

# Managing Uncertainty in Agile Release Planning

K. McDaid<sup>1</sup>, D. Greer<sup>2</sup>, F. Keenan<sup>1</sup>, P. Prior<sup>1</sup>, P. Taylor<sup>2</sup>, G. Coleman<sup>1</sup>

*1. Dundalk Institute of Technology, Dundalk, Co Louth, Ireland*

*{kevin.mcdaid, frank.keenan, paul.prior}@dkit.ie*

*2. Queens University Belfast, Belfast, BT7 1NN*

*{des.greer, p.taylor}@qub.ac.uk*

## Abstract

A clear and realistic release plan is central to the strategic planning activities of the firm developing the software. This paper supports existing agile methods by developing a novel but relatively simple statistical methodology to predict the real time to develop selected functionality. In so doing it provides the product owner with a decision support mechanism to determine the likelihood of completing releases on time for any combination of stories. In this way it is consistent with the best Extreme Programming (XP) practice of selecting stories of two types for a quarterly release, ones that are key and must be delivered and ones that are considered as “Slack” and that can be developed if time permits. A case study is used to explain the proposed methodology.

## 1. Introduction

Release planning is the process of deciding when in the future a releasable software product should be available and what functionality should be included in each released version of the software product. It refers to the creation of a high-level plan that covers a period of typically up to three or four releases into the future. Crucially, it guides the strategic planning activities of the organization. Product companies producing regular releases to satisfy key customers are particularly reliant on release planning.

One approach to release planning is to fix a set of release dates and then determine how much functionality can be achieved by those dates. Alternatively, one could start by choosing the functionality and then derive the release dates. Irrespective of the approach taken, the software organization must assess the value of the functionality against the cost and time to develop the system. Unfortunately, these can be highly uncertain characteristics, and any framework must acknowledge this truth.

Software development projects, including those utilizing agile methods, have a clear need for reliable release plans. Projects with contracts and those that must meet a firm deadline and include a reasonably firm set of functionality are examples that come with greater consequences of being wrong. Similarly, projects that are planned far in advance or ones for which requirements are not well understood involve a significant amount of uncertainty which should be allowed for so that missed deadlines that will impact on a company’s reputation are to be avoided. Small and newly established companies are acutely aware of the need to protect their reputations by delivering the functionality promised by the dates indicated. These same companies are the keenest proponents of agile development methods such as Extreme Programming [2] and SCRUM [9]

Building on current planning practices in agile methods this paper proposes a new mechanism to improve the release planning for such companies by allowing for the uncertainty in the time to complete the development work specified in the release plan.

The paper is structured as follows. Section 2 introduces agile methods and the current approaches to release planning. Section 3 explains the background and the rationale for the proposed methodology. A case study in Section 4 applies the methodology to a real problem and assesses its usefulness to the small company under examination. Section 5 concludes the paper and includes a short discussion as to how the business value of the functionality can be included in the methodology.

## 2. Release Planning For Agile Methods

Agile supporters have tackled the problem of release planning. In Extreme Programming this was termed the “Planning Game” [2]. This activity refers to the decision process as to which user stories to include in the next and future releases, where ‘user stories’ are a high level description of user requirements. A release is normally

developed over a series of iterations, typically of one or two week duration, with feedback obtained from the product owner at the end of some or all of the iterations. It is important to note that release planning does not include detailed iteration planning that would indicate, amongst other things, which developers will work on which stories or plans.

The planning game starts with the listing of all potential stories identified by both the product owner and the development team. For a bespoke project the product owner will naturally be the customer. However, in the case of product development, the product owner can be a member of the sales or management team who has expert knowledge of the market and the needs of current and potential customers. In the case of small companies, as in our case study, the product owner is more likely than not to be the chief executive officer.

Once the stories have been identified the size of each story is estimated by the development team. This size can be expressed in terms of ideal development days or in terms of story points. In this paper we concentrate on ideal development days, also known as perfect engineering days. It gives the number of days it would take a developer to complete the story assuming that they are in a position to engage solely in development activity. To indicate the amount of development work possible in an iteration, the team provide the product owner with the team velocity. This is often based on previous projects and, in effect, gives the number of ideal development days that the team will manage on average over the period of an iteration. Similar planning approaches are used in other agile methods with extensive guidance given in [2] and [3].

Based on this information the product owner must then select user stories for each release and the appropriate release dates. These key decisions involve the prioritization of features for inclusion in releases and iterations. This, in effect, requires the customer or product owner, customer, to perform a sophisticated cost benefit analysis to decide the viability of features within release and budget constraints.

One simplistic approach is to assign a business value, or priority, to each story and to select stories to maximize value over time. Typically highest value cards are selected earliest and if the release date is set up front, cards are added so as to deliver maximum value by then. Business value is established from business deliverables. Ideally, the product owner and the sponsoring organization 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 uncertainty. allow for the inherent uncertainty. It is also important, particularly for the incremental development methods key to agile

processes, that these predictions can be updated as more information becomes available. Such a method is presented in this paper.

Other relevant approaches include those on requirements prioritization, reviewed in [5]. The methods could be adjusted to suit IID or even agile methods, but the effort required becomes excessive as the number of requirements rises. Several methods have been specifically built for an IID approach, such as those reviewed in [8]. One such method is EVOLVE [4],[7] which takes into account priorities from stakeholders, dependencies between requirements and effort limits for increments. On the face of it this method is quite suitable for agile planning, but requires coordination of stakeholders in prioritizing 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.

### **3. An Improved Methodology for Managing Uncertainty**

The need for honest release plans within agile methods that acknowledge the actual time it takes to complete tasks, and the inherent variation in these estimates, is well-understood. Without these, the developing team has the tendency to over-commit and consequently under-deliver. These broken promises lead to reduced morale and to distrust between developers and the product owner. These considerations are to the forefront of the minds of today's customers chastened by accounts of cost and time overruns in the development of software systems.

This is highlighted in the latest set of key practices for Extreme Programming [10] which includes the practice, termed "Slack", of only signing up to for what the team is confident of achieving. Within this approach it is always possible to add more stories, time permitting, thus delivering more than was actually promised. This practice acknowledges that there is a significant amount of uncertainty in the estimated time to complete releases. Naturally the question arises as to how much functionality should be scheduled for each release.

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 [12], [13] & [14].

We propose an alternative, but related, methodology to deal with uncertainty in release planning for agile methods. The novel approach in our view better reflects the sources of uncertainty and can thus yields more accurate and useful results. When combined with a usable tool, the methodology has the capacity to empower the product manager to explore many combinations of stories. In so doing the research supports the product manager in his/her key role in the selection of requirements for future releases. Furthermore the method can be extended to aid selection of the assignment of stories to iterations within releases.

The methods starts with the current practice for estimating the real time it will take a development team to complete a story or set of stories. As explained earlier this requires, for a single story, two distinct estimates as follows:

1. Estimation of the “size” of a story as represented by ideal days to develop the story in question.
2. Estimation of the likely project velocity represented by the ideal development days that will be completed by the team over an iteration.

These are combined to find the number of iterations required to complete the story.

Naturally, both of these estimates are subject to uncertainty. There is a variety of factors that can affect the level of this uncertainty. In the case of estimates of story size, some of the factors at play are the complexity of the story, the level of understanding of the requirement and experience of the development team. Similarly, project velocity can change from iteration to iteration during which differing amounts of developers time is spent on activities other than development such as team meetings or communication with customers and managers. This uncertainty can be even more pronounced in the case of small software product companies who may have to drop all development work during an iteration to concentrate on sales activities or on maintenance and support actions for key existing customers.

To date work [3], as outlined earlier in this section, has focused only on the uncertainty associated with estimation of story size. 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.

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. Section 5 suggests how the method can be expanded to allow for the uncertainty in the business value of stories.

#### 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 organizations. 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.

This goal is complicated by the uncertainty they face in the planning of projects. They use an agile development process but find 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 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 accurate 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.

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 two estimates, a most likely and a pessimistic value, in ideal days for the size of each sub-story. It is 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.

**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

The stories and sub-stories (called tasks) selected by Company Z together with most likely and pessimistic values for sub-stories are given in Table 1. 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.

We use a lognormal distribution to model the variation possible in the real size of a task or sub-story using ideal development days as the measure. This is based on work in [15] which shows that the actual time

taken to develop a piece of functionality relative to the estimated time follows a lognormal distribution. The analysis is based on extensive data some of which is provided in [17]. The parameters of the distribution for each task can, through standard statistical methods, be estimated from the most likely and pessimistic values. For example the lognormal curve for task 6 is shown in Figure 1 with the height of the curve at a time point representing the probability that the task will take this length of time to complete in ideal development days. A lognormal distribution is chosen as it reflects the tendency of a development teams to significantly underestimate the size of stories. This is in contrast with the symmetrical shape of the normal distribution often used to represent ideal completion times. A further alternative is the beta distribution but this requires more data and assumes an upper bound on the completion time which is difficult to specify.

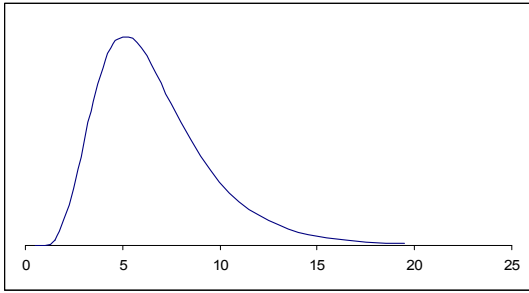
**Table 2: Size of tasks in ideal development days**

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

Once the two estimates are provided for each task, lognormal parameter values for each task are then determined that specify the probability distribution for the size of the task. The distribution for the size, in terms of ideal development days, of a release based on the selected stories is then found by simulating from each of the task distributions that make up the selected stories. 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 the individual tasks generates the uncertainty in the overall size of the functionality selected for a release.

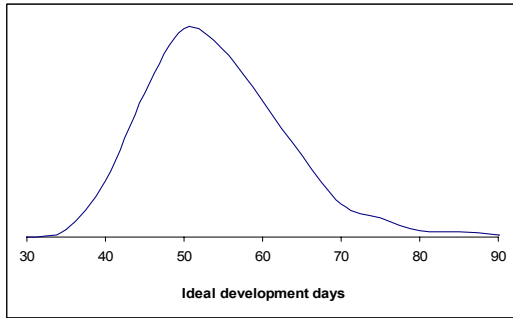
To illustrate through an example, assume that Company Z wish to develop Stories 1, 2 and 5 during the next release. This would involve the completion of tasks 1 to 9 and 14 to 20. We simulate from the lognormal distributions for these tasks and add the

results. This gives one possible size in ideal days for the release. Repeating this process a large number of times, 10,000 say, results in a distribution for the size of the release.



**Figure 1: Lognormal distribution for size of task 6**

This distribution for stories 1, 2 and 5 is shown in Figure 2. Although the average size the release is 55 ideal development days the curve shows that there is an appreciable chance that the size could exceed 80 days. The curve indicates that there is almost no chance that the size will be less than 30 days or more than 90 days.



**Figure 2: Distribution of Ideal time to complete stories 1, 2 and 5**

Given the size of the combined stories in ideal days, it is still unclear as to how many calendar days or weeks it will take to develop this functionality. Agile methods calculate the likely duration of the release by dividing the size of the release by the project velocity, expressed as the number of ideal development days work completed in an iteration [3].

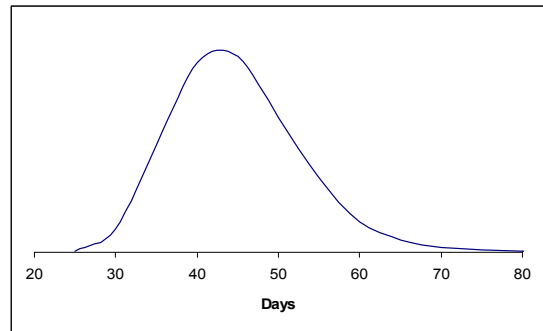
There is natural variation from iteration to iteration in the amount of time the team spends on development work. This is particularly true for Company Z where the small team is often called to deal with urgent maintenance work for existing releases. This leads to significant uncertainty in the project velocity.

In contrast to other research, the methodology proposed here allows for this variation in the project velocity. Specifically, we propose that the project velocity can vary from one iteration to the next based on a normal distribution with a mean and standard deviation. These could be specified by the development team and management or calculated from previous project velocity values if these are available and

considered to be representative of what may happen in the upcoming release.

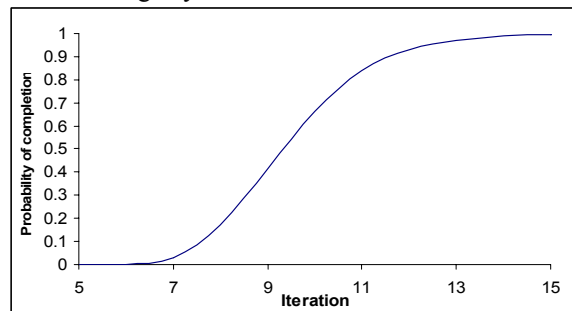
We make the assumption that these project velocity values are independent in that one low or high value does not mean the following iteration will also result in a low or high value. It would not be overly difficult to include a step in the methodology to allow for this dependency, likely to be relatively weak. However the need to keep the method as lightweight as possible [1] overrides this consideration.

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 5 in the upcoming release. The distribution for the calendar time to develop these stories is shown in Figure 3 with a range from 20 to 80 days. The average time is found to be 46 days or 9.2 weeks or iterations.



**Figure 3: Calendar time to develop stories 1, 2 and 5.**

Management need to know the likelihood that the release will be completed by the end of each iteration. Figure 4 shows, through a cumulative distribution, the probability that the release is completed by the end of each 5 working day iteration.



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

Table 3 shows these values for the iterations from 8 to 13. While the average time to complete is 9.2 iterations, the table shows that, to be 90% certain of completing the desired functionality, management would have to plan a release date 12 iterations into the future. The additional three week period represents the slack

time that should be included in the plan to ensure that the release is completed on time. Of course, additional stories can be added as stories are completed on time. The question at to when the decision to add additional functionality can be taken is an interesting one currently under investigation.

**Table 3:Probability of release completion by iteration**

Iteration	8	9	10	11	12	13
Probability	.24	.47.	.71	.86	.95	.98

## 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 releases will be completed on time.

The important question remains however as to which stories should be selected. As outlined earlier this is a complicated decision based on the business value of the proposed stories with high value stories selected first. Usually, and certainly in the case of product companies, the business value of a story is a highly uncertain measurement, and any mechanism for selecting stories must acknowledge this uncertainty. Also, the selection of features should be based on a return on investment (ROI) measure allowing for the cost of development of stories.

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.

The authors of this paper propose a lightweight methodology for the planning of releases that reflects the main sources of uncertainty in the estimation of development time for companies following an agile process. The approach is illustrated through a case study.

The method is consistent with the Extreme Programming practice of selecting a conservative set of stories for development in a release to ensure that the firm delivers on their promises. This is achieved through scheduling of a slack period and this paper explains how this can be specified with confidence.

Although the case study looks at a single release, the approach can be applied to any number of releases into the future. Furthermore, the method can be reapplied, even at the end of each iteration, to allow for progress to that point. In this way a reliable release plan can be developed on an ongoing basis.

There are a number of assumptions with the model that may require further consideration. Specifically, the authors assume that any errors in the estimation of stories are independent. It is our belief that this may not be true as teams that understate the size of one story may also underestimate the size of others. While it is straightforward to deal with this complication in the model, the strength and nature of this correlation has not yet been established in the literature. Most importantly this correlation would lead to a higher level of uncertainty. There may also be a weak correlation between the project velocity values from one iteration to the next as support and maintenance activities started in one iteration can carry over to the next. This again would lead to greater uncertainty in overall estimates and could be modelled using a time series approach. That said, the importance of keeping the process as lightweight as possible is well-understood.

## REFERENCES

- [1] Agile Manifesto, [www.agilemanifesto.org](http://www.agilemanifesto.org).
- [2] Beck, K., *Extreme Programming Explained*, Addison Wesley, Reading, MA, 2001.
- [3] Cohn, M., *Agile Estimating and Planning*, Prentice Hall, 2004.
- [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] Karlsson, J, Wohlin, C and Regnell, B., *An evaluation of methods for prioritizing Software Requirements*, *Information and Software Technology*, 39 (1998), pp 939-947.
- [6] Larman, C., Basili, V. R.: *Iterative and Incremental Development: A Brief History*. *IEEE Computer*, Vol. 36(6), pp. 47 – 56, 2003.
- [7] Ruhe, G. & Greer, D. *Quantitative Studies in Software Release Planning under Risk and Resource Constraints*, *Proceedings of the IEEE-ACM International Symposium on Empirical Software Engineering*, 262-271, 2003.
- [8] Saliu, O., Ruhe, G., *Software Release Planning for Evolving Systems*. *Innovations in Systems and Software Engineering*, Vol 1 (2005), Issue 2, pp 189-204.
- [9] Schwaber, K. and Beedle, M, *Agile Software Development*, Prentice-Hall, 2002..
- [10] Beck, K. & Andres, C., *Extreme Programming Explained* (2ed), Addison Wesley, Reading, MA, 2004.
- [11] DeMarco, T., *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency*, Broadway Books, New York, NY, 2001.
- [12] Reinertsen, D.G., *Managing the Design Factory: A Product Developer's Toolkit*. Free Press, 1997.
- [13] Newbold, R.C., *Project Management in the Fast Lane: Applying the Theory of Constraints*, St. Lucie Press, 1998.
- [14] Leach, L.P., *Critical Chain Project Management*, Artech House, 2000.
- [15] Little, T., *Agility, Uncertainty and Software Project Estimation*, <http://www.agilealliance.com>
- [16] <http://www.agilealliance.com/articles/littletoddagilityunce>, accessed 1 March 2006.
- [17] DeMarco, T., *Controlling Software Projects*, Prentice-Hall, 1982

