Towards Efficient Risk-identification in Risk-Driven Development Processes

Andreas Demuth
Institute for Software Systems Engineering
Johannes Kepler University
Linz, Austria
andreas.demuth@jku.at

Markus Riedl-Ehrenleitner
Institute for Software Systems Engineering
Johannes Kepler University
Linz, Austria
markus.riedl-ehrenleitner@jku.at

Roland Kretschmer
Institute for Software Systems Engineering
Johannes Kepler University
Linz, Austria
roland.kretschmer@jku.at

Alexander Egyed
Institute for Software Systems Engineering
Johannes Kepler University
Linz, Austria
alexander.egyed@jku.at

ABSTRACT

Today's software projects face an environment of continuous change and evolution. In order to handle evolution in development environments (e.g., requirements, technology) effectively, over the last decades well-established development processes have been adapted significantly and new process models have been proposed. For example, there is a wide range of agile processes which are risk-driven and which not only handle but embrace frequent change. However, agile and risk-driven processes still suffer from a lack of support for automatic and generic identification of certain risks.

In this paper, we present an approach for automatic, fine-grained identification of high-risk development artifacts and high-risk-areas within those artifacts that is easy to integrate with common development processes.

The approach has been implemented in a proof-of-concept prototype. First validation results indicate that the approach is highly scalable and provides continuous feedback about a development project's technical risks.

Keywords

Process support, project management, traceability, consistency checking, change propagation

1. INTRODUCTION

In today's software projects, project managers have to face several challenges simultaneously. First, the complexity of software systems increases steadily [1]. While in the past it was common to develop standalone software systems, today's development projects typically deal with systems that are integrated with existing infrastructures and that interact with numerous other systems. Thus, development projects nowadays require sophisticated management and documentation techniques in order to handle this high level of complexity. Second, the frequency in which new products or technologies hit their target markets is also increasing [1]. Producers of IT-systems or services update products at an impressive pace and drive technological progress forward by frequently revealing new technologies. This means that during the development of a new system, the system's expected environment of operation is changing (e.g., used technologies change, or the interfaces of external systems evolve). For example, new technologies may become available that provide faster or more reliable information the system under development needs for operation. In this case, the system under development must be adapted quickly to the new third-party technology. Moreover, frequent evolution and revolution of technology has also changed customers' expectations. While it was common practice in the past to define system requirements once at the start of a project, customers nowadays change opinions more frequently and expect that significant changes to the system under development's specification can still be made later on [1].

Flexible, agile, and risk-driven processes such as the Spiral Model [2], Extreme Programming [3], or Scrum [4] have been proposed to address frequent evolution and reduce project risks [5]. Indeed, these processes suggest tasks that focus on determining, evaluating, and minimizing project risks frequently and keeping a close relation to stakeholders in order to get informed about any change request as soon as possible.

Typically, there is a distinction between organizational risks and *technical risks* [6]. While for organizational risks there are typically mitigation strategies available that can be applied by the project management (e.g., enforce docu-

ICSSP'16, May 14-15, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2904354.2904364

mentation to avoid loss of knowledge if an employee quits unexpectedly), the identification and evaluation of technical risks is quite difficult [6, 7]. Even though risk-driven processes suggest tasks and best-practices that allow project managers to quickly discover arising needs for changes (e.g., through regular stakeholder meetings or continuous evaluation of alternatives) [5], these processes typically do not provide means for automatic determination of *high-risk-areas* in development artifacts (i.e., areas in development artifacts whose change may require a significant amount of follow-up changes to be performed) [7]. An example for such a high-risk-area in a development project would be a single method in source code that is called from various places across the system to implement numerous use cases. If this method was changed, it would potentially cause a large number of use-cases to be no longer implemented correctly and thus require many adaptations in the source code. Moreover, it is typically necessary to determine manually which development artifacts, and in particular which areas within those artifacts (e.g., which requirements or which pieces of source code), impose the highest risks for a project. Unfortunately, this is not only time consuming but also error prone, as it requires extensive knowledge of not only a single but all of a project's development artifacts [8]. Thus, it is quite likely that in today's highly complex projects, areas of high risk remain undetected.

In this paper, we present an approach that addresses these issues by providing fully automatic determination of high-risk-areas in projects' development artifacts. The approach relies on the *DesignSpace* [9] as a platform for development artifact and process integration. Specifically, incrementally built trace information within and between development artifacts is used to determine areas in individual development artifacts that impose significant project risks. In contrast to other approaches, the fine-grained integration of development artifacts in the DesignSpace allows for relations between different development artifacts to be also considered for finding high-risk-areas.

The approach has been implemented in a software prototype. Empirical validation results indicate that the proposed approach scales and provides continuous feedback about high-risk-areas in development artifacts.

2. EXAMPLE AND MOTIVATION

To illustrate the need for fine-grained development artifact integration and automatic detection of high-risk-areas, we introduce a small example from the mechatronics domain. In this example, a software-controlled robot arm is developed that can move objects from one point to another. A CAD drawing of the robot arm as well as an illustration showing its movement angles and important variables for calculations is depicted in Fig. 1(a). The project's development artifacts are depicted in Fig. 1(b). Specifically, these are (from left to right): spreadsheets for calculations, electric models for simulations, and source code for controlling the robot arm. These development artifacts depend on each other. For example, the control software uses information provided in the calculation spreadsheet and in the electric model. The electric model uses information from the calculation spreadsheet and also from the controller software. Finally, the calculation spreadsheets uses some information from the electric model. We assume that the development project follows an incremental process (e.g., the

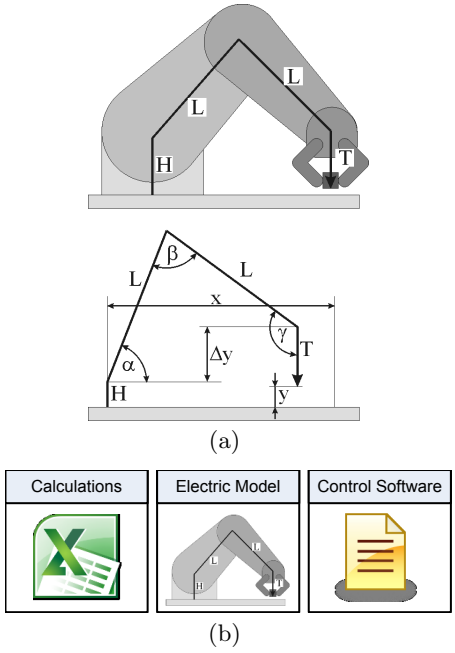

Figure 1: Illustration of robot arm (a) and its development artifacts (b).

Spiral Model [2]) in which a new process iteration is about to start. At this point, risks should be identified, evaluated, and the highest risks should then be addressed during the iteration.

Unfortunately, there is little to no guidance on how to find and evaluate technical risks in development projects in an objective manner, and thus processes typically rely on estimations by experts [10]. To obtain an overview of potential technical risks, representatives of the teams responsible for the different artifacts are asked to identify the technical risks associated with their respective development artifacts. For that purpose, these representatives also have to identify high-risk-areas within the development artifacts (i.e., those parts of the artifacts that are the most crucial). Then, all stakeholders of the project have to rank those risks and they have to find an agreement about which risks to address in the next process iteration. Indeed, this task is quite complex as it requires a deep understanding of how the individual development artifacts are related and how important each individual development artifact is for overall project success [8, 10]. While representatives of the individual teams may or may not have a solid understanding of the high-risk-areas within their respective development artifact (e.g., which part of the source code is most critical), they typically cannot properly estimate the importance of the development artifact for the project due to a lack of knowledge of other domains. Thus, the ranking of risks is often done in an ad-hoc manner and it is typically based on vague estimations and guesses made by project managers. Indeed, there are metrics defined for certain types of development artifacts that help stakeholders to estimate their complexity (and thus their general technical risk), such as lines of code or feature counts. Unfortunately, such metrics do exist only for certain domains and certain types of development artifacts (e.g., software artifacts [6]). There is clearly a lack of

guidance that is based on objective measures and that can be provided for arbitrary development artifacts.

In our example, the next process iteration may focus on the robot's control software because the software team's representative believes that there are still some major challenges to be tackled, whereas those responsible for the calculations and electric model artifacts are confident that their respective development artifacts do not impose serious project risks. In the next sections, we will present a novel approach that provides objective guidance for indentifying risks of arbitrary development artifacts.

3. DESIGNSPACE

Before we discuss in detail how our proposed approach detects high-risk development artifacts and high-risk-areas within individual development artifacts automatically and efficiently, and how it provides guidance for change propagation after system adaptations, we present in this section the core concepts of the *DesignSpace* [9], the development artifact integration and knowledge management platform the approach relies on.

The DesignSpace is a knowledge sharing and development artifact platform. Its key principle is that any knowledge that is used during a development project is stored centrally in a data cloud, using a uniform representation of knowledge. Indeed, stakeholders of a development project are used to certain tools for handling development artifact generation and editing. For instance, managers often use spreadsheets for calculations and word processing tools for documenting system requirements (e.g., Microsoft Office). Engineers, on the other hand, typically use sophisticated tools for performing complex modeling and simulation tasks (e.g., ProEngineer for CAD drawings); and software developers are commonly using integrated development environments (IDEs) to write source code (e.g., Eclipse IDE). Since stakeholders typically use tools that are mature and efficient for performing certain tasks, the DesignSpace does not replace these tools, but it allows all stakeholders to continue working with their favorite tools and it mirrors continuously the knowledge available in stakeholders' tools. This is done by translating development artifacts (e.g., spreadsheets or source code) to the DesignSpace's uniform representation of knowledge—at a fine level of granularity. For example, each individual cell of a spreadsheet is represented by a separate entry in the DesignSpace. Similarly, source code is represented as its corresponding abstract syntax tree; there is one entry in the DesignSpace per syntax tree node. The mirroring may be based on files or it may be done by tool adapters that directly synchronize development artifacts with the DesignSpace as they are edited by stakeholders in tools. Whichever method is used, the DesignSpace continuously synchronizes its mirrored information with development artifacts as they evolve.

By mirroring all development artifacts at a fine level of granularity and by using a uniform representation of knowledge, the DesignSpace allows for further services such as inter- and intra-artifact linking of knowledge and consistency checking. For our running example, Fig. 2 depicts how the development artifacts are represented uniformly in the DesignSpace. Note that for each individual development artifact (i.e., control software, calculations, and electric model) there are numerous elements due to the fine-grained integration approach the DesignSpace uses (for now, please ignore the color-coding of development artifacts and individual elements as well as the numbers assigned elements). For instance, there is a single element for each cell of the calculation spreadsheet. Similarly, the electric model is represented using multiple elements, each representing an atomic piece of knowledge from the electric model (e.g., individual variables). The same is true for the control software, where there is a single element for each node of its corresponding abstract syntax tree.

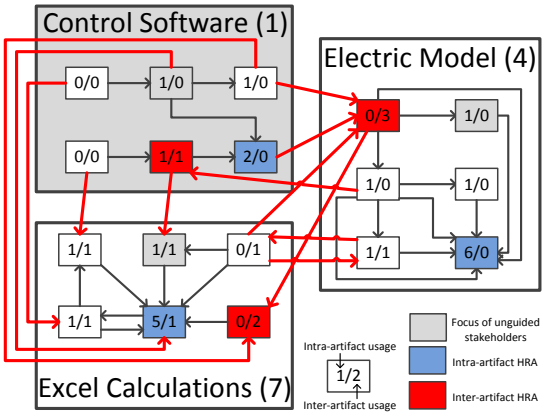

Figure 2: DesignSpace representation of robot arm development artifacts.

Moreover, there are connections between the elements of the individual artifacts (drawn as black arrows) as well as connections between elements of different artifacts (drawn as red arrows). A connection between two elements means that the source element is somehow relying on the target element (e.g., calling a method, or reusing a calculated value). Although a specific meaning may be attached to any connection in the DesignSpace, for our purposes the sheer existence of a connection is sufficient for detecting high-risk development artifacts and high-risk-areas, thus we omit a more detailed discussion here. In the DesignSpace, connections between elements are created incrementally as development artifacts (and their individual elements) are edited. During artifact editing, stakeholders are expected to document whenever they reuse knowledge from other artifacts by establishing an inter-artifact connection (i.e., red arrow in Fig.2) between the edited element and the element that represents the reused knowledge. Thus, there is no significant overhead during development editing and the documentation of knowledge reuse. Intra-artifact connection (i.e., black arrows in Fig. 2 are created automatically when individual development artifacts are mirrored to the DesignSpace.

In the next section, we will discuss how the DesignSpace's services are employed to perform efficient and automatic detection of high-risk development artifacts and high-risk-areas within individual artifacts.

4. AUTOMATIC RISK-IDENTIFICATION

With all development artifacts integrated in the DesignSpace and the knowledge being linked, those development artifacts that impose the highest risks in a development project can be identified automatically. Moreover, it is possible to identify within each development artifact where the highest risks are located, regardless of its kind (e.g., source code, CAD drawing, spreadsheet).

4.1 High-Risk Development Artifact Identification

Intuitively, in a development project, the risk that an individual artifact imposes on project success increases with the amount of knowledge that it contains and that is also used by other artifacts. Indeed, adapting an artifact on which numerous other artifacts rely may require a potentially large number of co-adaptations to be necessary. Thus, it is crucial that such artifacts reach a stable state as early as possible.

We developed a new service for the DesignSpace that automatically tracks the usage of individual artifacts and that continuously provides information about which artifacts are most crucial in the project. For each individual element within a given development artifact, our service continuously tracks how strongly it is used by (elements of) other artifacts. This is done by counting the number of the element's incoming inter-artifact connections. The number of inter-artifact connections for each artifact is depicted as the second number of the elements' labels in Fig. 2 (e.g., `0/3` means that the element has three incoming inter-artifact connections).

As discussed above, when using the DesignSpace, connections between elements of different development artifacts are created and managed incrementally whenever a development artifact is edited and knowledge from another development artifact is used. Therefore, the count of incoming inter-artifact connections can be updated efficiently during development. Whenever an inter-artifact connection is established, its target element's count of incoming inter-artifact connections is increased; whenever a connection is removed, its target element's count of incoming inter-artifact connections is decreased. These operations are atomic and do not impose a significant overhead.

The total risk of a development artifact is defined as its total usage by other development artifacts (i.e., the sum of incoming inter-artifact connections for all of its elements). In Fig. 2, each development artifact's total risk is written in parenthesis after its name (e.g., the Excel calculations artifact has a total risk of 7). Note that this total risk is calculated without requiring individual (or groups of) stakeholders to be aware of the role of a development artifact in the system under development. For each development artifact, those elements that are connected most often to other elements are also highlighted with red background in Fig. 2.

Our service continuously provides information about which development artifacts (and which of their elements in particular) are used most often by others. Stakeholders, especially project managers, are thus provided with objective information about each individual development artifact's importance for the project, without having to rely on experts' subjective opinions.

Coming back to our running example, in Section 2 we assumed that stakeholders identified the control software to be a high-risk artifact based on the subjective opinion of the software team's representative (as indicated by its grey background in Fig. 2). However, our service reveals that the control software is hardly used by other development artifacts but that both the Excel calculations and the electric model artifacts impose potentially higher risks. Therefore, the stakeholders may want to revise their decision and focus on stabilizing the calculations rather than optimizing the control software during the next iteration. Note, however, that our service provides objective information and thus guidance, but it does not prescribe any automation for further development—stakeholders are free to ignore the information.

4.2 High-Risk-Area Identification

While the identification of high-risk development artifacts is important to determine on which artifacts to focus during an iteration, usually all development artifacts are evolved continuously. Therefore, it is also necessary in risk-driven development processes to determine which parts within a given development artifact impose the highest risk, even for development artifacts that are of generally low risk.

The identification of such high-risk-areas within development artifacts is done similarly to the identification of high-risk development artifacts. Intuitively, the more often a given element in a development artifact is used by other elements (of that artifact), the more crucial it is that this element becomes stable and thus a reliable source of knowledge for reuse. Our service tracks for each element in the DesignSpace the number of incoming intra-artifact connections. This count is depicted as the first number of the elements' labels in Fig. 2 (e.g., `5/1` means that the element has five incoming intra-artifact connections). Note that these connections are managed automatically and incrementally by the DesignSpace when development artifacts are edited. Thus, the counter for each individual element can be updated efficiently whenever connections are created or deleted without imposing significant overheads. Also note that this detection of high-risk-areas supports any kind of development artifact. While there are approaches that detect critical areas for some kinds of artifacts (e.g., source code profilers detect method usage), for a wide range of artifacts there are no approaches available. For instance, our approach detects cells in Excel spreadsheets that are used frequently to calculate other cells' values.

For our running example, in Fig. 2 the most crucial element of each artifact is highlighted with blue background. Again, note that our approach provides objective information about how often the knowledge represented by individual elements is reused by others. Stakeholders, on the other hand, cannot provide reliable information about intra-artifact reuse of knowledge as in today's development projects artifacts are typically edited by multiple stakeholders and individuals are not aware of how knowledge is reused (e.g., one engineer defines a variable to be used in a diagram, and another engineer uses the variable in a simulation). Therefore, our approach provides valuable and objective guidance about which parts of a development artifact to focus on.

Indeed, the elements of highest risk are high-risk areas of atomic size (i.e., consisting of a single element). At the same time, they can be seen as the center element of a larger high-risk area that contains not only the high-risk element, but also those elements directly (or transitively) connected to it. Such larger high-risk areas can be computed easily using traditional search algorithms (e.g., [11, 12]).

4.3 Process Integration

Once high-risk elements and areas have been identified, this information can be used to further guide the development process. For upcoming process iterations, for instance, project managers want to focus the planned effort on high-risk areas in order to eliminate the associated risks as early in the process as possible. Note that there are numerous

ways of presenting such information and that visualization is beyond the scope of this work. However, our approach provides information detailed enough the visualize high-risk areas effectively.

5. PRELIMINARY VALIDATION

To validate our approach, we demonstrated its technical feasibility and performed first scalability tests.

Technical Feasibility: To demonstrate the technical feasibility of our approach, we have implemented it as a prototype service for the DesignSpace integration framework.[1] The service continuously tracks the number of incoming inter- and intra-artifact connections for individual elements and provides a sophisticated API for obtaining information about high-risk artifacts and high-risk-areas programmatically.

Scalability: To assess the scalability of the developed prototype service, we performed scalability tests in which various development artifacts were mirrored to the DesignSpace and connections between individual elements were changed. We observed that the processing time for updating the risk-information of individual elements remained constant for different sizes and kinds of development artifacts, regardless of whether inter- or intra-artifact connections were changed. Typically, risk-information was updated within milliseconds after a connection change was performed.

6. RELATED WORK

Risk-identification in development project is an active field of research. Next, we discuss some approaches that are close to ours. Antinyan et al. [6] proposed an approach for risk prediction in software development projects that uses software metrics to determine the risks of individual parts of a system. Instead of defining a risk using discrete values for its probability and impact, they use metrics to come up with a continuous measure. However, their approach handles only software artifacts and does not support arbitrary development artifacts in general. With our proposed approach, technical risks are detected in any development artifact. Letier et al. [7] state that high level of uncertainty in early phases of software engineering projects is a severe issue. They argue that it is important to correctly estimate the value of information before deciding whether or not efforts should be made to increase system understanding before making decisions. Ohlsson et al. [13] also state that the early detection of high-risk development artifacts is crucial in today's software projects. When using the DesignSpace for development artifact integration, our service provides additional information at basically no cost and thus automatically reduces the level of uncertainty. Moreover, our approach can be applied as soon as development artifacts are defined. Feedback about high-risk-areas and high-risk artifacts is provided continuously and reflects ongoing system evolution. Thus, it provides valueable guidance for decision making at all stages of a development project.

7. CONCLUSIONS AND FUTURE WORK

In this short paper, we have outlined a novel approach for automatically and efficiently detecting high-risk development artifacts in development projects and high-risk-areas within those artifacts. The proposed approach relies on fine-grained development artifact integration through the DesignSpace integration framework and provides continuous information about knowledge reuse. The approach has been implemented and first evaluation results suggest that it is highly scalable. For future work, we plan on comparing the quality of guidance provided by our approach with that given by actual stakeholders that identify risks manually.

[1]DesignSpace framework and prototype service of approach available at: isse.jku.at/tools/dsspc/xadr.zip (pw: dsisse)

Acknowledgments

The research was funded by the Austrian Science Fund (FWF): P25513-N15 and P25289-N15, and the Austrian Center of Competence in Mechatronics (ACCM): C210101.

8. REFERENCES

[1] B. Boehm, S. Koolmanojwong, J. A. Lane, and R. Turner, "Principles for successful systems engineering," *Procedia Computer Science*, vol. 8, pp. 297–302, 2012.

[2] B. W. Boehm, "A spiral model of software development and enhancement," *IEEE Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[3] K. Beck, "Extreme programming: A humanistic discipline of software development," in *FASE*, 1998, pp. 1–6.

[4] L. Rising and N. S. Janoff, "The scrum software development process for small teams," *IEEE Software*, vol. 17, no. 4, pp. 26–32, 2000.

[5] S. V. Shrivastava and U. Rathod, "Categorization of risk factors for distributed agile projects," *Information & Software Technology*, vol. 58, pp. 373–387, 2015.

[6] V. Antinyan, M. Staron, W. Meding, A. Henriksson, J. Hansson, and A. Sandberg, "Defining technical risks in software development," in *IWSM Mensura*, 2014, pp. 66–71.

[7] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *ICSE*, 2014, pp. 883–894.

[8] V. Antinyan, M. Staron, W. Meding, P. Österström, E. Wikstrom, J. Wranker, A. Henriksson, and J. Hansson, "Identifying risky areas of software code in agile/lean software development: An industrial experience report," in *CSMR-WCRE*, 2014, pp. 154–163.

[9] A. Demuth, M. Riedl-Ehrenleitner, A. Nöhrer, P. Hehenberger, K. Zeman, and A. Egyed, "DesignSpace – An Infrastructure for Multi-User/Multi-Tool Engineering," in *SAC*, 2015.

[10] X. Wu, X. Li, R. Feng, G. Xu, J. Hu, and Z. Feng, "OOPN-SRAM: A novel method for software risk assessment," in *ICECCS*, 2014, pp. 150–153.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[12] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[13] N. Ohlsson, A. C. Eriksson, and M. E. Helander, "Early risk-management by identification of fault-prone modules," *Empirical Software Engineering*, vol. 2, no. 2, pp. 166–173, 1997.