

# Allowing for Task Uncertainties and Dependencies in Agile Release Planning

Kevin Logue, Kevin McDaid, Des Greer

## Abstract

*The ability to produce and execute a clear and realistic release plan can be the determinant factor between success and failure of a software project. The creation of such a plan is extremely difficult, key factors such as time and cost to develop chosen functionality and also the likely return are subject to a high level of uncertainty. This paper proposes a relatively simple statistical methodology to account for the uncertainties in the time to develop functionality. The method applies primarily to the eXtreme Programming (XP) method and allows for both uncertainty in story size and in project velocity. In so doing it provides key stakeholders with the opportunity to manage uncertainty in the planning of what functionality to include in upcoming releases. The technique is lightweight in nature and consistent with existing agile planning practices. A case study is provided to illustrate and explore the proposed methodology.*

## 1. Introduction

Release planning is the process of creating a high-level plan that identifies when a releasable version of a software product should be made available and also what functionality should be included. Typically a release plan will look over a period of 2-3 releases into the future. This plan can then be used by an organisation to aid in strategic planning activities such as product promotion or customer training.

The process of creating a release plan can often prove difficult. Different stakeholders may have radically diverse views as to the importance of a particular feature. A strong need exists to understand the technical preference of features and also the ability to balance between required and available effort. The problem is further complicated by the level of uncertainty in the time and cost to develop chosen functionality and in the likely return value.

Software development projects, including those utilising agile methods, have a clear need for reliable release plans. Those that produce regular releases or are under strict contractual obligations are particularly reliant on the quality of their release plans. In these areas failure to supply agreed functionality or to meet a set deadline can be devastating to the company's reputation. This can be particularly problematic for small or start-up companies who may be reliant on a small pool of customers for survival. It is these same companies that are some of the most enthusiastic supporters of agile development methods such as Extreme Programming [2] and SCRUM [5].

One possible approach for release planning is to fix a specific release date and then assign functionality based on how much work can be accomplished within that timeframe. Another approach is to choose the desired functionality and attempt to derive a release date. Regardless of the approach chosen the software organisation must compare the expected value of each requirement against expected cost and time to develop. By not allowing for the uncertainty in these values the organisation increases their chances of running over budget or over schedule.

Building upon work carried out by McDaid, Greer *et al* [1], this paper proposes a method to account for uncertainty in the size of a task and also in the project velocity. When combined with a usable tool, the methodology has the capacity to empower the product manager to explore possible release plans for many different combinations of stories. Furthermore the method can be extended to aid selection of the assignment of stories to iterations within releases. The method has been developed to be lightweight as to better compliment existing agile planning methods. The work is demonstrated through a simple case study and compared to the original model.

The paper is structured as follows. Section 2 introduces agile methods and examines a number of existing release planning approaches. Section 3 explains how this methodology accounts for uncertainty in story size and project velocity, providing our reasoning for the outlined method. Section 4 applies the method to a real problem and assesses the usefulness of the method. This section also compares the model to a closely related model. Section 5 concludes the paper and outlines potential for extending the method to allow for the sequential nature of planning in incremental development in general.

## 2. Agile Release Planning

In Extreme Programming (XP) release planning is carried out through a process called the “Planning Game” [2]. The main focus of the planning game is to determine what can be accomplished by a given deadline and what to do next. During this process the development team decides which user stories to include in the next and future releases. These user stories represent user requirements given as a high level description. Each release is developed over a number of iterations, with each iteration typically in the region of one to two weeks. By carrying development out in such a manner the development team can incorporate feedback from the product owner at the end of each iteration. Through this practice the team is able to implement what they have learned from earlier iterations and implement any changes agreed during consultation with the product owner. Through this process a greater emphasis is placed on steering the project as opposed to specifying a detailed and fixed plan for the development process.

The planning game begins with both development team and product owner identifying all possible user stories. In an ideal development situation the product owner will be an end user of the product or actual project customer. However, more often than not the product owner is a member of the sales or management team who have expert knowledge of the market and customer needs. In the case of small companies, as in our case study, the product owner is likely to be the chief executive officer.

Once all initial user stories have been identified the size of each story is estimated by the development team. The size of user stories can be expressed as either ideal development days or in terms of story points. In this paper we concentrate on ideal development days, also known as perfect engineering days. This gives the number of days it should take a developer to complete the story assuming that they are in a position to engage solely in development activity. Next the development team attempts to estimate the project velocity, how much work can be carried out within an iteration. The project velocity can be found by examining previous iterations taking into account the experience of the development team. Similar planning approaches are used in other agile methods with extensive guidance given [2][3].

With both story size estimates and the expected velocity in hand the product owner then assigns user stories to each release and sets appropriate release dates. This, in effect, requires the customer or product owner to perform a sophisticated cost benefit analysis to decide the viability of features within time and resource constraints.

One simplistic approach to allocating user stories is by estimating the business value, or priority, to each story and to select stories to maximise business value over each release. Usually, highest value requirements are allocated earliest. In event that the release date is set up front, stories can be allocated so as to deliver maximum value by the specific deadlines. Business value is established from business deliverables. Ideally, the product owner and the sponsoring organisation would negotiate this in the presence of the customer.

Whatever approach is used, it is important that an agile method provides the customer with estimates for the time to complete releases that can be achieved with a very high level of certainty. It is also important, particularly for the incremental development methods central to agile processes, that these predictions can be updated as more information becomes available. Such a method is presented in this paper.

A number of methods have been specifically built for an iterative and incremental approach. One such method is EVOLVE [4], which takes into account stakeholder priorities, requirement procedural constraints and also effort limits. On the face of it this method is quite suitable for agile planning, but requires coordination of stakeholders in prioritising requirements and a trust in that judgement. Further, the effort estimation is based on requirements and not explicitly on the constituent tasks that deliver those requirements. Given that user stories are a common vehicle for representing user requirements in agile approaches and that these tend to be at a high level, estimation is more suitable after the tasks involved have been identified.

A similar approach, to that proposed within this paper, has been presented in [12]. The method described used a fuzzy system to account for uncertainty in both expected resources required to implement a task and the overall amount of resources available. This technique recognises the difficulty in providing precise crisp estimates early within the development process. Instead the development team is asked to provide minimum, maximum and most likely values to allow estimates to take the form of fuzzy triangular numbers. Using a probabilistic approach the method compares the required resources with the resources that are available to the project. Unlike traditional Boolean logic this comparison results in the degree to which the resource constraint has been met. In doing so the stakeholders are presented with a series of possible plans with varying degrees of satisfaction and the level to which the resource constraint has been met.

Unlike the method described above our method takes purely a probabilistic approach. A key difference between the two is that Fuzzy Logic measures the degree to which an event occurs while probability measures the chance that an event may occur [13]. Furthermore our method has been designed to compliment the iterative nature of an agile project.

### **3. Handling Uncertainty in Story Size and Project Velocity**

As previously mentioned in this paper, the creation of a release plan is extremely difficult. Developing teams have a tendency to under estimate the size of a particular problem and in so doing decrease their chances of delivering on time and to specification. These broken promises can seriously damage the trust that exists between development team and customer.

The latest set of key practices for Extreme Programming [2] include the practice, termed “Slack” [2][3], in which an amount of time is left unallocated in case the agreed functionality takes longer than anticipated to develop. The key principle is that the team should deliver what is planned through

provision of slack period to allow for overruns. If development proceeds on time then additional stories can be developed. This practice acknowledges that there is a significant amount of uncertainty in the estimated time to complete releases. The method proposed herein is entirely consistent with this practice and provides a mechanism for selecting how much slack time to allow.

Recent attempts to address schedule risk management for projects with a high level of uncertainty are based on the use of schedule and feature buffers [3]. Feature buffering involves the identification of “must have” user stories, representing up to 70% of the planned effort, which are given priority in the release. Other stories are only developed once these priority ones have been completed.

Alternatively, and more commonly, schedule buffers are used. First, it is necessary to quantify the uncertainty through the assignment of a duration range for a user story rather than a single estimate. This, as outlined in [3], can be achieved through the specification of 50% and 90% time points which represent the likely and pessimistic time for completion of user stories. This yields an estimate of the likely variation in the time. If a normal distribution is assumed, than an overall measure of variation for combinations of user stories can be easily calculated. User stories are selected provided the likely pessimistic time for completion of the selected combination of stories is within the proposed timeframe for the release. The sizing of a buffer in this way has been previously suggested in [6][7][8]. However, the normal distribution is not suitable for modelling time to complete a develop task as it is restricted in shape due to its symmetry.

We propose an alternative, but related, methodology to deal with uncertainty in release planning for agile methods. The method provides key stakeholders with the opportunity to manage in a more effective manner the uncertainty in the release planning process. It is not designed to generate a single optimal combination of stories; rather it is concerned with developing a set of story combinations which are optimal or near optimal. The decision makers can then choose from these combinations bearing in mind other considerations. In this way the method guides the decision makers, an approach advocated in many works [9].

The method is compatible with the current planning game in XP and starts with the current practices of gathering all user stories and estimating both the size of each story and also the expected project velocity. Using these estimates the development team is able to determine the number of iterations required to complete each release.

The method recognises that both these values are subject to levels of uncertainty. In the case of story size a number of factors can introduce uncertainty into the estimation such as task difficulty, experience of staff and also the understanding of the desired user story. Similarly, project velocity can change from iteration to iteration resulting in differing amounts of work being carried out between iterations. This can be caused by members of the core development team being required to temporarily leave the project to focus on other activities such as supporting existing software systems. This is particularly true for small organisations that do not have the man power to absorb these occurrences.

Building on existing work we propose to both improve the methods used to represent uncertainty in story size and to propose a mechanism to represent the uncertainty in project velocity over an iteration. These building blocks can then be combined to yield overall uncertainty in the time to develop individual or selected combinations of stories. In this way a set of stories can be chosen for future releases with the confidence that there is a minimal chance that they will not deliver on their commitments.

In summary the method elicits both optimistic, pessimistic and most likely values for story sizes and for project velocity. These are then combined using simulation to yield a distribution for the

variation in the combination of stories selected for releases. The detailed description and associated research for the methodology, we feel, is best presented through a case study conducted with a real company which we label “Company Z”. This is presented in Section 4. The paper is concluded in section 5 with a brief discussion on possible ways to expand the model.

#### 4. Case Study

Company Z (so-called for anonymity) is a small software firm that develops a single very high value product for a small number of very large organisations. They operate on the basis of quarterly releases of their main product. Typically a release would include new functionality driven by the needs of key customers and new functionality designed to attract new customers. They wish to be able to plan two or three releases in the future, selecting from a wide range of possible functionality for their innovative product.

As a relatively young product company with a small number of key important customers they are acutely aware of the need to deliver releases on time with the promised functionality. Of course, as a small company they have to do this within a tightly restricted budget. This calls for reliable release planning.

An agile development process is used but it is found that, while they have a good understanding of the functionality that could be added to the project, they have in the past struggled to provide accurate estimates of the size of stories. This has led to time and cost overruns. These overruns have been exacerbated by the need to perform ongoing maintenance and repair work driven by the needs of their key customers, a practice that impacts on the project velocity through the amount of time that can be spent during iterations on new development.

For these reasons the company need a reliable release planning mechanism that allows for the uncertainty in both the estimation of story size and in the estimation of project velocity given by the amount of development conducted during iterations. Furthermore, to reflect the highly complex decision problem as to which functionality to select, management desires a methodology that allows them to experiment with various combinations of stories across releases. In particular the method should present useful estimates of the time to complete each release.

Presently the method does not automate the optimal selection of stories and release times based on benefit values for the competing stories. This is a very complicated question that requires the manager to collate and compare a large number of disparate and uncertain factors. We return to this issue in Section 5 and for the moment focus on providing useful estimates for the release time given a selection of user stories.

We begin by assuming that the product owner has identified a number of stories for inclusion in upcoming releases. Agile methods would naturally require the development team to produce single value estimates of the story size in story units or in ideal days. These would be developed by first splitting the story in a small number of sub-stories or tasks and aggregating the estimates for each of these.

*Table 1: List of Stories and Tasks*

Story	Tasks
1	1,2,3,4,5,6,7,8
2	1,2,3,4,7,8,9
3	2,7,10,11,12
4	2,3,5,7,12,13
5	14,15,16,17,18,19,20
6	21,22,23,24,25,26,27,28

We propose a similar approach with the proviso that the level of granularity is chosen to allow the team to isolate activities that overlap between stories. However, to allow for uncertainty in the size of stories, the development team must provide three estimates, a most likely, a pessimistic value and an optimistic value, in ideal days for the size of each sub-story.

The stories and constituent sub-stories (called tasks) selected by Company Z are shown in Table 1 with optimistic, most likely and pessimistic values for sub-stories given in Table 2. The common tasks are identified through their numbers. Details of stories are not given for confidentiality reasons. Note that details of 6 of the 10 stories are shown. These involve 28 tasks with significant overlap between stories one, two, three and four. This overlap reflects any dependencies between stories and it is important to break stories down to a level of task detail that allows any dependencies to be modelled.

We use a triangular distribution to model the possible variations in the real size of a task. Likely completion time can be statistically simulated using pessimistic, most likely and optimistic completion times for each task. For example the triangular distribution for task 4 is shown in Figure 1 with the height of the curve at a time point representing the likelihood that the task will take this long to complete in ideal development days. The Triangular probability distribution is well recognised as a suitable distribution when the true distribution of data is unknown. It is conceptually simple and easy to understand, requiring only minimum, most likely and maximum values. Furthermore its lower and upper boundaries are well defined.

*Table 2: Size of tasks in ideal development days*

<b>Task</b>	<b>Optimistic</b>	<b>Most Likely</b>	<b>Pessimistic</b>		<b>Task</b>	<b>Optimistic</b>	<b>Most Likely</b>	<b>Pessimistic</b>
1	0.25	0.5	0.75		15	2	3	6
2	0.5	1.5	2		16	1	2	4
3	0.75	1	1.25		17	2	3	4
4	15	20	30		18	0.25	0.5	1
5	0.2	0.25	0.3		19	1	2	3
6	4	6	10		20	1.75	2	4
7	1.75	2	2.25		21	1.25	1.5	1.75
8	2	3	5		22	1.75	2	2.25
9	4	5	6		23	0.5	1	3
10	1	1.5	2		24	1	1.25	2
11	4	5	6		25	5	6	12
12	2	3	5		26	2	3	5
13	4	5	6		27	4.75	5	6
14	0.75	1	1.25		28	2.75	3	5

An alternative to the Triangular distribution is the Lognormal distribution as originally used in [1] to represent the variation in the size of stories. The Lognormal is a positively skewed distribution, meaning that the distribution has a significant tail on the upper side. Thus it has the capacity to allow for development times to be potentially very high but also to be bounded effectively by 0 on the lower side. As there is no lower or upper bound to this distribution it was assumed (after discussion with domain experts) that there is only a 10% chance that the actual size of the sub-story exceeds the specified pessimistic value. It is also assumed that the chance that the story size is more than the most likely time is the same as the chance it is less than that value. Thus the mode is the same as the median. Using this information, estimates of the two parameters of the distribution can be made.

Using the estimates elicited from experts at the company the method can then simulate the size of a release, in terms of ideal development days, based on the task distributions of each selected story. These sizes are aggregated and the set of aggregated times is used to give the distribution for the overall size. In this way the uncertainty in the size estimates for each individual task generates the overall size uncertainty of each particular release.

To illustrate this, assume that Company Z assigned Stories 1, 2 and 3 to next release. This would involve the completion of tasks 1 to 9 and 11 to 12. We simulate the triangular distributions for each of these tasks and add the results. This gives one possible size in ideal days for the release. Repeating this process over a large number of runs, 10,000 say, results in a distribution for the size of the release.

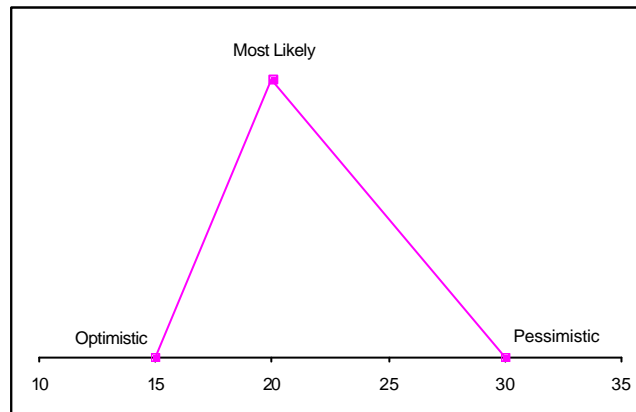


Figure 1: Triangular Distribution for Task 4

The distribution for stories 1, 2 and 3 is shown in Figure 2 using both triangular and lognormal distributions. Although both distributions return similar averages of 51.5 and 51.9 respectively, it is evident that the level of variance contained within the lognormal distribution is considerably higher.

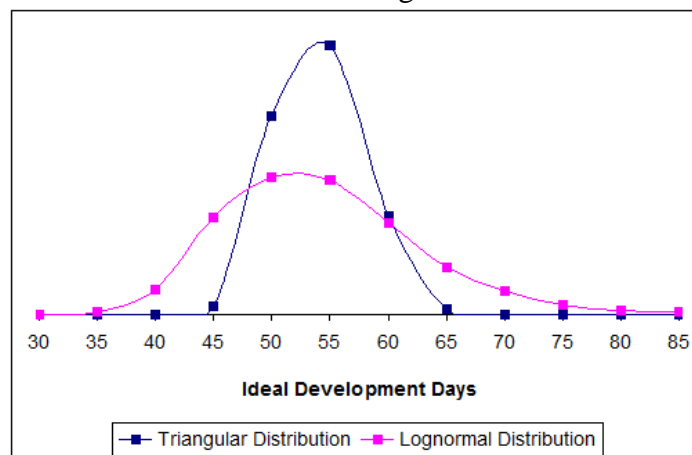


Figure 2: Ideal development days to complete stories 1, 2 and 3

As previously mentioned no upper or lower boundaries are defined for the lognormal distribution resulting in a greater range to that of the triangular distribution, 35 to 85 days compared to 45 to 65 days.

Given the size of the combined stories in ideal days it is still unclear how long the release will take to develop in real or calendar days. Existing agile practices calculate the duration of a release by dividing the size of the release by the expected project velocity. As previously discussed in this paper project velocity is often uncertain. This is particularly true for Company Z where the small

team is often called to support existing releases. As such their project velocity values are unavoidably prone to high levels of uncertainty.

To simulate the uncertainty in project velocity we choose the well understood normal distribution. The parameters of this model, mean and standard deviation, could be specified by the development team or calculated by examining previous project velocities. In the current model these project velocity values are assumed to be independent in that a high or low velocity does not mean the next iteration will witness the same fluctuations. However, the sequential nature of agile development [3] makes it possible to recalculate project velocity in the event that project velocity values are found to be consistently inaccurate.

Company Z, employing 2 full time developers, has estimated that the project velocity mean is 6 ideal days per iteration with a standard deviation of 1 day, where an iteration is a one week period. Taking this into account we can now produce a distribution for the likely duration for developing stories 1, 2 and 3 in the upcoming release. The distribution for the real development time for these stories is shown in Figure 3. The average time is found to be 43 days or 8.6 iterations or weeks.

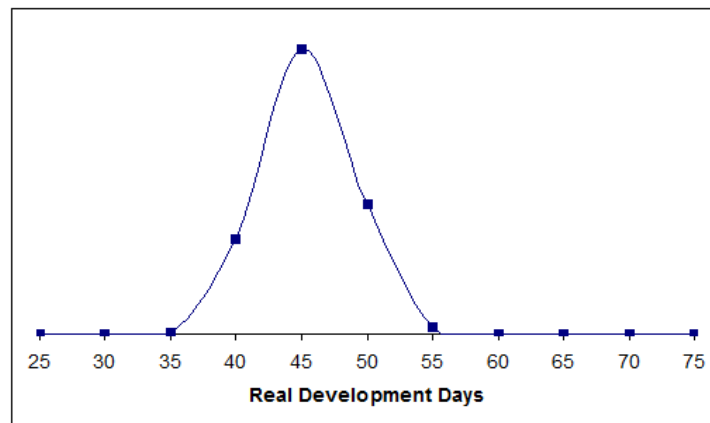


Figure 3: Calendar time to develop stories 1, 2 and 3

The goal of the method is not to produce a single optimal solution. Instead we recommend that a range of data be provided to support rather than dictate decision making [9]. To that end, the best use of the simulated data is to provide management with the likelihood of completing the release by the end of each iteration. Figure 4 shows, through a cumulative distribution, the likelihood that the release is completed by the end of each iteration where the triangular distribution is used.

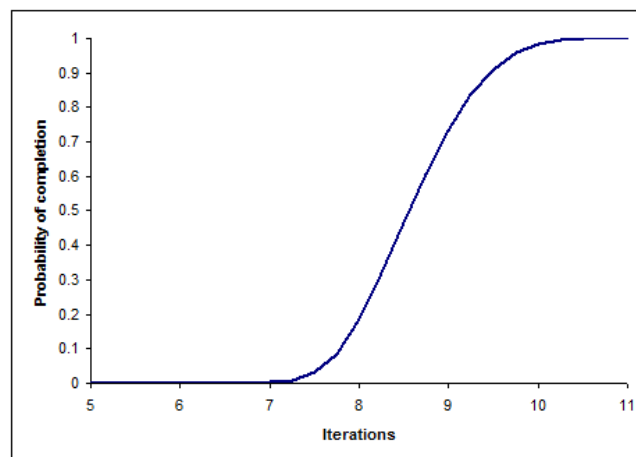


Figure 4: Likelihood that release is completed by the end of each iteration

Table 3 shows these values for the iterations from 7 to 11. While the average time to complete is 8.6 iterations, the table shows that, to be 90% certain of completing the desired functionality, management would have to plan a release date 10 iterations into the future. The additional 1.4 week period represents the slack time that should be included in the plan to ensure that the release is completed on time. Further slack can be added based upon the project manager's discretion. Of course, additional stories can be added as stories are completed on time.

*Table 3: Likelihood of release completion by iteration*

<b>Iteration</b>	7	8	9	10	11
<b>Probability</b>	0.24%	18.66%	73.46%	98.58%	100.00%

## 5. Conclusion

To this point the research explains a mechanism that allows customers or product owners to select combinations of stories and to choose resulting release times with the confidence that the releases will be completed on time. The method proposed is consistent with the best XP practice of scheduling slack to ensure selected functionality is delivered on time. While the technique has been automated through spreadsheet technology work will shortly commence to implement the tool in a customised application.

The initial task of selecting user stories can be extremely difficult and is an area that requires further research. As outlined earlier in the paper the business value for implementing specific functionality is also susceptible to high levels of uncertainty and this is particularly true for product companies. Any tool designed to help with the selection of user stories should acknowledge this uncertainty. Furthermore it is the belief of the authors that is not sufficient to solely base this selection on business value. Instead we propose the use of a return on investment (ROI) approach that takes both business value and also cost to implement into account.

The methodology presented in this paper can be extended as follows to allow for these added complications. First a distribution for the business value of each story would be developed, again through elicitation of an optimistic, expected and pessimistic value. Following this a distribution for the ROI of a story, incorporating the uncertainty, is found by simulating from both the business value and size distributions and dividing one by the other. An overall ROI for combinations of stories would follow.

The issue of interdependencies between tasks is another area that requires further attention, particularly in light of the sequential nature of release planning. While the model in its current form accommodates dependencies through shared tasks, it fails to provide guidance as to how times to complete one task may influence the time to complete another task. This relationship could be due, amongst a large number of reasons, to the fact that the same developer is implementing both tasks or that the tasks both require a programming language that the team have little familiarity with. Existing methodologies [3] suggest that in the event that estimations in a completed iteration are found to be too low, all estimation of remaining stories should be scaled accordingly. We feel a more subtle approach is required where estimates are scaled according to the development times of related stories completed in previous iterations. We feel that this is a key research question for iterative and incremental decision making, but acknowledge that there is a danger that the elicitation of the extra information required from the development team may not be consistent with an agile process.

The model in its current form is aimed towards an agile environment, however we believe that it would also be applicable to more plan orientated IID approaches such as the Rational Unified Process® (RUP®) [10].

Planning within RUP focuses on the identification of the tasks required to achieve a certain objective. Once these tasks are known the development team estimate the size of a task using the function point metric. As with agile methodologies the quality of estimates used in the creation of a plan are key to its reliability. Current RUP practices suggest the use of Wideband Modified Delphi [11] to help developers in generating these estimates, a process quite similar to Extreme Programming's planning game. This involves eliciting estimates from each participant and thus would naturally lend itself to estimating variability or uncertainty in the development times for tasks. This could be framed within a more formal method such as the one proposed within this paper.

## 6. Acknowledgement

This research is supported by the Irish Technological Sector Research initiative under the Post-Graduate R&D Skills Programme.

## 7. References

- [1] McDaid, K., Greer, D. , Keenan, F., Prior, P., Taylor, P., Coleman, G., Managing Uncertainty in Agile Release Planning, Proc. 18th Int. Conference on Software Engineering and Knowledge Engineering (SEKE'06), pp 138-143, 2006.
- [2] Beck, K., Andres, C., "Extreme Programming Explained", Addison Wesley, Reading, MA, 2001.
- [3] Cohn, M., "Agile Estimating and Planning, Prentice Hall", 2005.
- [4] Greer, D. & Ruhe, G., Software Release Planning: An Evolutionary and Iterative Approach, J. Information and Software Technology, vol. 46, issue 4, pp 243-253, 2004.
- [5] Schwaber, K. and Beedle, M, "Agile Software Development", Prentice-Hall, 2002.
- [6] Reinertsen, D.G., "Managing the Design Factory: A Product Developer's Toolkit. Free Press", 1997.
- [7] Newbold, R.C., "Project Management in the Fast Lane: Applying the Theory of Constraints", St. Lucie Press, 1998.
- [8] Leach, L.P., "Critical Chain Project Management", Artech House, 2000.
- [9] Ruhe, G., Saliu, O., "The Art and Science of Software Relapse Planning", IEEE Software, issue 6, vol. 22, no. 6, pp 47-53, 2005
- [10] Rational Software, [www.rational.com/](http://www.rational.com/)
- [11] Kroll, P., Kruchten, P., "The Rational Unified Process Made Easy – A Practitioner's Guide to the RUP", Addison-Wesley, 2005.
- [12] Ngo-The, A., Ruhe, G., Shen, W., "Release Planning under Fuzzy Effort Constraints," icci, pp. 168-175, Third IEEE International Conference on Cognitive Informatics (ICCI'04), 2004.
- [13] Steeb, W., "Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithms, Gene Expression Programming, Wavelets, Fuzzy Logic - with C++, Java and Symbolic++ Programs", World Scientific Publishing Co., Inc. 2003