring continuous interaction and decision making of the player as well as several hours of playing time. Moreover, there is also a variety of “classic” process simulation tools available in business administration and other domains such as ADONIS [HJK97] etc. that also seem well suited for software process simulations at a first glance. However, they currently neither support software process specific roles or artefacts, nor do they have obvious connection points to architectural models so that a significant amount of effort would be necessary to tailor them for the simulation of software processes.

3.2 Open Challenges

In summary, the approaches presented so far have only limited expressiveness when it comes to modelling software development processes. For example, activities are typically limited to the coarse-grained phases found in the waterfall model and the only artefact that is modelled is the final system. “Access" to intermediate documents is typically not possible; neither there is a way to influence the ordering of necessary development activities or potential dependencies between them. Moreover, it is also not possible to assign one of Brooks’s three task partitioning archetypes to the activities so that this important aspect remains completely unsupported so far.

The behaviour or experience of human actors in various activities is typically also not reflected by simulation tools. This implies that it is for example not possible to compare the effects of implementing a system based on a sequential flow of activities, as mandatory when a sequential process model is used, with an iterative one. Moreover, since existing systems do not target such a fine-granular analysis of the software process, it is also not clear, how a user-friendly interface to such systems could look like.

4 Vision

We believe that a sophisticated simulation tool that offers capabilities to model software development processes on a detailed level with activities, artefacts and people involved could lead to a significant improvement in project planning quality. Since software processes are typically complex, we advocate that as many elements as possible should be automatically arranged, ideally based on a model of the requirements. We envisage a simulation tool that should be able to derive a probability distribution for the effort of the planned project similar to existing tools. Moreover, it should also be able to generate a work breakdown structure of the activities needed to implement these requirements based on the findings of KAMP. And finally, a project plan that brings these activities into a meaningful order and assigns the available developers respectively other stakeholders to them should also be supported. In order to simplify the arrangement of artefacts and activities during the process, we envisage the creation of a set of predefined process templates that enable the simulation tool to automatically schedule activities according to a selected process model, such as Scrum [Sch09] etc. Other important parameters include an estimate of the complexity of the system's requirements as well as of the capabilities of the developers implementing it, as it is common practice in proven effort estimation approaches (such as COCOMO [Boe81]) and the high-level simulation tools based on them.

4.1 Medium-Term Approach

Admittedly, the previous vision and especially the derivation of a project plan from a requirements model will be very challenging since those models are mostly created in textual form today. However, we believe, that with the considerable foundations available within the extended PCM and KAMP, a solution based on architectural models is within reach. Hence, we are confident that a wide automation of the following process should be feasible.

Starting with a Palladio model of the desired system's architecture, it comprises the following phases:

Phase 1: Apply KAMP, respectively an extended version of it, to derive a list of necessary activities required to implement the designed architecture. Or in other words, create a work breakdown structure for the implementation.

Phase 2: Automatically derive a project plan and a Palladio process model (cf. PCM process extension) of it by arranging the activities derived from phase 1 according to the desired development process. Several processes can be compared by representing each of them as a process model and conducting a simulation study for each paradigm.

Phase 3: Conduct a Palladio process simulation for each process model: The simulation predicts various performance aspects such as the execution times of the process and the activities within the process or the utilization of each human actor involved in it. In addition, it may also evaluate temporal constraints. For each simulation study, the results are compared in order to find a process model that fits best into the given context. Ideally, the generated models are optimized automatically, e.g. by applying search-based optimization techniques. Moreover, what-if analyses may be conducted by predicting the impact of changes to the models. For example, it may be interesting how allocating additional human actors or rearranging activities changes the overall process performance.

4.2 Mapping SE Process Elements to Palladio Business Process Elements

In order to underline the short-term feasibility of the vision presented in the last subsection, let us present a juxtaposition of modelling elements already available in the extended PCM and the central elements required to model software development processes.

Table 1: Comparison of Software Process Elements and Business Process Elements

Software Process Element	**Appropriate Palladio BP Extension Modelling Element**
Artifact/Document	n/a
Human Resources	Human actor
Role	Organizational role (of a human actor)
Developer Task (Action/Activity)	Actor step, activity
Task Type	n/a
Communication Dependency	Process control flow
Clustering of activities into phases	Hierarchical composition of steps/activities
Team	n/a
Execution time for an activity	Processing time of an actor step
Deadline	n/a

As illustrated by the table 1, Palladio already offers various modelling elements that can be used with relative ease for supporting the modelling of software development processes. However, as is also visible, there are some elements that are still missing to date, namely – artefacts, task types, teams and deadlines. An additional layer for grouping developers and other stakeholders into teams is also required. This can probably be created easily through extending the existing organization environment model (cf. [HHP12]) with a corresponding structure.

The business process model can be extended by model elements to represent artefacts and their dependencies, i.e. the necessary communications and artefact flows, between potentially parallel activities that may slow down the process. Finally the three archetypes of different activity parallelisation characteristics as identified by Brooks must also be integrated for a proper modelling of software processes. The existing simulation behaviour can be easily extended or adapted by specifying new so-called traversal strategies (e.g. for a new model element) and registering it in the simulator framework. Whenever, the simulation encounters a model element (e.g. an activity) in the process model, the simulation behaviour specified in the corresponding traversal strategy is executed. For example, the activity may be assigned to an actor who is suitable to perform the activity. Therefore, the behaviour related to the aforementioned model elements, such as different activity types or dependencies between artefacts and activities can be reflected easily by adapting existing traversal strategies and creating new strategies where necessary. Similarly, the behaviour specification of human actors can be adapted. A human actor can be seen as a special kind of processing resource (cf. [HHP12]). Thus, merely the corresponding scheduling policies need to be adapted. Deadlines correspond to constraints in process execution, for example, the deadline of a certain activity may be 5 time units after its processing has been started. This can be evaluated for each simulation run in retrospective by comparing simulation results, i.e. the execution time of the activity in this case, with the constraint, i.e. the deadline. Such verification is already possible using the existing tooling, however, currently, the comparison has to be conducted manually. In the future, a sensor tailored for the identification of deadline violations can be implemented based on the existing sensor framework.

4.3 Practical Usability

With the enhancements just described, the medium-term vision should be implementable relatively quickly. However, in order to provide a practical and usable tool, some additional preparation works need to be executed. First, usability is certainly an important issue that needs to be dealt with, since simulation models will be complex and must be usable for non IT experts. Thus, we envisage that templates for various process models are created based on the activities identified by KAMP so that users do not need to model every project completely from scratch. Second, it will also be important, to calibrate with realistic project data that can be obtained from seminal work in the area of software cost estimation. Finally, prediction quality and usability must be evaluated, ideally with the help of well documented finished projects that can be “reverse engineered” in order to obtain simulation models that can be compared with the actual project outcomes.

5 Conclusion

In this position paper we have presented a vision for extending Palladio with additional elements that would not only allow the simulation of software and business process performance, but also the performance of a development process that aims on creating an actual system from a given architectural model. We have demonstrated that with relatively small extensions to the Palladio meta-model creating a unique software development simulation tool should be possible: It would allow deriving and analysing fine-grained simulation models for development processes based on architectural models largely automatically. Since existing tools treat the software development process as a black box where individual activities cannot be influenced by, and have no connection to existing architectural models, we believe that this proposal is a significant step forward towards a better support and control environment for project managers that goes far beyond the simple Gantt chart based tools available today.

References

[AbH89] T. Abdel-Hamid. The dynamics of software project staffing: a system dynamics based simulation approach. IEEE Transactions on Software Engineering, *Vol.* 15, Iss. 2, 1989.

[BKR09] S. Becker, H. Koziolek, and R. Reussner. „The Palladio component model for model-driven performance prediction”, Journal of Systems and Software, Vol. 82, 2009.

[Boe81] B. Boehm. Software Engineering Economics, Prentice Hall, 1981.

[Bro75] F. Brooks. The mythical man-month. Addison-Wesley, 1975.

[DrL00] A. Drappa, J. Ludewig. Simulation in software engineering training. Proceedings of the 22nd International Conference on Software Engineering, 2000.

[Gla06] R. Glass. The Software Estimation Crisis, in Ebert, & Dumke: Software Measurement, Springer, 2006.

[HHP12] R. Heinrich, J. Henss, and B. Paech. “Extending Palladio by business process simulation concepts”. In S. Becker, J. Happe, A. Koziolek, and R. Reussner, editors, Palladio Days 2012 Proceedings, pages 19-27, Karlsruhe, 2012.

[HJK97] J. Herbst, S. Junginger, H. Kühn: Simulation in Financial Services with the Business Process Management System ADONIS, 9th European Simulation Symposium, Society for Computer Simulation, pp. 491-495, 1997.

[Hum11] O. Hummel: Aufwandsschätzungen in der Software- und Systementwicklung (in German), Spektrum Akademischer Verlag, 2011.

[Jon08] C. Jones: Applied Software Measurement, McGraw-Hill, 2008.

[KMR99] M. Kellner, R. Madachy, D. Raffo. “Software Process Modeling: Why? What? How?”, Journal of Systems and Software, Vol. 46, No. 2, 1999.

[Lar04] C. Larman. Applying UML and Patterns (3rd ed.), Prentice Hall, 2004.

[Law06] A.M. Law: Simulation Modeling and Analysis, Mcgraw-Hill, 2006.

[LZG84] D. Lazowska, J. Zahorjan, G. Graham, K. Sevcik. Quantitative System Performance – Computer Systems Analysis Using Queuing Network Models, Prentice-Hall, 1984.

[MeH11] P. Merkle and J. Henss. “EventSim – an event-driven Palladio software architecture simulator,” in Palladio Days Proceedings, Becker, Happe, Reussner, Eds., 2011.

[RVP99] D. Raffo, J. Vandeville, R. Martin: Software Process Simulation to Achieve Higher CMM Levels, Journal of Systems and Software, Vol. 46, No. 2, 1999.

[Sch09] K. Schwaber: Agile Project Management with Scrum, Microsoft Press, 2009.

[Som10] I. Sommerville: Software Engineering (9th ed.), Addison-Wesley, 2010.

[Sta12] Standish Group, Inc. Chaos Report 2012.

[StR09] J. Stammel, R. Reussner. KAMP: Karlsruhe Architectural Maintainability Prediction. Proc. of the 1. Workshop GI-Arbeitskreises Langlebige Softwaresysteme, 2009.

[You03] E. Yourdon: Death March (2nd ed.), Yourdon Press, 2003.