Complementing Measurements and Real Options Concepts to Support Inter-iteration Decision-Making in Agile Projects

Zornitza Racheva[1], Maya Daneva[1], Luigi Buglione[2]

[1] *University of Twente, Netherlands, {z.racheva, m.daneva}@utwente.nl*
[2]*École de Technologie Supérieure (ETS) / Engineering.IT, luigi.buglione@computer.org*

Abstract. *Agile software projects are characterized by iterative and incremental development, accommodation of changes and active customer participation. The process is driven by creating business value for the client, assuming that the client (i) is aware of it, and (ii) is capable to estimate the business value, associated with the separate features of the system to be implemented.*

This paper is focused on the complementary use of measurement techniques and concepts of real-option-analysis to assist clients in assessing and comparing alternative sets of requirements. Our overall objective is to provide systematic support to clients for the decision-making process on what to implement in each iteration. The design of our approach is justified by using empirical data, published earlier by other authors.

Keywords: agile projects, decision-making process, measurement, business value, ROA, PSU.

1 Introduction

Over the past years, the adoption and adaptation of software measurement techniques by agile software project organizations have visibly increased [3][4][28][30]. Publications on agile software economics indicate the type of information that project decision makers need in order to understand or control aspects of the software product or process. In practice, however, most of the measurement techniques, available to project organizations to deploy for this purpose, were not designed with the agile project context in mind and therefore only partially address this context. This situation is complicated due to the matter that the unique role of clients in agile software processes is not reflected or supported by the existing measurement models. As agile principles suggest [22], a fundamental characteristic of agile development is that the client is responsible for prioritizing the requirements and, thus, is in charge of the project outcome. Moreover, success of agile development effort is commonly linked to the extent to which it helps deliver customer's value [36].

In this paper, we approach the problem of supporting clients' decision-making processes at inter-iteration time. We propose the joint use of real-options-thinking and measurements as a solution. The objective of our solution approach is threefold: first, to provide systematic support to clients in planning for upcoming iterations, second, to enable target levels of business value be set by clients at each inter-iteration point, and third, to make sure clients avoid local optimization in each iteration. Our effort is part of a larger research initiative with the objective to provide midcourse decision support vehicles for the client, for example, when prioritizing requirements. In this paper, we represent the position that viewing decisions as options helps decision-makers to become aware of alternatives, to compare them and, consequently, to make the decision that insures best fit between the business goals and the current software functionality. We propose to use measures of the product and of the process for this purpose.

In what follows, Section 2 reports on related work on measurement and on real-options analysis in agile software development. Section 3 evaluates the fit of the real-options perspective to agile projects from clients' point of view. Section 4 presents our approach to measurement-based decision-making and provides insights gained through already published experiences to substantiate our work. Section 5 concludes with implications and pointers for future research.

2 Related Works

Our work draws on literature sources from three types: (i) agile project management methods, (ii) software measurement models for agile projects, and (iii) real option analysis (ROA) for IT investments. These are described in the subsections that follow.

2.1. Agile software development (ASD) and agile project management (APM)

When speaking about agile methodologies, a needed premise for any researcher is to recognize that, even if they can share the same principles, not all of them are directly comparable in terms of scope and content. A first distinction can be done between Agile Software Development (ASD) and Agile Project Management (APM). To emphasize the main differences, ASD is definable as "an evolutionary approach which is collaborative and self-organizing in nature, producing high-quality systems that meets the changing needs of stakeholders in a cost effective and timely manner" [5], while APM can be defined as "*the work of energizing, empowering and enabling project teams to rapidly and*

978-0-7695-3276-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SEAA.2008.27

reliably deliver customer value by engaging customer and continuously learning and adapting to their changing needs and environments" [9]. The following figure can clarify more such definitions: for the main agile methodologies, Abrahamson et al. [1] summarized the coverage level by software life cycle (SLC) phase and by activity type (project management; process; practices). Just to name a few, XP [10] is an ASD method, while SCRUM [40] is an APM method.

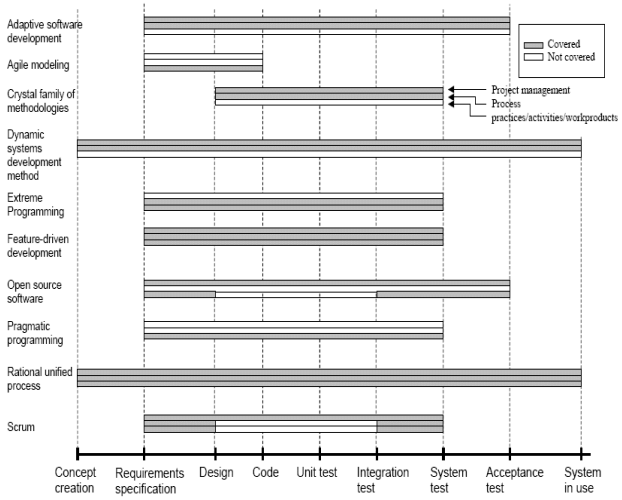

Figure 1. Agile methods by activity/SLC phase coverage

The transition from ASD to APM can be summarized as in Table 1. In the rest of the paper, our attention will be paid to those methods classified as APM ones.

Table 1. Transition to APM [9]

	Agile	APM Transition
Throughput	Flow of Value	Manage the flow, not activities
Teamwork	Small, Integrated Teams	Create an Integrated Team
	Customer Collaboration	Focus on the Project Context, not Content
	Continuous Improvement	More from Lessons Learned to Project Reflections
Leadership	Self-Organization	Coordinate

2.2. Measurement in agile software projects

Currently, both software measurement researchers and practitioners make an effort to modify existing measurement techniques [12], [16] or to create new ones [27], [28], [37] to serve the ASD/APM approach. Based on our review of literature sources (referenced in this section), we can classify the existing measurement approaches in three classes.

The first class includes measurements at code level aiming at quality improvement (also known as internal metrics). Examples are the System Design Instability (SDI) metric [2], the Running Tested Features [25], Knoernschild's code quality and design metrics [27] and Leffingwell's Iteration and Release Retrospectives [29]. Some internal measures are bundled as quality management tools, for example, Kunz et al. [28] describe distinct measurement techniques and their implementation into a measurement tool for quality management, to support refactoring. Ambler [6] recommends Quality Counts.

The second class refers to productivity/effort in each iteration and serves project management purposes. Examples are the Burn-down charts [6], Project Size Unit (PSU) [16], the Function Point Productivity [41], the Drag Factor trend [41], the 'outcome measures' from the XP Evaluation Framework [43] and the COCOMO-style effort model in [12].

The third class includes economic indicators which consider the outcome of the development process in terms of the added value that the product generates. Examples are the results of Rawsthorne [37] who suggests the new measurement technique of Earned Business Value (EBV), and of Elssamadisy [19] who defines rhythms related to delivering value to the client. Both may help the client to make midcourse decisions about the future of the project. Another commonly known indicator is the Break/Even Point [6] which can be used for this purpose as well.

The first class of measurement techniques can give valuable information about the quality of the code, but has the inherent drawback that it is not informative from client's point of view and offers almost no help in making decisions about project development. The economic and productivity measurement techniques can support decision-making process. Nevertheless, they also pose some problems in that they take as input client's estimations or judgements, which rest on the assumption that the client has in advance a clear idea about (i) the business value of each feature that is developed in the project, and (ii) the future business environment and project development. In fact, in an agile project the requirements and their priorities change during the project, thus reflecting current business situation.

Last but not least, the need of historical data for estimations could be a problem for many projects for two reasons: (i) the customer may not have enough experience with IT-projects and can not rely on past estimations; (ii) the developer may not have enough experience in the domain of the project and the past historical data is not applicable or reliable. As Buglione and Abran show, the most popular agile methodologies rely in their sizing procedures on subjective estimations, using completely different, incompatible and relative measures [14].

2.3. Options-based thinking of IT projects

The ROA [18] is first known as a decision support technique in the area of capital investments. The concept of 'real' means adapting mathematical models used to evaluate financial options to more-tangible investments. Since 1999, this concept has found its way into the area of appraising IT investments [8]. The core of the ROA for IT assets consists of: (i) the identification and the assessment of optional components in a project, and (ii) the selection and the application of a mathematical model for valuing financial options that serves to quantify the current value of choosing these components for inclusion at a later time. Optional components are project parts that can either be pushed ahead or pulled out at a later point in time when new information becomes available to the decision-makers. The option, therefore, is the right but not the obligation to spend a budget or put resources on a project. For example, it is possible to first implement a data mart, and then later decide to implement a data warehouse.

3 Options and agile software projects

3.1. Why we think it fits

Erdogmus [20] demonstrates that agile methods are especially appropriate for projects with high uncertainty. This, in turn, implies applying different decision-making approaches like ROA, decision theory [35], or robustness analysis [38]. This paper is solely focused on the use of real-options thinking. We think it is worthwhile exploring the use of ROA-concept as a decision-making vehicle because:

1. Unlike traditional techniques, it comprehends uncertainty and it responds to the dynamics inherent in agile project context.
2. It supports the clients of agile projects in the context of a spectrum of possibilities rather than in the context of a single or three (the best, likely or worst case) discrete set-ups, and it facilitates reprioritization as client's realities unfold over time.
3. It allows incremental expenditures while focusing on the critical pieces of software functionality essential to accomplish the project mission.
4. It rests on the understanding that not all requirements and architecture design options are of equal value.

There is awareness of the use of ROA in the agile software development [21][34][44]. In [21][44], options-thinking has been applied to XP and the authors put forward two XP value propositions, namely, that (i) 'delaying the implementation of a fuzzy feature creates more value than implementing the feature now' and that (ii) 'small investments and frequent releases create more value than large investments and mega-releases' [44, p.13]. Furthermore, [34] describe the options associated with delaying decisions and illustrate their points by using the Microsoft strategy (circa 1988) and [32] has put option thinking at work in XP and Scrum contexts. An important aspect in the analyses by these authors [21][32] is their focus on the viewpoint of the developers. The client's role and decision-making process has received only scamp attention. This motivated us to focus on the clients' perspective. In this sense, our work is complementary to their approach. To the best of our knowledge, applying option thinking in support of clients' midcourse-decision-making processes has not been addressed so far by the research community. In the community of software measurement practitioners, Longstreet [30] found that studying the client happens very rarely, too.

3.2. Definitions of client's options

This paper considers that real-options thinking can be applied two-fold in agile context: from the client's perspective as well as from the developer's one.

First, from client's perspective, real-options thinking can be deployed to prioritize the requirements at the start of each iteration so that the delivery of business value is optimized. Suppose, the business value (BV) for each individual requirement is known to the client, s/he can re-arrange the requirements in sets that form options. Clearly, an option will be worth having when the cost of setting it up is less than its BV (which in our case is the sum of the BVs of all requirements that form the option). The client can, then, compare the advantages of each option and select the one that has the optimal BV. The client can wait to the last responsible moment (as it is called in [34]) to make his decision on the set of requirements to be implemented and this allows her/him the chance to incorporate late breaking information and consider alternative sets of requirements. The term 'responsible' means that the client needs to understand the last point of time to make a decision without affecting the delivery of the project. If bad information comes in it costs the client nothing whereas if good information comes in the client gains value by having the option.

Second, from developers' perspective, the real-options thinking can support the implementation prioritization process. For example, the authors of [32] report on a practice of XP and Scrum developers who defer the decision about which story to develop until just before the coding starts. This allows them to incorporate information that arrives at the last moment, such as a new client request. In fact, the Scrum Backlog provides a forum where any idea for functionality can be recorded without requiring an immediate commitment to build it.

When discussing the 'options' in this paper, we explicitly take the client's viewpoint into account. This is, however, not about applying a new class of mathematical models. Instead, we look at it as a way of re-framing the discussion about client's spending and investment decisions in terms of options. The first step in re-orienting our way of looking at agile projects is to identify the options that exist in software decisions. Then, we will

describe how practitioners can incorporate options thinking into their decision-making processes.

When regarding the agile development method as a sequence of decisions to be made, we treat it as a series of options before or after each iteration. We call '**option**' the set of user requirements to be implemented in each iteration. Here we don't make a difference between functionality, quality of the product or documentation requirements. Each peace of work that the client requires from the developers has an impact on the resources spent (e.g. budget, time), and thus on the outcome of the project. What remains important is to consider a dynamic decision-making process, typically taking part in the beginning of each iteration. The following options could be considered from client's perspective (Table 2):

Table 2. Description of Options

Option	Description
Postpone	*Wait to determine whether to implement certain requirements without imperiling the potential benefits.*
Abandon	*Abandon the project (terminate at the current stage).*
Scope up	*Add new functionality or quality features, not scheduled previously.*
Scope down	*Remove already implemented or negotiated features.*
Switch	*Change or re-arranging the stack with requirements.*

Note that we do not consider the growth option, as it contradicts in its essence with the 'just enough' agile philosophy. However, we think, it is worth investigating the state of the practice in this regard.

3.3. Examples for client's options in agile projects

Furthermore, we looked at literature in agile software engineering to find examples of the types of options as presented in Table 2. Some of the results of this effort are described below:

1. The option of Postponing: the project of ThoughtWorks (an agile coaching firm) and a major US insurance company, working on re-writing a large Java application used in support of core business processes, is a case in point [42]. In this project, the clients structured their business requirements in so-called 'epics' which are a compound story framing a software feature in context of a business scenario. At inter-iteration time, the client went through the release plan's epic list and marked each one as 'Must have', 'Should have', Could have', and 'Won't have (this time around)'. Epics not part of the initial release plan and deemed lower in priority had been deferred until a future release.

2. The option of Abandoning: In many agile projects, the client has the right to cancel at the end of any phase, receiving the working, tested software from all phases completed so far. The experience of a Control System Manufacturer [31] indicates how clients can cancel a project early if they find it is not going as expected and thus loose minimal investment; for example, a project review found that only 20% of the projected business value had been achieved, which was used by the clients to conclude that the project should no longer be pursued.

3. The option of Scoping-up: This is an inherent part of any agile process and the varieties of features or functionality pieces that might be added in any iteration, all depend on the types of stakeholders on the client's side involved. As [7] indicates, operations and support people, architects, regulatory compliance auditors, senior management, all may change their requirements.

4. The option of Scoping-down: Yahoo!'s Mixd project [23] illustrates the use of this option. In their social mobile product experiment, the Yahoo! Advanced Product team cut features and learnt how removing complexity and assumptions in the product allowed people to use it in unintended way. The team prioritized the features by asking if the feature was absolutely necessary to help the users achieve their goal. They brutally cut on those features that diluted the key focus of the product. Dropping those features that they had specified earlier was a major conceptual shift, but turned out to be an easy shift to make, as it eliminated development complexity.

5. The option of Switching: because agile applications are developed in vertical slices instead of horizontal ones, the client never receives 100% of one tier completed before moving to the next one. This lets him/her switch some features and hook them together differently from the original set up. For example, at Sabre Airlines Solutions, clients compared alternative sets of features and switched to 'simplified functionality' at the beginning of each iteration they deemed an alternative set of requirements be at odds with their agile principles [44].

4 Integrating measurements into the inter-iteration decision-making process

Clearly, the delivery of business value for the client is the ultimate goal of an agile project and is an accepted position in the agile community [6][13][20][24][37]. In order to determine the business value of a feature, we build on experts' knowledge in the domain of quantifying business value of information technology. Such methods account for multiple aspects of the IT adoption, as operational, strategic managerial benefits from financial, customers' or process perspectives. Examples of such benefits are: improve process efficiency, reduced cost, increased productivity or better customer-needs satisfaction.

However, to the best of our knowledge of agile literature, there is no published approach of how to deliver, or

maximize business value in a systematic way. One way to go about it is to provide clients at inter-iteration time with accurate and easily available project information that is translatable in business terms, and hence, could support the analysis of the clients' options. We draw on a recommendation by Bowers [13], who stresses the importance of keeping focused on business value. She indicates that the client is supposed to be in control of the product, but has no power over the process. Yet, his role is maintaining at all times the vision about business value. Bowers's recommendation is to give the updates & high quality information for decision making. This is exactly where we want to provide help by offering a systematic decision-making procedure.

When regarding a project from a client's perspective, what we see is a sequence of iterations. Each one can be considered as a mini-project, representing a relatively independent and closed entity. At the beginning, it is a prioritized list with requirements and, at the end, it is working software, with certain features and qualities. We supposethat the goal of a project is to maximize the business value. This motivates the central role of BV for our analysis. As the BV is quantified based on the project work breakdown structure (WBS), we propose the procedure, represented on Figure 2, for the decision-making process. We acknowledge the fact that this procedure could be applied in any project, where a plan-driven project can be theoretically considered as an agile, with only one iteration. However, in our view, the procedure is specifically geared towards iterative projects, as it addresses the uncertainty by using ROA, improves the estimation for each consequent iteration by incorporating experience, and is focused on delivering business value throughout the project.

In the next subsections, we provide our motivation for our choice of measurement techniques along with their calculation rules.

4.1. The Business Value Measurement Technique

For our purposes, we use a part of the Earned Business Value (EBV) technique that was suggested in 2006 as an answer to criticism to agile methods as too developer-centric and providing difficult-to-understand feedback to businesses [37]. EBV is an APM vehicle that measures "how done we are from a business perspective" – namely the percentage of the known business value that is coded up and running. For calculating the BV of the options that the client has, we use the first part of EBV measurement technique, which measures the BV of each feature.

We adapt the method to the agile process in Figure 2 by suggesting that the BV should be calculated at the beginning of each iteration. At that time, the WBS should be composed based on the initial list with requirements for the iteration.

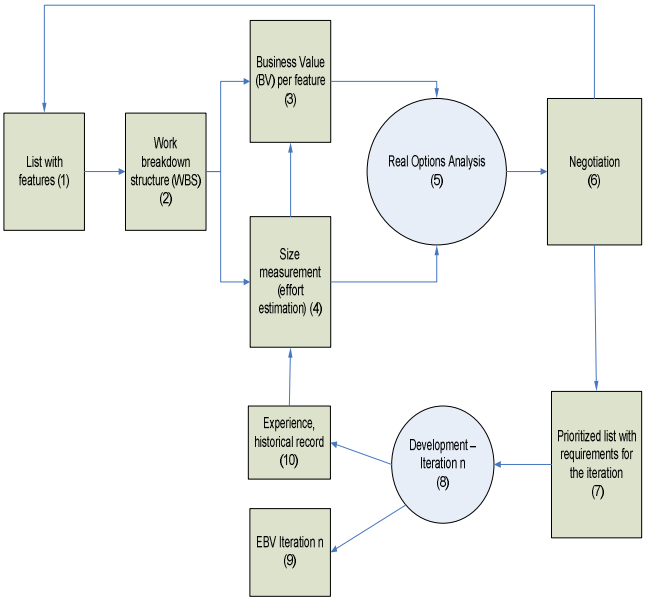

Figure 2. The metric-based decision-making process

We are motivated to use BV because: (i) it is a customer-oriented technique, not technical one; (ii) its calculation uses *stories* as smallest units of value. They are visible from both the inside and outside the project and represent the interface between the client and the developer[1]; (iii) while suggested as a PM tool from developer's side, it can easily be adapted to serve clients' estimations as well; (iv) it uses weights to the WBS legs and stories, that are assigned by the business or client.

The BV technique includes on the following calculation steps: First, the client assigns additive weights to the buckets of the WBS, which represent features and other organizations of stories. Given this assignment, we can now calculate the Business Value (BV) of any WBS bucket, including Stories. The idea is simple and has two parts. First, the value of the whole tree is set 100% (or 1), which means that doing all the work gives all the value. This value (of 1) is assigned to the top of the tree. Second, as we move down the tree, each bucket's value is the appropriate percentage of its parent's value, as compared to its 'siblings' - the other children of its parent. The calculation is recursive, and uses the formula below:

$$BV(bucket) = BV(parent) \times \left[\frac{wt(bucket)}{wt(bucket) + \sum_{siblings} wt(sibling)} \right] \quad (1)$$

Once the BV per story is calculated, we can calculate the BV for sets of stories, that form options, by adding up their BV.

[1] The authors of [24] use the XP concept of story from a functional viewpoint.

4.2. The Project Size Unit (PSU) technique

Agile projects usually adopt iterative effort estimation process, based on experience and analogy, without any standard quantitative size unit. In our approach (Figure 2), we decided to use the PSU sizing technique [16] because of the following reasons: (i) PSU is an 'early' sizing technique, which takes as its input assumptions for estimation at the bid phase, technical proposal, and the initial project WBS. This makes it very suitable for estimation in agile projects, as each new iteration can be considered as a mini-project. (ii) The PSU method is based on sizing units – this means it is a quantitative and explicit measurement. It allows for unified approach independent from concrete agile methodology. (iii) The unit of estimation is a single requirement, as perceived by the client (e.g. what XP calls 'user stories' or FDD 'features'). (iv) It takes into consideration the quality attributes and other non-functional tasks as documentation or PM efforts. Thus, it measures the whole amount of work at project level. (v) Gathering and using historical data can be facilitated. (vi) It uses the WBS as a base for sizing/estimation, which means the same data is used as an input, as for the BV calculation.

While PSU is a recent technique, evidences already indicate its merits:

- PSU has been recognized in Atos Origin Italy in 2003 as a valid size technique in the Sw-CMM ML3 certification process by the lead appraisers;
- two master theses from the University of Alcalà compared PSU with IFPUG FPA and COSMIC in terms of estimation, through the analysis of 33 projects. First results showed that PSU returned a higher R^2 in both cases.
- currently, another PSU application is being performed at a customer site.

The application of the PSU implies the following: each user requirement (the so-called user story) is complemented by additional non-functional part – quality and technical, in an improved version, called US^2 (2^{nd}-generation US) [16][15][2]. They are typically introduced by the developer during the requirements elicitation phase. Further, the developer details how each part of the user stories translates into tasks, including the functional, quality and technical aspects of a story, thus creating an initial draft of the WBS. This has the positive side effect of making the 'hidden' (for the client) effort, explicit. This includes the implementation of quality attributes, as well as technical tasks like *refactoring*. As they can be a significant part of the whole effort, it is of mutual benefit for both the client and the developer to have explicit estimations for them. In the light of the client's perspective that we take in this paper, this quantitative approach can support the client's in making requirement prioritization decisions.

The PSU counting procedure is as follows:

1. The client provides the functional side of US^2.
2. The analyst complements any US^2 with quality and technical information, translating US^2 into tasks and creating a series of mini-WBS, and then submitting a rough estimation to be evaluated by the client.
3. The client evaluates the proposals against a series of previously defined verification criteria. He can approve the stories and go to step 4 or must motivate the request for changes and go back to step 2. This is performed in a discussion while both sides applying and re-adjusting their estimations, till an agreement is reached.
4. The project manager (or Master, in the SCRUM language) aggregates all the verified and validated US^2, creating the initial project WBS, with a split by the proper number of iterations, according to the project constraints.

4.3. Application of ROA

This section presents the last step in our decision-making process, in which we apply ROA.

Suppose preliminary sets of requirements for the project are defined (step 1 on Figure 2). Based on that, the WBS is created (step 2). It serves as a basis for the developing team to estimate the effort (step 4), for example, by using PSU [16] as a sizing technique. Simultaneously BV per each user story (feature) can be estimated (step 3), based on the WBS and the effort estimations. The client, then, prioritizes the list based on max(BV), which is step 5. Based on these two estimates, the scope of the iteration 1 can be defined by the prioritized list with requirements (see step 7 on Figure 2).

This process can be iterative, requiring repetitive estimations and calculations, until mutual agreement is reached in step 6, which is negotiation. The remainder of the project can be considered as a start of a new mini-project – in which, again, the client has to prioritize the user stories, can decide to add new features, change the order of requirements to be implemented, or change quality characteristics. The decision for prioritization is taken based on options-thinking, as described in section 4. As an input at the start of this mini-project (iteration) the developer has to provide the client with information for his decision-making – namely size and effort estimation per feature or per set of features. This process is incremental and dynamic and any changes in the project context can be taken into consideration.

While real-options-thinking fits well with agile context, a number of challenges lie ahead in further developing this approach: (1) customizing ROA for agile decisions requires estimating the costs and benefits associated with the current features and analyzing their interaction; (2) the

[2] There are two types of US^2: Type 1 (at the start of iteration, - or "Sprint" in the SCRUM terminology - typically containing only non-functional tasks needed for the start-up of such block of the project) and Type 2 (containing at least the functional part, during all the other moments in time of the project lifetime).

current mathematical models used in options valuation require collection and analysis of statistical data about agile decisions. Such data may not be easily available for researchers to carry out case studies; (3) agile prioritization has a cultural aspect, so company's culture, or country's culture might favor or limit the application of certain options; so, we would need to understand the role of context when considering viable options; (4) the software architecture for a project needs to be built such that optional components are indeed really optional: it is neither necessary nor impossible to add them at a later moment in time. This means that we need an engineering approach that takes adaptability very seriously; (5) critics say the real-options-based approach is effective only if a company is genuinely prepared to cancel projects after their initial investment. And anybody who has spent any time in the corporate world knows that projects tend to acquire invisible momentum that is hard to stop.

5 Conclusions and Future Research

This paper approached the problem of supporting client's inter-iteration decision-making in agile software projects. We proposed a solution that jointly uses measurements and the concept of real options. Its overall goal is to help clients maximize the earned business value of what is delivered in each iteration. In our approach, we state that the prioritization of requirements and other decisions concerning project evolution can be regarded as a set of options. Our analysis on how to re-think the decision-making processes in agile software project context points out that it is possible to find arguments which, when assembled, call for a new quantitative approach to costs and benefits to be considered when determining when and how much to invest in software functionality. Although the agile community does emphasize the importance of investigating options and their cost/benefits relationships in agile projects, their published works [32], [37], cite isolated "islands" of solutions. Moreover, the options-thinking is mostly applied from developer's perspective and only rarely takes the standpoint of the client.

Our early conclusions from the above analysis is, therefore, that option-based support to client' decision-making is a research problem worthwhile investigating; we also propose to approach this problem in a disciplined way as suggested by authors who applied ROA to other IT areas [11]. We plan as an immediate follow-up action of this research effort the following steps: (i) elaborate on the options and choose a mathematical model for representing the options; (ii) apply the model on a real-life example and assess the results of the model. We plan to complement this research with four activities: (i) surveying different measures of business value for measuring and assessing their applicability in an iterative agile project settings, (ii) identifying information necessary for the client's decision making, (iii) creating a dynamic measurement procedure reflecting the iterative and incremental nature of the agile development process [12], and (iv) exploring the influence of project's context characteristics on the existing options.

Furthermore, while investigating the use of measures of business value we made the observation that business value is assessed based on stories, that is, on functional requirements. The question of how non-functional requirements are expressed as business value remains open. We think, it is worthwhile exploring possible answers to this question and plan to address it in our future research. As indicated in [15], it makes sense to complement story-based value measures with, for example, US^2.

Last but not least, we plan a number of case studies with industrial partners to test our solution approach. We need this in order to evaluate its validity [26], e.g. the generalizability of its results when applied to similar but different agile settings. We admit that different agile software process models have different assumptions and prerequisites, depending on the project context. Therefore, two interesting questions, which we plan to answer as part of our validation case studies, are these: (i) what are the agile project characteristics needed to support the decision-making process in Figure 2, and (ii) how this process needs to be adapted if these context characteristics are not present? Finding answers to these questions can help us move further towards our goal to propose and tailor a set of measurement techniques to a specific project context.

References

[1] Abrahamsson P., Salo O., Ronkainen J. & Warsta J., Agile Software Development Methods. Review and Analysis, VTT Publication # 478, 2002, URL: http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

[2] Alshayeb, M., W. Li, An Empirical Study of System Design Instability Metric and Design Evolution in an Agile Software Process, J of Systems and Software, 74(3), 2005, pp. 269–274

[3] Ambler, S., Agile Adoption Rate Survey : March 2006, Ambysoft website, URL: http://www.ambysoft.com/surveys/agileMarch2006.html

[4] Ambler, S., Agile Adoption Rate Survey : March 2007, Ambysoft website, URL: http://www.ambysoft.com/surveys/agileMarch2007.html

[5] Ambler, S., Agile Master Data Management (MDM), 2007, URL: www.agiledata.org/essays/masterDataManagement.html

[6] Ambler, S., Scaling On-Site Customers, Dr Dobb's Journal, Dec, 2007.

[7] Ambler, S.W., Measure Me Wisely, Dr Dobbs's Journal, July 2005, http://www.ddj.com/architect/184415360

[8] Amram, M., Kulatilaka, N., Real Options: Managing Strategic Investment in an Uncertain World. HBS Press, Cambridge, 1999, ISBN 0875848451

[9] Augustine, S., Managing Agile Projects, Prentice-Hall, 2005, ISBN 0131240714

[10] Beck, K. & Andres, C., Extreme Programming Explained. Embrace Change, 2/e, Addison-Wesley, 2005, ISBN 0-321-27865-8

[11] Benaroch, M, Managing Information Technology Investment Risk: A Real Options Perspective. J of Management Info Systems, 19(2), Fall 2002, pp. 43-84, URL: http://: myweb.whitman.syr.edu/mbenaroc/PAPERS/jmis/jmis-1.pdf
[12] Benediktsson, O., D. Dalcher, K. Reed, M. Woodman, COCOMO-Based Effort Estimation for Iterative and Incremental Software Development, Software Quality Journal, 11(4), 2003, pp. 265-281.
[13] Bowers, A., Leading From A Position Of No Power: A Customer's Perspective of an Agile Team, URL: http:// www.infoq.com/presentations/alexia-bowers-agile-leadership
[14] Buglione L. & Abran A., Improving Estimations in Agile Projects: issues and avenues, Proceedings of the 4th Software Measurement European Forum (SMEF 2007), Rome (Italy), May 9-11 2007, ISBN 9-788870-909425, pp.265-274, URL: http://www.dpo.it/smef2007/papers/day2/212.pdf
[15] Buglione L., Meglio Agili o Veloci? Alcune riflessioni sulle stime nei progetti XP, XPM.it, Febbraio 2007, URL: www.xpm.it
[16] Buglione, L., Project Size Unit (PSU) – Measurement Manual, v1.21, November 2007, http://www.geocities.com/lbu_measure/psu/psu.htm
[17] Cearley, D.W., Fenn, J., Plummer, D.C., Gartner's Positions on the Five Hottest IT Topics and Trends in 2005, Gartner, Publication Id. G00125868, 12 May 2005, URL: www.gartner.com/DisplayDocument?doc_cd=125868
[18] Childs, P. D.; Ott, SH.; Triantis, A.J., Capital Budgeting for Interrelated Projects: A Real Options Approach, J of Financial & Quantitative Analysis, 33(3), 1998, pp 305-335.
[19] Elssamadisy, A., A.Johnson, Rhythms As Agile Diagnostics, Agile Journal, June 2006, http://www.agilejournal.com/index2.php?option=com_content&do_pdf=1&id=55
[20] Erdogmus, H. The Economic Impact of Learning and Flexibility on Process Decisions, IEEE Software, Nov/Dec 2005, pp.76-83
[21] Erdogmus, H., Favaro, J., Keep Your Options Open: Extreme Programming and Economics of Flexibility. In Extreme Programming Perspectives, M. Marchesi et al (eds). Addison-Wesley, 2002, URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.1786
[22] Fowler, M., The New Methodology, December 13 2005, URL: www.martinfowler.com/articles/newMethodology.html
[23] Gatz, S., G. Benefield, Less, Never More: Launching a Product with Critical features and Nothing More, Proceedings of the Agile Conference 2007 (Agile 2007), Washington, DC, USA. IEEE CS, ISBN 0-7695-2872-4, pp. 324-327
[24] Hartmann, D.; Dymond, R., Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value, Proceedings of the Agile Conference 2006 (Agile 2006), Minneapolis, Minnesota, USA. IEEE CS, ISBN 0-7695-2562-8, pp.124-136, URL: http://: /www.berteigconsulting.com/AppropriateAgileMeasurement.pdf
[25] Jeffries, R., A Metric Leading to Agility, in XProgramming, June 2004, http://www.xprogramming.com/xpmag/jatRtsMetric.htm
[26] Kitchenham B., Pickard L., Pfleeger S., Case Studies for Method and Tool Evaluation, IEEE Software, 12 (5), 1995.
[27] Knoernschild, K., Using Metrics To Help Drive Agile Software, in Agile Journal, June 2006, URL:www.agilejournal.com/articles/the-agile-developer/using-metrics-to-help-drive-agile-software.html
[28] Kunz, M.; Schmietendorf, A.; Dumke, R., How to measure Agile Software Development, Proceedings of IWSM/MENSURA 2007 (2nd Int'l Conf. on Software Process & Product Measurement), Palma de Mallorca, Spain, Nov 2007, pp. 319-325
[29] Leffingwell, D., Iteration and Release Retrospectives: The Natural Rhythm for Agile Measurement, Agile Journal, June 2006, URL: www.agilejournal.com/articles/articles/iteration-and-release-retrospectives:-the-natural-rhythm-for-agile-measurement.html
[30] Longstreet, D., Agile Methods and Other Fairy Tales, URL: www.softwaremetrics.com/Agile/Agile%20Paper.pdf
[31] Mahanti, A., Challenges in Enterprise Adoption of Agile Methods – a Survey, J of Computer & Information Technology, 2006, 14(3), 196-207, URL: http://cit.zesoi.fer.hr/downloadPaper.php?paper=752
[32] Matts, C., Maassen, O., "Real Options" Underlie Agile Practices, June 8 2007, URL: http://www.infoq.com/articles/real-options-enhance-agility
[33] Meszaros G., J.Aston, Agile ERP: "You don't know what you've got 'till it's gone!", Proc. of the Agile Conf. 2007 (Agile 2007), Washington, DC, USA. IEEE CS, ISBN 0-7695-2872-4, pp.143-149
[34] Poppendiek, M., T. Poppendiek, Implementing Lean Software Development: from Concept to Cash, Addison-Wesley, 2006, ISBN 0321437381
[35] Pratt, J.W., H. Raiffa, R. Schlaifer, Introduction to Statistical Decision Theory, MIT Press, Cambridge, MA, 1995, ISBN 0585223149
[36] Principles behind the Agile Manifesto, 2001, URL: http://agilemanifesto.org/principles.html
[37] Rawsthorne, D., Calculating Earned Business Value For An Agile Project, Agile Journal, June 2006, URL: http://www.agilejournal.com/articles/articles/calculating-earned-business-value-for-an-agile-project.html
[38] Rosenhead, J., Robustness Analysis: Keeping Your Options Open, In J. Rosenhead et al. (eds.) Rational Analysis for a Problematic World Revisited: problem structuring methods for complexity, uncertainty and conflict, Wiley, Chichester, 2001, pp. 181-207, ISBN 978-0-471-49523-9
[39] Rusk, J., Cutting Scope in not the (Whole) Answer, 2004, URL: www.agilekiwi.com/cutting_scope.htm
[40] Schwaber K., Agile Project Management with SCRUM, Microsoft Press, 2004, ISBN 073561993X
[41] Tengshe, A., Establishing the Agile PMO: Managing Variability across Projects and portfolios, Proc. of the Agile Conf. 2007 (Agile 2007), Washington, DC, USA. IEEE CS, ISBN 0-7695-2872-4, pp. 188-193
[42] Ton H. A Strategy for Balancing Business Value and Story Size, Proceedings of the Agile Conference 2007 (Agile 2007), 13-17 August 2007, Washington, DC, USA. IEEE Computer Society 2007, ISBN 0-7695-2872-4, pp. 279-284
[43] Williams, L., W. Krebs, L. Layman, A. Anton, P. Abrahamsson, Toward a Framework for Evaluating Extreme Programming, Int'l Conf. on Empirical Assessment in Software Eng. (EASE) 2004, Edimburg (UK), ISBN 0-86341-435-4, pp. 11-20, URL: http://agile.csc.ncsu.edu/lmlayma2/papers/WKL04.pdf
[44] Williams, M., Jay Packlick, Rajeev Bellubbi, Scott Coburn, How We Made Onsite Customer Work - An Extreme Success Story, Proc. of the Agile Conf. 2007 (Agile 2007), Washington, DC, USA. IEEE CS, ISBN 0-7695-2872-4, pp. 334-338