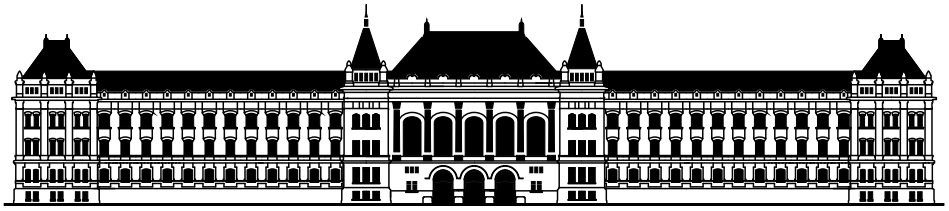

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS

# Models and Algorithms for Integrated Agile Software Planning and Scheduling

by

Ákos Szőke

Supervisor: Prof. András Pataricza, DSc.

Electrical Engineering and Informatics
Department of Measurement and Information Systems

March, 2014

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
Electrical Engineering and Informatics
Department of Measurement and Information Systems

# Models and Algorithms for Integrated Agile Software Planning and Scheduling

PhD. Dissertation Summary

by Ákos Szőke

The focus of my PhD research is on software release and iteration planning to support the development coordination in agile development organizations in both co-located and distributed environments. The main contribution of the research proves the design, application and validation of an Integrated Agile Planning Approach (IAPA). My elaborated IAPA supports development **c**oordinators to cope with the complexity and the dynamics of software development by providing conceptual models, optimization models and algorithms for semi-automatic software release and iteration planning.

Industrial software development is a highly complex and dynamic process. The success of software organizations depends on the effectiveness and efficiency of development activities where development coordination takes a key role. The coordination usually faces a decision problem which aim is to determine which features should be delivered in the next sequence of releases. The resu**l**t of this decision making is manifested by release plans. These plans give a description about which features to implement in which software systems' deliveries to provide maximal business value. Once the features are selected the coordination usually meets an other decision problem which goal is to find out how to realize the planned features. The outcomes of this decision making are expressed by iteration plans, which establish resource allocation to the realization tasks of features while they consider the constraints of delivery.

The presented contribution is fundamentally different from the existing – mainly intuitive – methods in its information fusion and its mathematical optimization approach. Integration of m**a**nagerial and software engineering information, and the provided algorithmic optimization methods easily resolves complex decision situations, gives the business increased visibility, and it can also provide constantly up-to-date decision supports considering changes necessitated by shifting business priorities.

The IAPA was worked out in the frame of the Pythia **r**esearch project which was supported in part by a GVOP grant.

The IAPA is implemented in three prototypes and has been successfully applied in everyday basis for many years at a software development organization. The effectiveness and efficiency of the approach are supported with empirical evidence from industri**a**l case studies and post-mortem simulations.

# Contents

# Acknowledgments

First of all, I would like to thank my supervisor Prof. András Pataricza DSc. for helping my work during the last few years. I learned a lot during this time and I am convinced that this knowledge will help me in the future. He proposed the initial direction of the research for me that triggered several parallel tracks and led to the results described in the dissertation.

I received support for my research from Dr. György Strausz PhD. associate professor also, my colleague at my workplace at Multilogic Ltd – where I have been working as a project leader and a CTO for 10 years. Our relationship dating back prior to the beginning of my PhD studies. Our conversations generated important contributions in the dissertation.

I am very grateful to my colleagues at the Multilogic for the stimulating environment. Discussions with many of them on various topics helped me in my research work. I should also acknowledge Dr. András Balogh PhD. and Dr. István Ráth PhD. (BME MIT FTSRG/OptXware), András Millinghoffer, Dr. Péter Antal associate professor PhD. (BME MIT) and Gábor Kiss (Multilogic) for the a fruitful joint work in Pythia Project that resulted in PYTHIA PROJECT PLANNER™ prototype.

I am grateful to Gábor Károlyi (Deputy Director of the IT Development department at Hungarian Post) and his group, Magdolna Udvardy (Director of IT Strategy department at Hungarian Tax and Financial Control Administration (APEH)), Dr. Sándor Dolgos (Head of IT Development department at BBraun Medical Hungary), Péter Halácsy (Head of Product and co-founder of Prezi.com) and colleagues at Prolan for their support during my visit on the field. The joint work on several projects were very interesting and valuable for me. The inspiring conversations kept me productive enough to make significant progress in my work even during the short visits.

I am also grateful to Dr. Marjan van den Akker PhD. (assistant professor at University of Utrecht) for the data sets to validate my algorithms, Dr. Zornitza Racheva PhD. (associate at University of Twente), Dr. Vladimir Mandic PhD. and Dr. Kari Liukkunen PhD. (associate professors at University of Oulu), and Dr. Val Casey PhD. (assistant professor at Dundalk Institute of Technology) for the fruitful discussions, joint work, and/or their encouragement and support.

I received very valuable feedbacks and inspirations from many agile software practitioners also. Most of them I met at the Budapest Agile Meetup series organized by the Agile Alliance Hungary.

Part of my work has been supported by the GVOP R&D grant (GVOP-3.3.3-05/1.-2005-05-0046/3.0) and realized by the cooperation of Multilogic and OptXware.

Finally, I would like to dedicate this Dissertation to those people who did everything to establish a stable background for me – they gave love, encouragement and support: Klára (my wife), Klárika and Zita (my daugthers), my parents Éva and Sándor. Without their support, I could not have finished this work.

# 1 Synopsis

Modern software development methodologies (e.g. RUP, XP, FDD, Scrum [29, 30, 31, 32]) rely on *Iterative Development Processes (IDP)* and implement different best practices depending on multiple factors like application domain, development environment etc. The common characteristic of different IDPs is the staged-delivery (or incremental) approach which constitutes to cyclic development lifecycle. This lifecycle defines a delivery process of software product versions. This process starts with an initial planning of the software version delivery and in each cycle it ends with software product deployment to the customer's site. The core element of IDP is a harmonization of software version scheduling with the priorities of the different functionalities. Scheduled delivery produces a gradually evolving series of subsets of the final product that increasingly covers the essential needs of the user along his/her priorities. This multifaced process provides a vehicle for analysis and experience based feedback during the development and gradual training for the customers. Its principle is derived from client-valued functionality perspective by delivering tangible full functional software for the individual subproblems within the predefined constraints, namely scope, resource, time and quality [30, 32].

In the late 1990's several new methodologies began to get increasing public attention. Each had a different combination of new ideas and altered old ideas. However, they all emphasized 1) close collaboration between the programmers and business experts; 2) face-to-face communication between programmers and customers (as more efficient than written documentation); 3) frequent delivery of new deliverable business value; and 4) tightly cooperative and self-organizing teams [33]. These characteristics formed the idea of agile software development.

The agile methodology was created as a reaction to the unsuccessfulness of plan-based (or rigid) methodologies – that led to the software crisis in the late 90s – to provide support the adaptation to the frequently changing and rapidly evolving user demands. The plan-based approach can be considered as 'rationalized and engineering-based approach' since it claims that problems are fully specifiable and optimal and predictable solutions exist for every problem therefore it follows the predictive planning strategy. It advocates extensive planning, codified processes, and rigorous reuse to make development an efficient and predictable activity [34, 35]. In contrast with plan-based methods, agile processes address the challenge of an unpredictable world by relying on 'people and their creativity rather than on processes' [34, 35]. Agile methods typically use the adaptive planning approach and it includes typically three kind of plans with frequent feedback loops: a coarse-grained long-time (release), a normal-grained short-time (iteration) and a fine-grained daily plan [36, 37, 38]. Each planning level is responsible for realizing the objectives of both the given and its superior level.

## 1.1 Motivation and Relevancy of the Research

Agile processes offer numerous benefits to organizations including quicker return on investment (ROI), higher product quality, and better customer satisfaction [39]. However they lack sound methodological support of agile release and iteration planning – contrary to the traditional, plan-based approaches. The previously cited survey [39] points out that the second and the fifth principal factors from the identified 26 ones are iteration and release planning respectively. Besides these facts the survey also revealed the 13 most commonly cited greatest concerns listed by respondents about adopting agile within companies. The three out of the five most important ones are 1) the loss of management control (36%), 2) the lack of upfront planning (33%) and 3) the lack of predictability (27%). These considerations are closely

connected with the present *informal* practice of agile planning. These critics underline the importance of providing a more established method for agile planning that lacks of solid theoretical basis currently due to its novelty. The aim of this research is to elaborate methodical support for agile release and iteration planning in both co-located and distributed environments.

In my professional life (I have been working as a project leader and a CTO for 10 years) I faces the before mentioned problems and difficulties on a daily basis. My personal experiences also reinforced the deficiencies of the agile release- and iteration planning: such as lack of upfront planning and the lack of predictability. According to my experiences we could have been persuaded any project sponsors without sound project plans to fund the project (both on the customer's and the developer's side). Therefore, in my practice these deficiencies were surmounted with intuitively and manually constructed project plans – in spite of the fact that this kind of planning resulted in suboptimal and sometimes contradictory plans.

## 1.2 Objectives of the Research

The growing pressure to reduce costs, time-to-market and to improve quality catalyzes transitions to more automated methods and tools in software engineering to support project planning, scheduling and decisions [40]. Although, there are some tenets to manual agile planning [41, 38] algorithmic solution could not be found due to the relatively novel agile software development.

Considering the differences of traditional and agile software process execution (i.e. process sequence, planning lookahead, activity performance), the planning aspect of decision support of agile release and iteration scheduling are *important and up-to-date* research areas since the informal planning and scheduling approaches are highly labor-intensive and error-prone. Moreover, even a minor modification on the input data (e.g. requirements, constraints and objectives) may require rewriting the complete plan – although the perpetual changes in the input data is a basic characteristic of agile environments.

To address this situation the goal of my research was to support scheduling aspect of decisions at release and iteration levels. The research objectives of my work were the following ones:

**RO1** *Improve the productivity of agile software development planning* by introducing interactive, semi-automated methods in different phases of the development process.

**RO2** *Reduce cognitive complexity of agile software development planning* to resolve complex decision situations easily by formulating mathematical models.

**RO3** *Improve the quality of agile software development planning* to provide lower level risks by considering all major planning factors (e.g. dependencies, capacities) in mathematical optimization models.

**RO4** *Support decisions in agile software development planning* to tailor the best plan for the specific project context and users' and/or customers' feedbacks by altering constraints, capacities and priorities.

**RO5** *Improve communication and coordination of distributed agile software development teams* by introducing semi-automated methods for the allocation of features to the distributed teams.

## 1.3 Research Methodology

The introduced objectives determined the direction of my research. The first step of the research was to explore the decision space of agile project planning. For this reason I have constructed a consistent ontology-styled information model to specify the components of this decision space. This model involves the main concepts and their relations.

Next, I have formulated three combinatorial optimization problems to model the three levels of agile project planning. These formulations provide improved efficiency and effectiveness on the different decision levels. The main novelty of my approach lies in *the mathematical precise formulation of the problems*.

Third, I have developed algorithms that solve the formulated optimization problems. The algorithms were implemented as a part of my PROPAS$^{TM}$ (Project Planning and Scheduling) Matlab toolbox. The components of this toolbox are separately published[1] under the MIT licence[2] that makes the free access, reuse and validation possible.

Finally, I validated my agile project planning approaches. To support the experimental validation, I developed an MS SharePoint-based tool – named SERPA$^{TM}$ – according to the elaborated agile conceptual model. This tool was used to collect data for simulations from different real development projects. The tool has been used in everyday basis for many years at a software developer company for project planning and monitoring.

I used two validation types to check, establish and reinforce the findings of the contributions by analyzing them from multiple perspectives – this approach is known as *triangulation*. One of the validation types was a real life experiment (small number of real contexts) and the complementary one was a laboratory experiment (large number of artificial contexts) [42].

The real life experiment helped to validate the contributions at a software developer company in real agile software development contexts. Although every agile development process implementation is different, the applied software process at the selected company can be regarded as typical.

On the other hand, the laboratory experiment helped to generalize the findings of the former experiments by investigating *representative* large number population of generated problems (120, 360 and 480 different cases). In order to model the typical agile development situations I carefully generated the data sets in terms of *problem complexity factors* (such as features, team sizes, temporal dependencies) and *environmental factors* (such as team distribution) that constitute the parameters of the agile planning problems. The values of these factors were used as inputs during the laboratory experiment (simulation).

In order to determine the possible values for the problem complexity factors (parameters) I used the results of literature reviews ([43, 44, 45, 46, 47]), surveys ([37, 39, 48, 49, 50]), conversations with agile practitioners (especially at Agile Hungary meetups[3]) and my personal experiences.

The two kinds of validation types were carried out using three well-known scientific methods:

1. **Post mortem analysis**: I extracted representative data sets from a software development company in order to make a comparison between the algorithmic method and the manual planning. The

[1] https://bitbucket.org/aszoke/matlab or via http://www.kese.hu/
[2] The MIT licence http://opensource.org/licenses/MIT
[3] http://www.meetup.com/AgileHungary/

goal of the analysis was to compare the manual and the optimized approaches using the same input variables.

2. **Case study**: I applied my algorithmic approach parallel with the manual approach in a real-life pilot project to make comparison between the two approaches. A special integrated scheduling tool, named PYTHIA PROJECT PLANNER™ prototypic software, was also implemented to support the study.
3. **Simulation**: I carried out simulations on great number of generated representative data sets – by varying parameters of the different scheduling problem – to get an insight into the performance and quality of the my approach and to filter out the statistical staggering of different agile planning situations and the variance of different development organization.

In the Chapter 7 of my dissertation the Tables 6.1-6.3 summarize the followed research methodology: the identified problems, my elaborated solutions, its validations and the results.

# 2 New Scientific Results

The main contribution of the thesis lies in the design, application and validation of an Integrated Agile Planning Approach (IAPA). The elaborated IAPA supports development coordinators to cope with the complexity and the dynamics of software development by providing mathematical models and algorithms for agile release- and iteration planning. The presented approach is different from the existing approaches in two aspects:

1. *The presented approach proposes* ***mathematical precise formulations of agile planning and scheduling problems****. The problem formulation realizes managerial (resource and timing data) and software engineering (deliverable requirements and defect corrections) information fusion to form a common, consistent repository for both development and coordination. This consistent repository can collect all necessary information for release and iteration planning of software projects. Therefore, it can provide a solid basement of scheduling decisions in both co-located and distributed environments.*
2. *The presented approach proposes* ***mathematical optimization methods to support semi-automatic plan generations*** *that is based on the repository. The provided algorithms support decision making in complex agile planning situations, gives the development process increased visibility, assists in doing what-if analysis, and it can also provide adaptation support considering changes necessitated by shifting development priorities.*

The contributions have additional important characteristics: the presented optimization models represent more general problem classes of planning and scheduling. Since these models only define the structure of the problems, the scheduleable elements (in these cases: features, tasks) may be completely different entities (e.g. packages, CPU operations) in other contexts. Moreover, the presented problem formulations can be easily refined by constraint relaxation (e.g. removing temporal constraints) or by constraint completion (e.g. adding resource intensity constraints) to provide solution to other similar planning and scheduling problems.

I formed three contributions relating to the previous research goals 1.2. *Contribution 1* and *Contribution 2* presents my novel concepts, models and algorithms for agile release planning and agile iteration planning respectively. *Contribution 3* draws up my novel method to extend the agile release planning to distributed

environments. All of my contributions constitute to the building blocks of the Integrated Agile Planning Approach (IAPA). In Figure 1 my contributions are summarized within the agile software development lifecycle to provide a visual overview of the research: *Contribution 1* refers to agile release planning *Contribution 2* relates to agile iteration planning and *Contribution 3* refers to agile feature distribution in distributed development environments – all of them constitute the Integrated Agile Planning Approach (IAPA). This figure points out the inputs and outputs of the different planning steps.

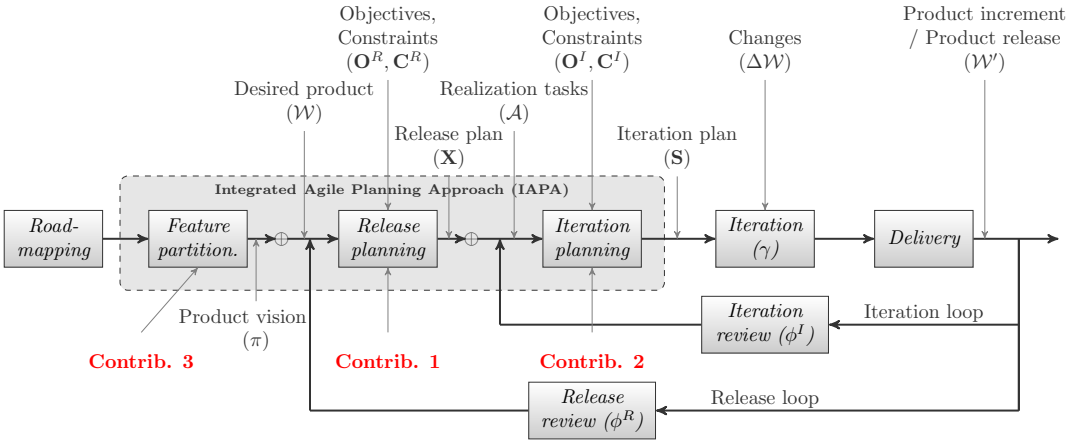


FIGURE 1: Agile Planning Cycles.

## 2.1 Contribution 1

The following subsection draws up my novel concepts, models and algorithms for agile release planning. This elaborated method shows more smooth and fully padded iterations ($> 80\%$ of the total iterations are optimal), and prevents resource overload comparing to the manual approaches. (This contribution is detailed in the Chapter 4 of the dissertation.)

In the following, first my contribution is described, then its most mortem and simulation-based validation are presented, and finally some practical implementation remarks can be found.

**C1: I have elaborated a novel method (namely Agile Release Scheduling (ARS)) for agile release planning to provide improved efficiency and effectiveness in release-centered decisions.** [C6, E13, B2, G18, I26]

**C1.1** *I have constructed a consistent ontology-styled information model of agile release scheduling (namely Agile Release Scheduling Model (ARSM)) to specify the components of the agile release scheduling decision space. This model involves the main concepts and their relations, and can be applied as a conceptual model of agile release planning tools.* [C6]

**C1.2** *I have formulated agile release scheduling as an optimization problem (namely Agile Release Scheduling Problem (ARSP)) to provide features' allocation considering team's resource capacity and minimized number of iterations as planning objective. The main novelty of my approach lies in the mathematical precise formulation of the problem.*

*The elaborated problem formulation is solved with an extension of the BINARY MULTIPLE KNAPSACK optimization model to cover wide-ranging release planning situations with the expression of: 1) date-driven/scope-driven scheduling, 2) dependencies between features, 3) team capacities, 4) feature priorities, 5) staged-delivery and maximal deliverable value to the customer.* [C6, E13, B2]

**C1.3** *I have developed a branch-and-bound binary multiple knapsack algorithm, which finds the global optimum (time complexity is $\mathcal{O}(o * n^2)$, where $n$ and $o$ is the number of features and iterations respectively)* [C6, E13].

### 2.1.1 Mathematical Precise Formulation of Agile Release Planning

The essential aim of release planning ($\mathfrak{S}_{AR}$) is to determine a *feasible* coarse-grained plan for the development that determines which feature ($\mathcal{W}$) in which iteration ($\mathcal{I}$) will be delivered ($\mathbf{X}$ – feature-to-iteration assignments). The *optimized* version of the release planning problem can be derived by selecting the extreme-valued plan from the potentially feasible alternatives. This optimal release plan can be considered as a representative of KNAPSACK problems (KP) in which a set of items (features) are given with their value (business priority) and size (required effort), and it is desired to select one or more disjoint subsets (iterations) so that the sum of the sizes in each subset does not exceed a given bound and the sum of the selected values is maximized [51].

Without sound decision support the following typical constraint (i.e. $\mathbf{C}^R$) are managed *informally* (implicitly and intuitively) by development coordinators: 1) *iterations* ($\mathcal{I}$ – where features should be assigned), 2) *dependencies* ($\mathcal{D}$ – temporal constraints between features), 3) *resource capacities* ($\mathcal{R}$ – set of developers, $e$ – team effectiveness factor, $\mathbf{c}$ – resource demands during releases), 4) *priorities* ($\mathbf{p}$ – importance of each feature delivery), 5) *effort* ($\mathbf{w}$ – effort for developing of each features), 6) *release length* ($l^R$), and 7) *iteration length* ($l^I_k$).

Therefore, optimality of plans (i.e. maximal business value) is heavily based on the development coordinator's right senses – nevertheless optimized project plans are crucial from technical and economic point of view. More formally, I can say that the design space of agile release scheduling is made up of the previous factors ($\mathcal{W}, \mathcal{I}, \mathcal{D}, \mathcal{R}, e, \mathbf{c}, \mathbf{p}, \mathbf{w}, \mathbf{l^R}, \mathbf{l^I_k}, \mathbf{X}$) – in short ($\mathcal{W}, \mathbf{C}^R$) –, and its objective is to find an optimal mapping of

$$\mathfrak{S}_{AR} : (\mathcal{W}, \mathbf{C}^R) \rightarrow \mathbf{X} \tag{1}$$

while it considers maximal value delivery.

### 2.1.2 Experimental Validation of Contribution 1

To validate my contribution, first I carried out a post mortem analysis on seven real-life representative data sets extracted from a software development company [52]. Although every agile development process implementation is different, the applied software process at the selected company can be regarded as typical in terms of organization size (6 developers), applied agile methods (Scrum-like development process) and techniques (XP development practices). At this organization, the release scheduling process is made up of the typical agile planning steps. The goal of analysis was to compare the manual and the optimized approaches using the same input variables.

Additionally, I also carried out simulations on 120 generated representative data sets – by varying parameters of the scheduling problem (i.e. resource capacities ($\mathcal{R}$) iteration capacity ($\mathbf{c}$), release length ($l^R$), iteration length ($l^I_k$) and dependencies ($\mathcal{D}$)) – to get an insight into the performance and quality of the presented approach and to filter out the statistical staggering of different agile planning problems.

**Post Mortem Analysis and Computational Experiments**
During validation the following key questions were addressed:

- **C1Q1**: *What are the staffing requirements over time?* (i.e. dynamics of $\mathcal{R}$);
- **C1Q2**: *How many iterations do we need per release?*; (i.e. $o \triangleq l^R/l^I$) and
- **C1Q3**: *How buffers for contingencies are allocated?* (i.e. $BpR \triangleq \Delta\mathcal{R} = \mathcal{R}_{planned} - \mathcal{R}_{actual}$)

In the historical case, $17\%$, $59\%$, $17\%$ of the total iterations were over- or underloaded more than or equal to $20\%$, more than or equal to $10\%$ but less than $20\%$, less than $10\%$ respectively. Only $7\%$ of the total iterations were optimal. In contrast, in the optimized case, $0\%$, $4\%$, $12\%$ of the total iterations were over- or underloaded more than or equal to $20\%$, more than or equal to $10\%$ but less than $20\%$, less than $10\%$ respectively. But $84\%$ of the total iterations were optimal. As a result, in the optimized case: staffing requirements (c.f. **C1Q1**) showed more smooth and fully padded iterations that mean the algorithm strives to 1) prevent resource overload – which often yields increasing delivery risks, and prevent resource underload – which captivates economically badly utilized iterations.

Iteration counts per releases (cf. **C1Q2**) of the optimized case exhibited slight differences: in some situations (3 cases out of the total 7 cases) the algorithm had to add one more iteration to prevent resource overload. Therefore, in real-life situations, the algorithm should be used in an iterative manner by altering the schedule parameters (e.g. altering $l^I$ or $\mathcal{R}$) – which leads to economically even better schedules.

Finally, if we consider the buffer per releases (c.f. **C1Q3**) that is used for contingencies in practice, we can realize major differences also. The optimized case points out the buffer allocation property of the algorithm: time buffers (cf. $BpR$) are moved to the end of releases due to the optimality criteria (packing as many items into the iteration as possible). This characteristic indicates that the contingencies are moved to the end of the release while dependencies are considered by the algorithm, which is more advisable to mitigate risks of delivery slippage [53].

Based on these data sets, the quality of the release plans was defined in terms of the distances ($\Delta$) from the theoretical maximal resource utilization ($\mathbf{c}$). This distance characterizes the underload of iterations only, since overloads are avoided by the $\mathbf{c}$ constraint of the release scheduling model. As a result of the simulation, I can statistically conclude that the number of dependencies ($\mathcal{D}$) negatively correlate with the degree of resource utilization – therefore, the degree of release plan quality. On the other hand, both the number of features and the extent of iteration capacity ($\mathbf{c}$) positively correlate with the degree of plan quality. However, in all simulation cases, the optimal packing was reached more than $95\%$ of the total cases. Comparing all the differently parameterized simulations with the historical case, despite the simulation problems were much more complex than the historical one, the algorithmic approach could outperform the manual approach in terms of release plan quality. The simulation reinforced the findings of the post mortem analysis.

The results of experiments on representative real-life data sets and the numerous representative generated data sets (120) indicate in the same way that my approach provides practical value as a decision support method for agile release planning.

### 2.1.3 Practical Implementation for the Experimental Validation

To support the experimental validation, on the one hand, I developed an MS SharePoint-based tool – named SERPA™ – according to the conceptual model of release planning (c.f. **C1.1**) that is presently used at [52]. On the other hand, I implemented a release planning algorithm (c.f. **C1.2-3**), namely `mksched`, in Matlab [54]. The algorithm is implemented as a part of my PROPAS™ (Project Planning and Scheduling) Matlab toolbox. This tool was used to support the simulations.

## 2.2 Contribution 2

The following subsection draws up my novel concepts, models and algorithms for agile iteration planning. This elaborated method significantly improves load balancing of resources ($\approx 4-5\times$), significantly accelerates iteration scheduling production ($> 50\%$), enables more than $10\%$ increase in project execution's efficiency, and more than $50\%$ growth in planning efficiency comparing to the traditional approaches. (This contribution is detailed in the Chapter 5 of the dissertation.)

In the following, first my contribution is described, then its case study and simulation-based validation are presented, and finally some practical implementation remarks can be found.

**C2: I have elaborated a novel method (namely Agile Iteration Scheduling (AIS)) for iteration planning to provide improved efficiency and effectiveness in iteration-centered decisions.** [C5, I26, D7, E9, E14, E11, E8, I25, G18, G16]

**C2.1** *I have constructed a consistent ontology-styled information model of agile iteration scheduling (namely Agile Iteration Scheduling Model (AISM)) to specify the components of the agile iteration scheduling decision space. This model involves the main concepts and their relations, and can be applied as a conceptual model of agile iteration planning tools.* [C6, C5]

**C2.2** *I have formulated agile iteration scheduling as an optimization problem (namely Agile Iteration Scheduling Problem (AISP)) to provide governance support in feature implementation sequencing. The main novelty of my approach lies in the mathematical precise formulation of the problem. The elaborated problem formulation is solved with an extension of the RESOURCE-CONSTRAINED PROJECT SCHEDULING (RCPS) model that considers temporal constraints, team's resources, and defines makespan minimization scheduling objective. Moreover, it is an extension of the RCPSP to cover the iteration scheduling situations with the expression of 1) pre-assignments (i.e. assigning certain tasks to resources before scheduling) and 2) timeboxed iteration duration control.* [C5]

**C2.3** *I have developed a heuristic scheduling algorithm for the RCPSP problem. Its greedy strategy misses the global optimal solution, but produces quick (time complexity is clearly $\mathcal{O}(n+m)$) and usually sufficient results for practical applications.* [C5]

**C2.4** *I have constructed an information model to extend UML diagrams with planning objectives and constraints. The extension is constructed with the application of standard UML Profile extension mechanism [55]. I have formulated a mathematical formulae to determine the required effort for implementing a Use case model element based on the popular Use-case point estimation method (UCPM) [56]. With this information fusion*

*and the formulated equation semi-automatic model-driven planning can be exploited from UML artefacts applying the previous results (**C2.2-3**). Additionally, I developed a graph transformation to create an AoN graph (Activity-on-Node) from the schedule in order to make schedule visualization possible in popular project planning tools such as [57].* [E9, E14, E11, E8]

**C2.5** *I have constructed a generic extension of UML-based specification with business objectives, which are expressed with the Goal-oriented Requirements Language to form a common, consistent repository for both development and management. This information fusion enables easy alignment of system requirements with business needs. Moreover, it also enables automatic schedule generation applying the previous results (**C2.2-4**).* [E9, E14, E11, E8]

### 2.2.1 Mathematical Precise Formulation of Agile Iteration Scheduling

Generally, iteration scheduling ($\mathfrak{S}_{AI}$) aims at determining a *feasible* fine-grained plan for the development that schedules the implementation of selected features within an iteration ($\mathbf{S}$ – task-to-developer assignments ($\mathcal{A}$)) [38]. The *optimized* version of the iteration scheduling problem can be derived by selecting the extreme-valued schedule from the potentially feasible alternatives. This problem can be considered as a representative of RESOURCE-CONSTRAINED PROJECT SCHEDULING problem (RCPSP), in which the resource allocation consists in assigning time intervals to the execution of the activities (realization tasks) while taking into account both temporal constraints (precedences between tasks) and resource constraints (resource availability) and the minimal execution time objective [58].

Iteration planning is usually a complex task due to the following typical constraints (i.e. $\mathbf{C}^I$): 1) *precedences* ($\mathbf{P}$ – temporal precedences between realizations), 2) *balancing resource workloads* ($\mathcal{R}$ – avoiding resources overloading), 3) *effort* ($\mathbf{w}$ – effort for implementing of each task), 4) *pre-assignment* ($\mathbf{a}$ – manual assignment of the appropriate developer to some tasks) and 5) *iteration length* ($l^I$ – deadline of iteration end to provide staged-delivery).

In traditional approaches, scheduling is usually carried out by a project scheduling software package (e.g. [57]) that helps coping with constraints and objectives. These approaches require manual construction and take relatively long time (several hours) they are too heavyweight for IDP (particularly in agile environments) therefore often omitted. However without adequate decision-support they are managed *informally* (implicitly and intuitively) whose inherent discrepancies must be resolved during team's meetings [36, 38]. Informal approaches work well in smaller projects, however as the size and complexity increases scheduling becomes a very complex process and advocates tool support [59, 60]. More formally, we can say that the design space of agile iteration scheduling is made up of the previous factors ($\mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{w}, \mathbf{a}, \mathbf{l^I}, \mathbf{S}$) – in short: ($\mathcal{A}, \mathbf{C}^I, \mathbf{S}$) – , and its objective is to find an optimal mapping of

$$\mathfrak{S}_{AI} : (\mathcal{A}, \mathbf{C}^I) \rightarrow \mathbf{S} \tag{2}$$

while it considers minimal execution time objective.

### 2.2.2 Experimental Validation of Contribution 2

To evaluate the **C2.1**, **C2.2** and **C2.3** parts of my contribution, first I carried out a post mortem analysis on four real-life representative data sets extracted from a software development company [52]. Additionally,

a longitudinal experiment (a case study) was carried out to evaluate the **C2.4** and **C2.5** parts of my contribution on a real-life software planning situation. The pilot project was the whole module of a real-life application that is developed by [52]. Although every agile development process implementation is different, the applied software process at the selected company can be regarded as typical in terms of organization size ($6-8$ developers), applied agile methods (Scrum-like development process) and techniques (XP development practices). At this organization, the iteration scheduling process is made up of the typical agile planning steps (see Sec.2.2.1). The analysis goal was to compare the manually constructed informal and the semi-automatically constructed optimized approaches using the same input variables.

Additionally, I also carried out simulations on $480$ generated representative data sets – by varying parameters of the iteration problem (see Sec. 2.2.1: effort for developing technical tasks ($w$), resource capacities ($\mathcal{R}$), iteration capacity ($\mathbf{c}$), iteration length ($l_k^I$) and precedences ($\mathbf{P}$)) – to get an insight into the performance and quality of the presented approach and to filter out the statistical staggering of different agile iteration planning problems.

**Post Mortem Analysis and Computational Experiments for C2.1, C2.2 and C2.3**
During validation the following key questions were addressed:

*How does optimization-based iteration scheduling compare with informal one in terms of*

- **C2Q1**: *resource workload over time?* (i.e. dynamics of $\mathcal{R}$);
- **C2Q2**: *quality of the plans?* (i.e. $l^I$ is not exceeded and $P$ are dealt with automatically) and
- **C2Q3**: *feasibility of the plans?* (i.e. workload of $\mathcal{R}$)

The simulations demonstrated how the method could easily handle precedence constraints, significantly improve load balancing of resources (cca. $4-5\times$), produce higher quality and lower-risk feasible schedule, and finally, provide improved support in decision making for development teams (c.f. **C2Q1**).

As a result, in terms of coefficient variation (i.e. normalized measure of dispersion), the optimization-based scheduling provides $\approx 4-5\times$ a more balanced resource workload over time contrary to the intuitive method (c.f. **C2Q1**). The algorithmic method easily resolves the complex decision situation – as it handles precedences between tasks and avoids resource workloads – contrary to the historical case where these are managed intuitively during daily meeting. As a consequence these two capabilities of the algorithmic method ensure higher quality and lower-risk feasible plans in contrast to the historical case (c.f. **C2Q2-3**). The simulation on the artificially generated representative data sets reinforced the findings of the post mortem analysis.

The results of experiments on representative real-life data sets and the numerous representative data set ($480$ different cases) indicate in the same way that my approach provides practical value as a decision support method for agile environments.

**Case Study for C2.4 and C2.5**
A case study was carried out to evaluate my elaborated method (**C2.4**, **C2.5**) on a real-life software planning situation [61]. The pilot project was a whole module of a real-life application that is developed by [52].

To validate my elaborated method I stated the following hypotheses:

- **C2H1**: *The elaborated method (and tool) with the formulation of software requirements metrics, planning constraints and objectives as RCPSP enables improved production (i.e. less manual work) in project iteration planning contrary to the traditional and mainly manual method.*;
- **C2H2**: *The elaborated method supports what-if analysis by different parameterizing of the scheduling algorithm which leads to improved decision support (i.e. choose from several alternatives) in iteration planning decisions.*;
- **C2H3**: *The elaborated method with the derivation of iteration plans from requirements specification leads to precise requirement's level tracing.*;
- **C2H4**: *Fusioning of requirements metrics, project planning constraints and objectives leads to improved efficiency in project execution in addition to less synchronization and documentation overhead.*; and
- **C2H5**: *Derivation of AISP from software requirement specification enables improved production in project planning contrary to the application of the traditional, mainly manual, project planning method.*

This case study demonstrated that how the method could 1) significantly accelerate project scheduling production ($> 50\%$), 2) provide improved support in decision making, 3) supply precise Use case level requirement's tracing, and this method enabled 4) more than $10\%$ increase in project execution's efficiency, and 5) more than $50\%$ growth in planning efficiency in addition to less synchronization and documentation overhead.

### 2.2.3 Practical Implementation for the Experimental Validation

To promote experimental validation, I extended the SERPA™ web application with conceptual model of iteration planning (c.f. **C2.1**) at [52]. Additionally, I also implemented an iteration planning algorithm (c.f. **C2.2-3**), namely `lscap`, as a part of my PROPAS™ Matlab toolbox. These tools were used during the simulations. Moreover, a special integrated scheduling tool, named PYTHIA PROJECT PLANNER™ prototypic software, was also implemented with the UML2 technology on the Eclipse platform [62, 63]. This tool was used to support the case study.

## 2.3 Contribution 3

The following subsection draws up my novel concepts, models and algorithms for agile distributed release planning. This elaborated method necessitates $\approx 14 - 18\times$ less intensive communication and coordination needs and can provide $40\%$ better utilization of resources comparing to the traditional distributed agile release planning approaches. (This contribution is detailed in the Chapter 6 of the dissertation.)

In the following, first my contribution is described, then its validation is presented, and finally some practical implementation remarks can be found.

**C3: I have elaborated a novel method (namely Feature Partitioning Method) for distributed agile release planning to improve efficiency and effectiveness in release-centered decisions in distributed agile environments.** [C4, A1, E13, C3]

**C3.1** *I have defined Feature Architectural Similarity Analysis (FASA) analytic method to*

*determine architectural similarities between features (deliverable functional and non-functional requirements) that can be exploited to identify features that are implemented in the similar sets of software modules.* [C4, A1, C3]

**C3.2** *I have defined a Feature Chunk Construction method – by formulating a Feature Chunk Construction optimization problem (FCCP) – that rules the distribution of development work across sites considering minimization of communication and coordination needs among the dispersed teams. The main novelty of my approach lies in the mathematical precise formulation of the problem. The elaborated problem formulation is solved with edge cutting minimization optimization model (MINIMUM K-CUT) of graph partitioning. With this approach, by arranging development work according to the identified partitions, the communication needs and coordination complexity of the distributed team can be significantly decreased.* [C4]

### 2.3.1 Mathematical Precise Formulation of Distributed Extension of Agile Release Planning

Distributed extension of agile release planning ($\mathfrak{A}_F$) can be defined as a decision making process, where the goal is to determine an *feasible* feature chunk assignment to each distributed team before the local agile release planning is carried out. The *optimized* version of this problem can be derived by selecting the extreme-valued plan from the potentially feasible alternatives. This problem can be considered as a representative of *MINIMUM CUT* graph partitioning problem that requires finding a set of edges (communication paths) between vertices (teams) whose removal would partition the graph into connected components (feature partitions) [64]. The minimization objective reflects the aim of minimizing the intensity of communication and synchronization needs in order to minimize their negative effects – such as reduced team productivity, increased production interval, increased communication cost and difficult process control across distributed teams [43, 41, 65, 66].

Feature chunks ($\mathbf{W}^*$) are usually identified according to their *cohesiveness* from the standpoint of *architectural impact*. The higher the cohesiveness between features, the stronger the need to group those features together. To identify cohesiveness between features, I introduce a binary relation between features ($\mathbf{W}$) and software modules ($\mathcal{M}$), called `ImplementedIn` ($\otimes$), to express the fact that a given feature is implemented in a given software module. My goal is to group those features which are to be implemented in the similar set of software modules, i.e. they require similar architectural impact. With this approach, arranging development work (features) according to the identified feature chunks, it can significantly decrease the communication needs and coordination complexity of the distributed team ($\mathcal{T}$). Traditionally, the feature chunk composition is manually accomplished that requires relatively long time (several hours) and the optimality objective (partitioning the features into $k$ cohesive chunks) hardly can be realized. More formally, we can say that the design space of feature distribution is made up of the previous factors ($\mathcal{W}, \mathcal{M}, \mathcal{T}, \otimes, \mathbf{W}^*$), and its objective is to find an optimal $k$-partitioning of

$$\mathfrak{A}_F : (\mathcal{W}, \mathcal{M}, \mathcal{T}, \otimes) \rightarrow \mathbf{W}^* \tag{3}$$

### 2.3.2 Experimental Validation of Contribution 3

To validate my contribution, I carried out a post mortem analysis on seven real-life representative data sets extracted from a software development company [52]. Although every agile development process

implementation is different, the applied software process at the selected company can be regarded as typical in terms of organization size (18 developers), applied agile methods (Scrum-like development process) and techniques (XP development practices). At this organization, the release scheduling process is made up of the typical agile planning steps. The fix-membered teams worked in different locations, so they could not see or speak often in person that resulted from geographical separation. Communication was mostly based on video conferences, phone calls and emails; since all developers were Hungarians, there was no language, cultural or time zone barriers. The goal of analysis was to compare the manual and the optimized approaches using the same input variables.

Additionally, I also carried out simulations on 360 generated representative data sets – by varying parameters of the iteration problem (see Sec. 2.3.1: feature counts ($\mathbf{W}$), team settings ($\mathcal{T}$) and module relatedness ($\otimes$)) – to get an insight into the performance and quality of the presented approach and to filter out the statistical staggering of different agile iteration planning problems.

**Post Mortem Analysis and Computational Experiments**
During validation the following key questions were addressed:

*How does Feature Partitioning Method can be compared with the intuitive one in terms of*

- **C3Q1**: *communication and coordination needs?* (i.e. $\nabla_{CCI} = f(\phi^I, \phi^R)$);
- **C3Q2**: *resource workload?* (i.e. $\nabla_{Res} = f(\mathcal{R})$); and
- **C3Q3**: *feasibility of the plans?* (i.e. workload of $\mathcal{R}$)

As a result, in the optimized case: 1) can break down the development work into cohesive feature chunks such a way that the dispersed teams are roughly working on the same set of modules. Therefore, communications predominately occur within grouped and jointly implemented features rather than between grouped features; 2) necessitates $\nabla_{CCI} \approx 14 - 18\times$ less intensive communication and coordination needs than the intuitive ones (c.f. **C3Q1**); 3) can provide $\nabla_{Res} = 40\%$ better utilization of resources that can provide lower implementation risk and more economical resource exploitation by decreasing resource over- and underload respectively (c.f. **C3Q2**); and 4) can provide higher quality feature distribution plans with the utilization of semi-automatic production of feature chunks, which makes it possible to re-partition the features any time within seconds in order to support what-if analysis or to adapt the plan to the continuously changing situations of agile environments (c.f. **C3Q3**). The simulation on the artificially generated representative data sets reinforced the findings of the post mortem analysis.

The results of experiments on representative real-life data sets and the numerous representative data set (360 different cases) indicate in the same way that my approach provides practical value as a decision support method for distributed agile environments.

### 2.3.3 Practical Implementation for the Experimental Validation

To promote experimental validation, I implemented the Kernighan-Lin (KL) graph partitioning method [67, 68] as a part of my PROPAS$^{TM}$ Matlab toolbox. This tool was used to support the simulations.

# 3 Practical Use

The targeted problems were specified, and the applied methodology was worked out in the frame of the Pythia research project[4] [73]. The Integrated Agile Planning Approach (IAPA) methodology was successfully used in several projects at a large international bank.

The scope of Pythia project also included the demonstration of the industrial applicability of the theoretical results through my three prototypes:

- **SERPA™ *prototype***: I realized a MS SharePoint-based web application [74, 52]. SharePoint is browser-based collaboration and a document-management platform, and its capability includes creating different lists. The previously constructed release and iteration information models were implemented as SharePoint lists. Thus, the portal was targeted as a collaborative workspace for developers, and a tool for the management to collect all planning information. With this web-based tool, developers can break-down features into technical tasks, indicate precedences, set effort estimation, status of tasks/defect corrections, and they also can share this information to facilitate communication. This prototype is actually a complete software. It is used in everyday basis for many years at a software developer company for project planning and monitoring[5].
- **PROPAS™ *prototype***: I have implemented the presented release and iteration scheduling and partitioning algorithms in Matlab to support managerial decisions based on data collected through the Serpa site [54]. My PROPAS™ toolbox not only contains the implemented planning, scheduling and partitioning algorithms, but involves graph theoretical and visualization capabilities also[6].
- **PYTHIA PROJECT PLANNER™ *prototype***: I[7] have implemented a special integrated scheduling approach with the UML2 technology on the Eclipse platform [62, 63]. I have chosen the popular Profile mechanism of UML2 to inject planning information to UML models and finally, XML input were produced by this tool for a popular project scheduling tool [57]. My contribution regarded to the specification, the design, and partly the implementation (information model and the scheduling algorithms) of the system. This prototype not only involves scheduling capabilities but also includes decision support capabilities based on Bayesian belief networks (BBN) technology for representing and analyzing causal models involving uncertainty. It provides a set of tools for constructing probabilistic inference and decision support systems on BBNs[8].

The effectiveness and efficiency of the approach are supported with empirical evidence from two industrial case studies ([E9]) and three post-mortem simulations on different representative data sets ([C5, E13, C6]). The results of experiments on representative real-life data sets and representative case study indicate that my approach can provide practical value as a decision support method for planning of agile software development projects.

Finally, I believe that my contributions of this research fit to SEMAT's endeavor with providing sound theory and methodical support to the agile software release and iteration planning field of software engineering.

[4]The development is supported in part by the GVOP grant (GVOP-3.3.3-05/1.-2005-05-0046/3.0). The solution is also implemented by the Multilogic Ltd [52] and OptXware Llc. [69]. Validation of the case study results are performed at Multilogic Ltd [52], and the simulation data sets were gained from the IT department of Hungarian Post [70], IT department at Hungarian Tax and Financial Control Administration (APEH) [71], Prolan Corp. [72] and from Multilogic Ltd [52].

[5]The prototype is applied at Multilogic Ltd.

[6]`https://bitbucket.org/aszoke/matlab` or `http://www.kese.hu/`

[7]The implementation was carried out with the cooperation of Multilogic Ltd, and OptXware Llc.

[8]The BBN-based decision support capabilities was developed in the cooperation of Multilogic and the Intelligent Systems Research Group of Measurements and Information System Department at Budapest University of Technology and Economics [75].

# 4 List of Publications

## Book Chapters

[A1] Akos Szoke. "Horizons in Computer Science Research". In: ed. by Thomas S. Clary. Vol. 5. New York: Nova Science Publishers, 2011. Chap. A Workload-balancing Method for Agile Distributed Software Development Environments, pp. 161–193.

## Articles in an Edited Book

[B2] Akos Szoke. "Bin-packing-based Planning of Agile Releases (OUTSTANDING RESEARCH PAPER)". In: *ENASE 2008/2009 Revised Best Papers*. Ed. by Stefan Jablonski Leszek Maciaszek César González-Pérez. Vol. 64. Communications in Computer and Information Science (CCIS). May 9-10, 2009. Milano, Italy: Springer-Verlag, May 2010, pp. 133–146.

## Journal Papers Published Abroad

[C3] Akos Szoke. "Optimized Feature Distribution in Distributed Agile Environments". In: *PROFES '10: Proceedings of the 11th International Conference on Product Focused Software Process Improvement*. Ed. by Muhammad Ali Babar, Matias Vierimaa, and Markku Oivo. Vol. 6156. Lecture Notes in Computer Science. June 21-23, 2010. Limerick, Ireland: Springer-Verlag, June 2010, pp. 62–76. ISBN: 978-3-642-13791-4.

[C4] Akos Szoke. "A Feature Partitioning Method for Distributed Agile Release Planning (BEST RESEARCH PAPER)". In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by Will Aalst et al. Vol. 77. Lecture Notes in Business Information Processing ISBN: 978-3-642-20677-1. May 10-14, 2011. Madrid, Spain: Springer-Verlag, May 2011, pp. 27–42.

[C5] Akos Szoke. "Decision Support for Iteration Scheduling in Agile Environments". In: *PROFES '09: Proceedings of the 10th International Conference on Product Focused Software Process Improvement*. Ed. by Frank Bomarius et al. Vol. 32. Lecture Notes in Business Information Processing. June 15-17, 2009. Oulu, Finnland: Springer-Verlag, June 2009, pp. 156–170. ISBN: 978-3-642-02151-0.

[C6] Akos Szoke. "Conceptual Scheduling Model and Optimized Release Scheduling for Agile Environments". In: *Information and Software Technology* 53.ISSN: 0950-5849 (June 2011), pp. 574–591.

## Journal Papers Published in Hungary

[D7] Akos Szoke, Orsolya Doban, Andras Pataricza. "Quality-driven Optimized Resource Allocation". In: *Periodica Politechnica Electrical Engineering* 54 (Feb. 2010), pp. 71–78. URL: http://www.pp.bme.hu/ee/2010_1/pdf/ee2010_1_07.pdf.

## International Conference Papers

[E8] Akos Szoke. "Use case-driven Project Planning in System Development Projects". In: *51st EOQ Annual Congress*. European Organization for Quality Annual Congress. May 22-23, 2007. Czech Society for Quality. Prague, Czech Republic, May 2007.

[E9] Akos Szoke. "A Proposed Method for Release Planning from Use Case-based Requirements". In: *Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications*. Euromicro SEAA. September 3-5, 2008. Parma, Italy: IEEE Computer Society, Sept. 2008, pp. 449–456.

[E10] Akos Szoke. "Quality-driven Software Development". In: *The Fourth Conference of PhD Students in Computer Science*. Ed. by Tibor Csendes. CS2. Volume of extended abstracts July 1–4, 2004. Institute of Informatics of the University of Szeged. Szeged, Hungary, July 2004, p. 118.

[E11] Akos Szoke. "A Proposed Method for Project Planning from UML Models". In: *Work in Progress Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications*. Ed. by Erwin Grosspietsch and Konrad Klöckner. Euromicro SEAA ISBN: 978-3-902457-20-3. September 3-5, 2008. Parma, Italy: SEA-Publications: SEA-SR-20, 2008.

[E12] Andras Pataricza, Orsolya Doban, Akos Szoke. "Costs/Benefits of Using Formal Methods". In: *Proceedings of The International Conference on Dependable Systems and Networks*. DSN. Supplemental Volume of the 2004 International Conference on Dependable Systems and Networks June 28 - July 1, 2004 http://2004.dsn.org/sessions.html. Palazzo dei Congressi, Florence, Italy, June 2004, pp. 104–105.

[E13] Akos Szoke. "Agile Release Planning through Optimization". In: *ENASE 2009 Fourth International Conference on Evaluation of Novel Approaches to Software Engineering Proceedings*. Ed. by Leszek Maciaszek Stefan Jablonski. May 9-10, 2009. INSTICC. Milan, Italy: INSTICC Press, May 2009, pp. 149–160.

[E14] Akos Szoke. "A Proposed Method for for Automated Project Scheduling using Goals and Scenarios". In: *Proceedings of the 16th IEEE International Requirements Engineering Conference*. IEEE Requirements Engineering. September 8-12, 2008. IEEE Computer Society. Barceona, Spain: IEEE Computer Society, Sept. 2008, pp. 339–340.

## Journal Papers in Hungarian

[F15] Akos Szoke. "Szoftverprojektek Bayes-háló-alapú kockázatanalízise". In: *Magyar Távközlés* 2.HU ISSN: 0856-9648 (June 2006), pp. 18–24.

## Hungarian Conference Papers

[G16] Akos Szoke. "Quality Metrics and Models in Software Development Processes". In: *Proceedings of the 11th PhD Mini-Symposium*. ISBN: 963 420 785 5. February 3-4, 2004. Budapest University of Technology and Economics. Budapest, Hungary, Feb. 2004, pp. 36–37.

[G17] Akos Szoke. "Project Risk Management Based on Bayes Belief Net Using EMF". In: *Proceedings of the 13th PhD Mini-Symposium*. ISBN: 963 420 853 3. February 6–7, 2006. Budapest University of Technology and Economics. Budapest, Hungary, Feb. 2006, pp. 70–71.

[G18] Akos Szoke. “Improving Quality in Agile Software Development”. In: *Proceedings of the 12th PhD Mini-Symposium*. February 8-9, 2005. Budapest University of Technology and Economics. Budapest, Hungary, Feb. 2005, pp. 42–43.

## Publications on Other Fields

[H19] Akos Szoke, András Förhécz, and György Strausz. “Versioned linking of semantic enrichment to legal documents”. In: *Artificial Intelligence and Law Journal* Special 25th Anniversary Issue (4 2013), pp. 485–519.

[H20] Akos Szoke, András Förhécz, and György Strausz. “A Unified Change Management of Regulations and their Formal Representations based on the FRBR Framework and the Direct Method”. In: *Legal Knowledge and Information Systems - JURIX 2012*. Ed. by Burkhard Schäfer. Frontiers in Artificial Intelligence and Applications. IOS Press, 2012, pp. 147–156.

[H21] Akos Szoke et al. “Linking Semantic Enrichment to Legal Documents”. In: *Proceedings of the Workshop on Modelling Policy-making (JURIX-MPM 2011)*. Ed. by Adam Wyner and Neil Benn. 2011, pp. 11–15.

[H22] Gabor Korosi, Akos Szoke. “Elektronikus közbeszerzés II.” In: *Jegyző és Közigazgatás* 6.ISSN: 1589-3383 (Nov. 2003), p. 43.

[H23] Peter Halacsy, Gabor Kiss, Akos Szoke. “Az információ-visszakeresés modelljei”. In: *Magyar Távközlés* 3 (2003), pp. 30–37.

[H24] Akos Szoke, Krisztián Mácsár, and György Strausz. “A Text Analysis Framework for Automatic Semantic Knowledge Representation of Legal Documents”. In: *Workshop on Network Analysis in Law (In conjunction with ICAIL 2013: 14th International Conference on AI and Law)*. Ed. by Radboud Winkels. Rome, May 2013.

## Other Works

[I25] Akos Szoke. *Quality Metrics and Models*. Tech. rep. Budapest University of Technology and Economics, Jan. 2004.

[I26] Akos Szoke, Orsolya Doban, Andras Pataricza. *Minőségvezérelt projektmenedzsment*. Magic days (presentation). Visegrad, Hungary. June 2005.

[I27] Akos Szoke. *IT projektmenedzsment kvantitatív alapú döntéstámogatása*. Metódus day (presentation). Budapest, Hungary. Feb. 2007.

[I28] Akos Szoke. *Esettanulmány: Szoftverfejlesztési folyamat nemzetközi banki környezetben*. Agile Alliance Hungary Meetup (presentation). Budapest, Hungary. Mar. 2011.

# 5 References


[29] Per Kroll and Philippe Kruchten. *The rational unified process made easy: a practitioner's guide to the RUP*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0-321-16609-4.

[30] Kent Beck and Cynthia Andres. *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional, Nov. 2004. ISBN: 0321278658.

[31] Steve R. Palmer and Mac Felsing. *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001. ISBN: 0130676152.

[32] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN: 0130676349.

[34] Barry Boehm. "Get Ready for Agile Methods, with Care". In: *Computer* 35.1 (2002), pp. 64–69. ISSN: 0018-9162.

[35] Tore Dybå. "Improvisation in Small Software Organizations". In: *IEEE Softw.* 17.5 (2000), pp. 82–87. ISSN: 0740-7459.

[36] Craig Larman. *Agile and Iterative Development: A Manager's Guide*. Pearson Education, 2003. ISBN: 0131111558.

[37] Tsun Chow and Dac-Buu Cao. "A survey study of critical success factors in agile software projects". In: *Journal of System and Software* 81.6 (2008), pp. 961–971. ISSN: 0164-1212.

[38] Mike Cohn. *Agile Estimating and Planning*. NJ, USA: Prentice Hall PTR, 2005. ISBN: 0131479415.

[39] VersionOne. *7th Annual Survey: 2012, The State of Agile Development*. Full Data Report. 2013.

[40] B. Nuseibeh and S. Easterbrook. "Requirements engineering: a roadmap". In: *ICSE - Future of SE Track*. 2000, pp. 35–46.

[41] Tore Dybå and Torgeir Dingsøyr. "Empirical studies of agile software development: A systematic review". In: *Information and Software Technology* 50.9-10 (2008), pp. 833–859. ISSN: 0950-5849.

[42] Claes Wohlin et al. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000. ISBN: 0-7923-8682-5.

[43] P. Abrahamsson et al. *Agile software development methods - Review and analysis*. Tech. rep. 478. VTT PUBLICATIONS, 2002.

[44] Scott W. Ambler. "Survey Says: Agile Works in Practice". In: *Dr. Dobb's Journal* 31 (Mar. 2006), pp. 62–64.

[45] Sanjiv Augustine. *Managing Agile Projects*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005. ISBN: 0131240714.

[46] Andrew Begel and Nachiappan Nagappan. "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study". In: *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 255–264. ISBN: 0-7695-2886-4.

[47] T. Chau, F. Maurer, and Grigori Melnik. "Knowledge Sharing: Agile Methods vs. Tayloristic Methods". In: (2003), pp. 302–307.

[48] VersionOne. *5rd Annual Survey: 2010, The State of Agile Development*. Full Data Report. June 2010.

[49] VersionOne. *4rd Annual Survey: 2009, The State of Agile Development*. Full Data Report. June 2009.

[50] VersionOne. *3rd Annual Survey: 2008, The State of Agile Development*. Full Data Report. June 2008.

[51] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990. ISBN: 0-471-92420-2.

[53] Oya I. Tukel, Walter O. Rom, and Sandra Duni Eksioglu. "An investigation of buffer sizing techniques in critical chain scheduling". In: *European Journal of Operational Research* 172.2 (July 2006), pp. 401–416.

[55] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004. ISBN: 0321245628.

[56] Bente Anda et al. "Estimating Software Development Effort Based on Use Cases - Experiences from Industry". In: *4th International Conference on the UML*. Lecture Notes in Computer Science. Springer, 2001, pp. 487–502. ISBN: 3-540-42667-1.

[58] Christoph Schwindt. *Resource Allocation in Project Management*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2005.

[59] G. Ruhe and M.O. Saliu. "The art and science of software release planning". In: *Software, IEEE* 22.6 (Nov.-Dec. 2005), pp. 47–53. ISSN: 0740-7459.

[60] Aybüke Aurum and Claes Wohlin. "The fundamental nature of requirements engineering activities as a decision-making process". In: *Information & Software Technology* 45.14 (2003), pp. 945–954.

[61] Barbara Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger. "Case Studies for Method and Tool Evaluation". In: *IEEE Software* 12.4 (1995), pp. 52–62. ISSN: 0740-7459.

[63] F. Budinsky et al. *Eclipse Modeling Framework*. Addison-WesleyProfessional, 2003.

[64] Nili Guttmann-Beck and Refael Hassin. "Approximation Algorithms for Minimum -Cut". In: *Algorithmica* 27.2 (2000), pp. 198–207.

[65] J.D. Herbsleb and A. Mockus. "An empirical study of speed and communication in globally distributed software development". In: *Software Engineering, IEEE Transactions on* 29.6 (June 2003), pp. 481–494. ISSN: 0098-5589.

[66] Balasubramaniam Ramesh et al. "Can distributed software development be agile?" In: *Commun. ACM* 49.10 (2006), pp. 41–46. ISSN: 0001-0782.

[67] B. W. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs". In: *The Bell system technical journal* 49.1 (1970), pp. 291–307.

[68] P. Fjallstrom. *Algorithms for graph partitioning: A survey*. 1998.

# 6 Additional References

[33] Agile Alliance. *What is Agile Software Development?* Accessed on 27 Feb 2010. 2011. URL: http://www.agilealliance.org.

[52] Multilogic Ltd. *Multilogic Homepage*. 2008. URL: http://www.multilogic.hu.

[54] Mathworks Inc. "Matlab Homepage". In: (2008). Accessed on 28 May 2008.

[57] Microsoft Corp. *Microsoft Office Project 2003 SDK*. Accessed on 4 Sept 2007. 2003.

[62] Eclipse. *Eclipse Homepage*. URL: http://www.eclipse.org.

[69] OptiXware Llc. *OptXware Homepage*. URL: http://www.optxware.com.

[70] Hungarian Post Corp. *Hungarian Post Homepage*. 2008.

[71] Hungarian Tax and Financial Control Administration (APEH). *APEH Homepage*. Accessed on 4 Sept 2008.

[72] Prolan Corp. *Prolan Homepage*. Accessed on 4 Sept 2008. URL: http://www.prolan.hu.

[73] Multilogic Ltd. *Pythia Homepage*. URL: http://www.multilogic.hu/pythia/.

[74] Microsoft Corp. *Microsoft SharePoint 2007*. Accessed on 4 July 2008. 2008. URL: http://www.microsoft.com/sharepoint/.

[75] Budapest University of Technology, Dept. of Measurement Economics, and Intelligent Systems Research Group Information Systems. *ISRG Homepage*.