Cite this paper as

Walworth, T., Yearworth, M., Shrieves, L., & Sillitto, H. (2016). Estimating Project Performance through a System Dynamics Learning Model. *Systems Engineering, 19*(4), pp.334-350. doi: 10.1002/sys.21349

Estimating project performance through a system dynamics learning model

Abstract

Monitoring of the technical progression of projects is highly difficulty, especially for complex projects where the current state may be obscured by the use of traditional project metrics. Late detection of technical problems leads to high resolution costs and delayed delivery of projects. To counter this, we report on the development of a new technical metrics process designed to help ensure the on-time delivery, to both cost and schedule, of high quality products by a UK Systems Engineering Company. Published best practice suggests the necessity of using planned parameter profiles crafted to support technical metrics; but these have proven difficult to create due to the variance in project types and noise within individual project systems. This paper presents research findings relevant to the creation of a model to help set valid planned parameter profiles for a diverse range of system engineering products; and in establishing how to help project users get meaningful use out of these planned parameter profiles. We present a solution using a System Dynamics model capable of generating suitable planned parameter profiles. The final validated and verified model overlays the idea of a learning 'S-curve' abstraction onto a rework cycle system archetype. Once applied in System Dynamics this matched the mental models of experienced managers within the company, and triangulates with validated empirical data from within the literature. This has delivered three key benefits in practice: the development of a heuristic for understanding the work flow within projects, as a result of the interaction between a project learning system and defect discovery; the ability to produce morphologically accurate performance baselines for metrics; and an approach for enabling teams to generate benefit from the model via the use of Problem Structuring Methodology.

Keywords

Technical metrics; Planned parameter profiles; Rework; Action research; System dynamics;

1 Introduction

This paper describes part of a larger effort to deploy effective systems engineering metrics into Thales UK, a complex engineering organisation delivering a diverse range of systems engineering products. Thales UK covers several business domains and grew through a series of acquisitions. Each of the legacy companies had its own organisation, culture, and process maturity. The complexity of the overall problem, and an identified need for methods that encourage participation and shared learning, meant this research used approaches based around the notion of problem structuring, via an approach inspired by Soft Systems Methodology (SSM) (Checkland 2000). This has been structured around a series of cases studies, of which this paper forms a single study. The presentational style of this paper is determined by this research method, motivated by recent desire for the presentation of more informative narrative case studies by Omerod, inspired by the earlier work of Pickering (Ormerod 2014; Pickering 1995; Pickering 1993). We have drawn on methodological guidance about case studies from Yin (2009), and previous work in research methods for information systems (Lee & Hubona 2009; Baskerville & Wood-Harper 1996).

Thales has experienced difficulty in tracking the technical maturity of projects. This is partly due to the complex nature of the products under development (Sheard & Mostashari 2009). Complex project behaviour is very hard to explore using a simple response metric based on historical data according to arguments presented by Kurtz and Snowden (2003), who suggest that any reliance on historical data will insufficiently prepare projects for future development. Project management metrics traditionally measure historical performance to schedule and cost, an approach that has two key drawbacks: metrics that are lagging in nature; and therefore metrics that are unable to show the incremental design maturity required (Sillitto 2004; Frenz 2005; Walworth et al. 2013). This can result in technical performance shortfall, often characterized by late awareness and little warning of impending problems. This in turn leads to high resolution-costs and delayed delivery (Sheard & Mostashari 2013). To counter this, Thales has emphasised the use of technical metrics to track project progression. The value of these metrics for complex systems engineering projects is highlighted by Elm (2008; 2012) who shows a very strong correlation between projects with good Systems Engineering monitoring and control activities, and projects with high levels of success. The metrics used in Thales were selected based on industry best practice to provide a range of technical maturity perspectives (CMMI Product Team 2001; Sillitto 2004; INCOSE Measurement Working Group 2010). One of the metrics introduced as part of the technical metrics process was the Requirement Status Metric (RSM). The RSM is designed to track the progression of requirements from ‘new’ to ‘sold-off to customer’ across the project lifecycle. It is a stacked bar graph, and when requirement states change, the new state “eats up” the old state from the bottom of the graph. The RSM can indicate overall requirement numbers, the state of all requirements at each reporting period and, most importantly, the progression trends of requirement movement. A generic shape for the Requirement Status Metric is shown in Figure 1.

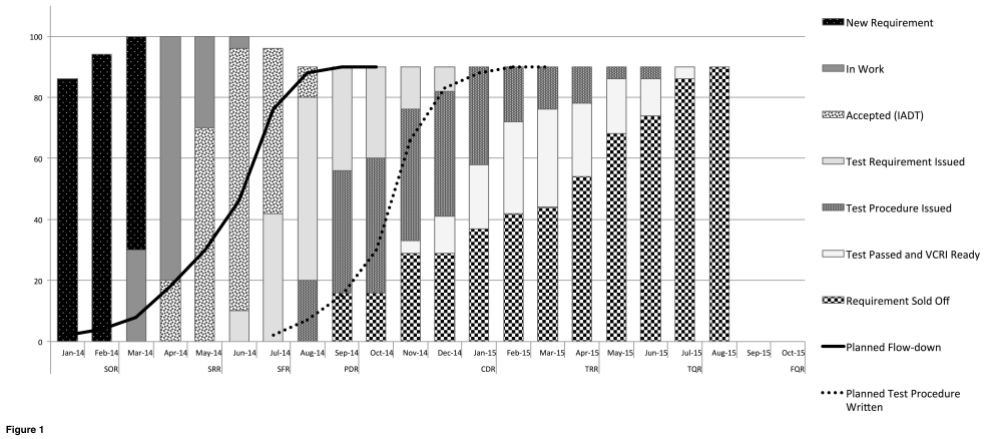

Figure 1

Figure 1 Generic Representation of the Requirements Status Metric

It was apparent that the metrics (Requirement Status Metric included) must be supported by the use of planned parameter profiles; control lines to support the project in understanding if it is 'on track', and indicate possible reasons for any deviations from plan (Rhodes & Valerdi 2007; Rhodes et al. 2009; INCOSE Measurement Working Group 2010). Initial guidance from Thales on the creation of planned parameter profiles was generated through interviews with Systems Engineering experts, and accordingly a Planned Performance Profile Guide for internal use has been written, guided by the work done by Rhodes et al. (2009) and based upon expected performance at lifecycle milestones. However, Rhodes et al. additionally suggest that planned parameter profiles should be set for each metric based on historical data and programme attributes. This research therefore looks to explore how to model planned parameter profiles on project based environmental conditions.

2 Problem Summary

In the creation of planned parameter profiles for Thales we initially observe a conflict between on one hand the emphasis on standardisation of approach and learning from experience, and on the other, encouragement for customisation and local tailoring. This conflict was made more apparent when we reviewed both historical and newly collected data for consistency and underlying meaning. It was observed that many of the metric distributions from the projects have a characteristic S-curve shape, but that there was a great deal of variation and noise in individual traces. A range of environmental conditions including team productivity and competence, project difficulty and complexity, technology readiness, and quality of work could all cause variation in this curve between projects. We therefore expected that a model based on historical data could be created to generate planned parameter profiles, based on these known project environmental conditions. In reality, however, this noise was too great to allow credible planned parameter profiles to be developed (Rhodes et al. 2009; INCOSE Measurement Working Group 2010).

We were left with the problem, therefore, of providing assistance in the creation of planned parameter profiles with little historical data. We attempted to explore this problem through observation, to understand why we could see this characteristic S-curve. Through interviews with experts we established that the slope of the central part of the S-curve was related to productivity when the project work rate had reached steady state; the reduced gradient at the start was attributed to start-up issues and 'learning how to do the task', and the tail-off at the end was attributed to either difficult aspects being left to the end of the project or to the required sorting out of issues and conflicts between project elements. This tail off can be attributed in part to hidden rework, latent defects, and customer changes to the flow of work (Davies & Hunter 2000). We will explore this through the rework cycle, a common archetype of project behaviour, whose application has been effective as a tool for exploring and predicting project behaviour (Lyneis & Ford 2007; Cooper & Lee 2009; Roberts 1974; Cooper & Mullen 1993).

The remainder of this paper will present how we took these basic observations and modelled them into a form in which we were able to explore this characteristic S-curve shape and its variation in line with a series of project based environmental conditions. We have structured our report to reflect the logical sequence of the reasoning behind our work and therefore present the methodology in the next section, followed by a brief rationale for the use of a learning system metaphor and then present the literature review.

3 Methodology

This paper will outline the results of a single case study. The question posed presented no clear hypothesis to test, instead promoting a focus on studying the problem situation. We aligned our approach with the process epistemology of studying organisational change, as described by de Ven & Poole (2005), and the presentation of work as a narrative account of change. Furthermore, our approach aligned with the soft systems thinking described by Checkland and Holwell (2004), where systems models are treated as *epistemic devices* and are used by stakeholders to help them *learn* about a problematic situation. As described in the introduction, the implementation of our methodology is based on Soft Systems Methodology (SSM). Two major iterations were undertaken; i) Initial literature understanding, model building, and review; and ii) further literature review, final model construction, and review of results. We turned to the existing literature to establish a modelling technique and then explored the observations made in relation to both the selected technique and existing models. System Dynamics (SD) was selected since the problem context was firmly grounded in dynamic behaviour and there was prior application in a series of project management situations (further explored in Section 5.1). The methods for model building will be explained in Section 3.1, and the methods used for validation and verification are explained in Section 3.2.

3.1 Model Building

We took an iterative approach to model design, following a similar approach to that given for the early stages of Group Model Building (Vennix 1996). Practically this meant we included the following stages from Vennix; i) initial speculation, ii) creation of a base model, iii) reference to various stakeholders for initial design check, iv) narrative analysis, and v) model development. Within this approach, we considered Sterman's (2000) generic framework for deriving an SD model. This process is a continual process of testing and revising mental models. Sterman's framework comprises a series of steps; i) identify the problem, to allow modelling of the problem and not the system; ii) develop a hypothesis of why the system is behaving the way it is; iii) test hypothesis, via the testing of mental models against the virtual and real world; and iv) test policy alternatives, to determine best policy alternatives. Our method was therefore to follow the steps identified by Vennix (1996), to create the model, and utilise Sterman's (2000) method for validating and verifying the model. In reality this was broken down into two phases, with phase 1 focusing on creation of a base model and phase 2 developing this model following validation and verification activities, though this will be explored simultaneously here.

3.2 Validation/Verification

Application of the model required proper validation and verification, where verification is meeting the needs of the user (and client) and validation is determining the ability of the model to provide an accurate representation of actual project behaviour. This section will include the use of narrative analysis, identified as particularly powerful in complex situations by Kurtz and Snowden (2003), and promoted by Sterman especially to test underlying assumptions, and sensitivity of results (2000, chap. 21).

Barlas (1996) identifies that any given validation process will vary dependent on the model type. He notes that many ways exist to differentiate between validation of models, though there exists a crucial differentiation between 'white box' and 'black box' approaches. A black box approach claims no causality in structure, is data driven in nature, and produces purely correlational results. A white box approach on the other hand requires the need for a causal description of individual relationships within the model. Therefore, the ability to differentiate between two major model forms, a black vs. white box approach, is effectively determined by the need to be able to understand the causal relationships, i.e. are we making any claims of causality or simply interested in ensuring real data meets predicted data? We therefore considered the purpose of this model. In our problem statement we sought to establish a causal relationship between quality and project performance, and therefore were led down the white box causal-descriptive approach to validation. This affected the selection of verification and validation methods, given the difficulty in verification and validation of a white box model where there are no established formal testing methods and the model has to have a sound philosophical basis (Barlas 1996).

The verification and validation methods were therefore selected to ensure we created a model that; i) met the needs of the user, ii) reproduced the desired behaviour, and ii) explained how this behaviour was

generated. The distinct lack of formal testing methods meant we had to rely on subjective judgement (Barlas 1996). This included expert review, inspections, walk-throughs, and consistency checking. The methods we selected were split into three distinct sections. The first of these was designed to check the model met the needs outlined by Thales – that the model could be used to explore how varying simple inputs changed the planned parameter profiles. We achieved this verification via expert review, and comparison with existing company process and guidance. Secondly we checked that the behaviour of the model was consistent with the empirical studies and available company historical data, achieved via consistency checking. These initial two tests indicated that the model was verifiable in that that it met the needs of Thales and that it ensured that the right behaviour could be generated. The final section involved validation of the model against results expected in practice. This was achieved through inspection of results and consistency checking of expected behavioural patterns, i.e. that varying a specific environmental condition gave the appropriate response *ceteris paribus*. A series of narrative tests and a sensitivity analysis were undertaken to check consistency against existing mental models.

4 The Learning System

As a part of initial speculation activities (see Vennix above) with a group of experts, mental models regarding workflow through projects were discussed and the metaphor of projects as 'learning systems' emerged with a high degree of common understanding. This metaphor postulates that if the Requirement Status Metric (RSM) was considered representative of how much is known about the customer problem within the project, then the underlying mechanism explaining the shape of the curve could be viewed as 'learning' or 'knowledge transfer'. The RSM is thus now viewed as a proxy of the *state of learning* within the project about the problem, the solution, and the interaction between problem and solution. The idea of re-interpreting the rework archetype in this way is novel, although there is related work by Morrison (2008) using System Dynamics to model the state of experience in learning a new task.

In our research we termed the 'knowledge gap' as the difference between what is known at the start of the project by the project team and what needs to be known to bring about successful completion. Therefore, a knowledge metric is relative to the knowledge gap. In the absence of any prior knowledge and perfect project performance the knowledge gap can be conceptualised as the total required knowledge remaining for a project to be successful, i.e. at the start of a project this will be 100% of required knowledge, and at the conclusion will be 0%. However, in reality the starting point is less than 100% of this idealised gap, due to projects only ever being started with some background information, and at the end of a project the gap is likely to be greater than 0%, as some unknowns may only be resolved following the end of the project (e.g. during service).

Through the same reasoning we can consider any rework/error detection element as a discovery system. This represents the search for learning that is incomplete, incorrect, or that no longer fits within the overall knowledge of the project. This knowledge can be discovered through looking for it, or it remains as undiscovered (the type of rework that causes project overruns in both cost and schedule).

We therefore speculated that if the two behavioural mechanisms we had observed were caused primarily by the dynamics of the knowledge metric and rework discovery processes, then by comparing these to existing models and empirical data we would be able to construct a useful model.

5 Literature Review

5.1 System Dynamics

System Dynamics (SD) was developed through research at MIT. Initially through the work of Forrester (1961; 1958), the use of SD as an approach to model and intervene in complex business dynamics has grown to a highly diverse and active area of research and application. A useful summary of the methodology and applications is provided by Sterman (2000). In terms of its application in this work we follow Morecroft (2007) who observes that *"… system dynamics modellers do not spy systems. Rather they spy dynamics in the real world and they organise modelling as a learning process, with the project team, to discover the feedback structure that lies behind the dynamics*"; in alignment with the view of Checkland & Holwell (2004) as discussed above.

System Dynamics as a modelling approach for use in project management has been well established, as reported by Lyneis and Ford (2007). They focused on the use of SD for single projects, and determine that there are four key problem groups in which SD has found to be useful: i) the use of SD to model explicit features of a project; ii) the rework archetype; iii) the use of dynamic understanding and feedback; and iv) SD as an approach to understand adverse effects and unintended consequences. According to this classification, the research reported here relates to 'Project Features', as it is modelling an actual system of development processes, though it contains an essential element of the 'Rework Cycle'. The use of SD for modelling research and development projects is seen from as early as 1974 where Roberts models how progress may be perceived and reported differently and the pressure that may result from this on productivity and resourcing (Roberts 1974). This has been continued in work by Cooper (2009; 1980). These models have the overall aim of investigating how changes made throughout a program will influence its progress and often include many other causal relationships outside of the immediate movement of work (Cooper & Lee 2009).

Throughout the Systems Dynamics literature are a series of existing models, commonly known as archetypes (Senge 1990; Braun 2002). We investigated these in comparison with the identified learning

system mechanisms, noting that two examples can often be seen. The first of these archetypes is the S-shaped logistic curves developed by Sterman (2000, chap.9), where this can be viewed as having a close relationship to a *learning system* (Morrison 2008). The second archetype is the rework cycle, which is seen extensively in the literature. First formalized by Cooper (1980; 1993; 2009) in a series of papers, and summarized by Sterman (2000, chap.19). The rework cycle consists of a four stock model where tasks flow from <Work To Be Done> through to <Work Really Done>. There is a primary flow between these two stocks alongside a concurrent rework cycle. This contains the stocks <Undiscovered Rework> and <Discovered Rework>. This representation is shown in Figure 2.

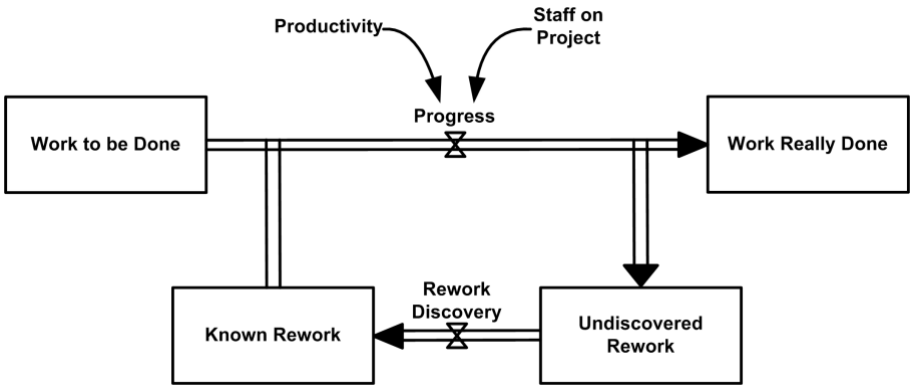

Figure 2 Rework Loop

5.2 Empirical Data Fitting

The lack of available historical data has meant a reliance on qualitative validation focussing on the overall morphology of the time histories of the stocks. One way of providing this validation is comparison of model results with the empirical data from other researchers. We began by looking at the work by Putnam (1978) to better understand the effect of rework on the shape of the graphs: <Work Really Done>, and <Learning Rate> against time. Putnam's results were obtained from analysing performance of software engineering projects, which we believe to be applicable to the complex systems engineering problems we are investigating, due to the origins of much of Systems Engineering practice in early Software Engineering practice.

This distribution, arising from the relationship between effort and delivery time for software projects has become known as the Putnam-Norden-Rayleigh (PNR) distribution. This is based initially on the Rayleigh curve, and the work in the 1960s and 1970s by Peter Norden (1958; 1960) to show the relationship between manpower and project duration. The curve produced shows the variation of effort for a project against time and looks like a normal distribution with an extended right hand side (Putnam 1978). This can be conceptualized as the distribution of likely project duration arising from the concatenation of a set of tasks, where the duration of each task is described as a sample from a three point estimate. The integration of this distribution with respect to time gives the characteristic S-curve shape with a flattened top section

as shown in Figure 3. This same shape is also seen in the work on the rework cycle and its implications as described by Mawby and Stupples (2002). They use SD to create a model that helps with decision making to allow early mitigation of change in projects (where the cost is much less). Mawby and Stupples introduce the notion of delay on the rework discovery, and their results show the overall work curve as affected by both work done (if quality is perfect), and the addition of rework. This is a good approximation of the PNR distribution, however in this case it emerges from the behaviour of the model. This helped to justify the use of SD, and supports the use of a model to generate behaviour that is recognisable from actual project behaviours.

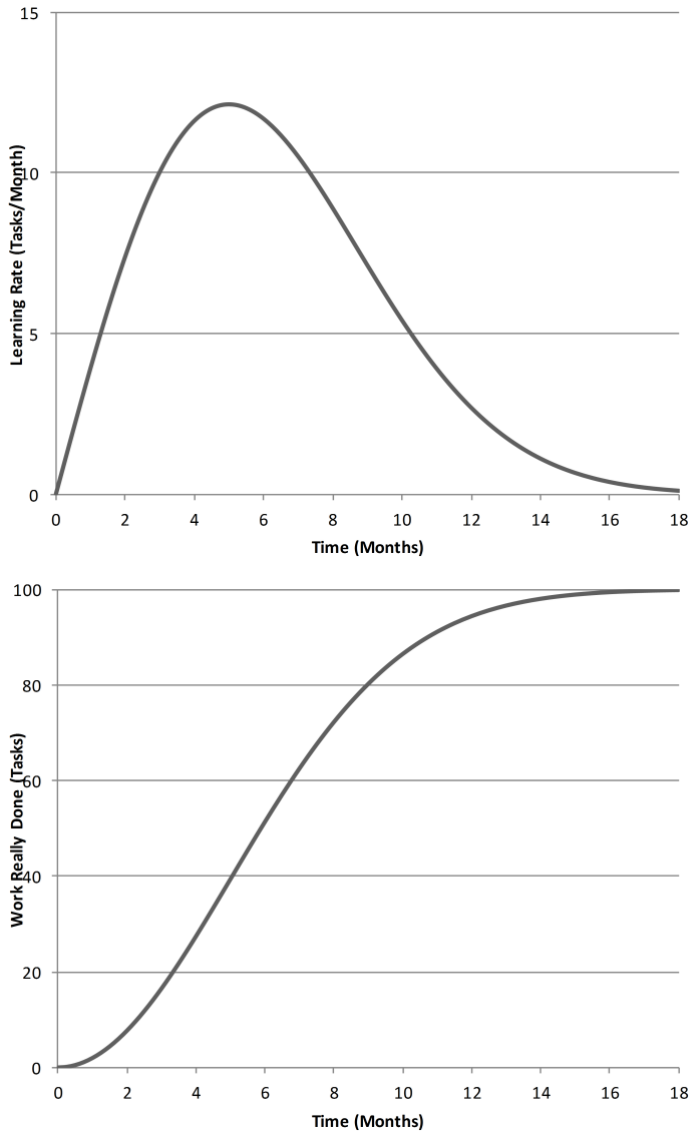

Figure 3 The PNR Distribution (Req't/Month vs. Time & No. of Req't vs. Time)

5.3 Error Detection

We selected the rework cycle archetype due to its ability to reflect the injection of latent defects or changes to the system being designed. In the rework cycle these defects or changes are represented by the flow into the stock <Undiscovered Rework>. We focused therefore on the application of an existing error

detection model. Very little literature exists with reference to the use of error detection models in systems engineering, with most models basing their findings in software development. Numerous of these are identified by Abdel-Ghaly et al. (1986), though the accuracy of each type may vary greatly. The most popular types are either classified as data-domain or time-domain (Gokhale et al. 1996). Data domain models are based on the philosophy that if all inputs to model are identified then the model can be run and reliability inferred. Using a data driven model was neither useful nor appropriate for the model that we are applying as they rely on an exhaustive search for system inputs, which was practically impossible for this research due to the high number of factors that effect a project. Time-domain models focus on the underlying process, using observed failure history to estimate residual errors. Time domain models can be further categorized between: homogenous Markov models; non-homogenous Markov models; and others (Putnam & Myers 2003). Since we have use a single stock for <Undiscovered rework> we are unable to differentiate between types of error, so the use of a homogenous Markov model is suggested. The Jelinski-Moranda equation was chosen due to its success in the modelling of many data sets (Jelinski & Moranda 1972). This model makes the following assumptions; i) The rate of fault detection is proportional to the current fault content, ii) The fault detection rate remains constant between fault occurrence (given the change assumed in i)), iii) A fault is corrected instantly, and iv) All faults have an equal chance of discovery. These assumptions produce an error detection model that was deemed suitable for use as a basis for our approach.

6 Model Creation

In reality two iterations of modelling occurred. This began with initial speculation and development of a shared model of projects as learning systems. Following this a review of System Dynamics literature surrounding the rework cycle and with reference to the diffusion model led to the creation of the initial model. Following this a short review was undertaken with experts, and the learning from this review alongside further comparison with the learning system metaphor was taken into consideration in the development of the final model. This paper will depict these phases as a single process, to aid in clarity.

The model can be analysed by viewing it in two halves, as shown in Figure 4. The top half corresponds to, the learning system, described as a diffusion model where the stocks <Work To Be Done> and <Work Really Done> form the basis of the learning. The bottom half corresponds to the effect of the rework cycle, the effect of misunderstanding and discovery of that rework.

6.1 The Learning System

The learning system interpretation of the rework model is suitable provided that we consider the units for the stocks and flows to be a proxy measurement of the state of required learning within the project. The collective learning of the project team is represented by a curve of increasing gradient that eventually

reaches a maximum rate corresponding to the maximum ability of the project team to learn how to implement the project. Following this, the rate of learning slows as the project team begins to deal with the consequences of what has been misunderstood. Applying this learning analogy means that we can continue to use the <Total Number of Tasks> to set the initial conditions for the model, whilst using the idea of bridging the 'knowledge gap' to explain the <Learning Rate>, which determines the rate at which work is actually being completed.

The dynamics of the model are therefore determined by two feedback loops, B1 'Learning Potential of the team' and R1 'Learning about the Problem'. The former (B1) is governed by the <Learning Power> and <Number of Staff> in the team. The strength of the latter (R1) is determined by the amount of <Effort> that is put in by the team. The environmental condition <Number of Tasks> is used as a constant twice: to normalise the size of the problem and to provide the initial value for the stock <Work To Be Done>.

By modelling the transition from <Work to be Done> to <Work Really Done> using a logistic model we have captured the idea of representing the project team as a learning system using these stocks as proxies for the state of learning. Ignoring for now the effect of the rework cycle (i.e. by assuming that <Quality of Work Done> is equal to 1.0) the change in state of learning, the <Learning Rate>, is calculated as follows

$$Learning\ Rate = (Number\ of\ Staff * Learning\ Power * Work\ to\ be\ Done) + (Work\ to\ be\ Done * Effort * Work\ Really/Total\ Number\ of\ Tasks)$$

Again, this is equivalent to Sterman's (2000) expression for logistic growth stated in Equation 9-39 and thus this top half of the model (i.e. without the rework loop) is structurally identical to Figure 9-18. For the remainder of our analysis and rest of the paper we assume that the <Total Number of Tasks> remains a constant.

The rework cycle is operative once the <Quality of Work Done> becomes less than 1.0. Tasks accumulate in the stock <Undiscovered Rework> at a rate determined by

$$Misunderstanding\ Rate = (1.0 - Quality\ of\ Work\ Done) *\ Learning\ Rate$$

which is consistent with the interpretation of numbers of tasks in the stock as a proxy for the state of learning, or in this case the amount of mis-learning that has taken place. We discuss the <Rework Discovery Rate> and <Schedule Rate> in the following section.

6.2 The Rework Cycle

We applied the same learning system to the error detection rate. Our initial efforts to base this value on the <Quality of Work Done> did not stand up to scrutiny through either narrative verification or tests against the proposed learning model. This gave the model the correct macro behaviour, but not for the

right structural reasons. We therefore look to develop a model that better reflected the detection of <Unknown Rework>, basing this on the Jelinski Moranda (1972) model, as discussed in Section 5.2.

To allow the model to produce a learning curve based only on user-based predictive inputs, we normalized the Jelinski-Moranda model in relation to time (within the model), and then implemented this modified model, using estimated environmental conditions. The environmental conditions used were reached by comparing the normalised Jelinski-Moranda model to projects in practice. We made the assumption that project resource (at a macro level) is fixed for a project, and therefore there are three environmental conditions that will affect the ability of the project to discover rework: the <Urgency> for the start of the discovery phase (related to the overall length of the discovery); the initial <Intensity> of the discovery effort; and the continued desire for intense discovery given by <Attention Span>. i.e. when do we start looking (and how long for), how hard do we look to start off with, and do we continue looking as hard the entire time?

6.2.1 Final Model

The final model is displayed in Figure 4. We include a summary of the environmental conditions present in this model, and how this relates to projects being undertaken in Table 1. It was obvious to the authors that causal links between these conditions could be made. Some of these links were considered, but, the decision was consistently made to keep the model as simple as possible, to allow exploration of the environmental conditions on an individual basis.

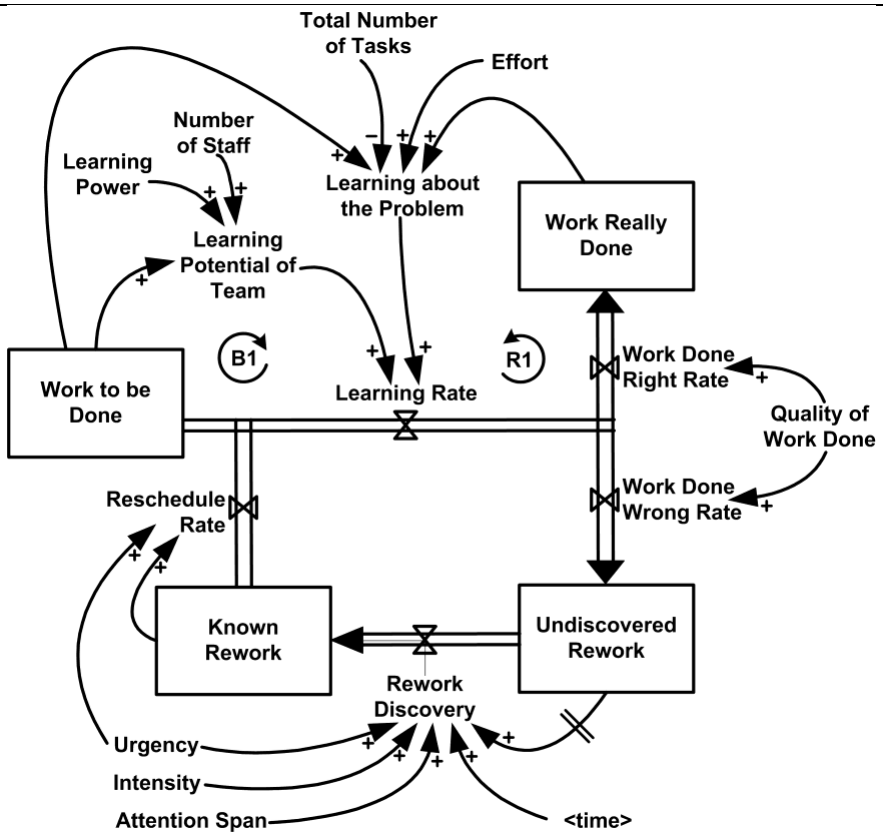

Figure 4 Final Model

Table 1 Environmental Conditions

Environmental Condition	Practice Notes
<Learning Power>; <Number of Staff>; <Learning Potential of the Team>	These are designed to represent the size of the project team, and the competency and skill of that project team. These impact the ability of that team to complete tasks (the 'learning potential').
<Effort>; <Learning About the Problem>	Differing from the ability of the team, these focus on the amount of work that a team actually do in relation to the project. This is affected by the size of the project in question, governed by the <Total Number of Tasks>.
<Total Number of Tasks>	This is a condition to indicate the amount of work required. We have represented this as the number of tasks, but it could be detailed as story points, requirements or similar (although the assumption is that they are identically sized units). In learning system terms this is related to the size of the knowledge gap to be bridged. This can be influenced by prior knowledge and understanding.
<Quality of Work Done>	An indication of how much work is done right or wrong as part of everyday work. This can be as a result of a multitude of reasons (see literature regarding causes of rework [REF TONNELLIER].
<Intensity>	In reality these refer respectively to: the commitment of resource and energy to search for hidden rework; the urgency with which resources are allocated to start looking for hidden rework; and the continued length of time that is spent of looking for remaining defects. These have strong connections to resourcing and quality, but are purposefully kept separate. This refers to the learning system discovery system, where the conditions can be seen to have the same effect.
<Urgency>	
<Attention Span>	

7 Results

7.1 Verification and Validation

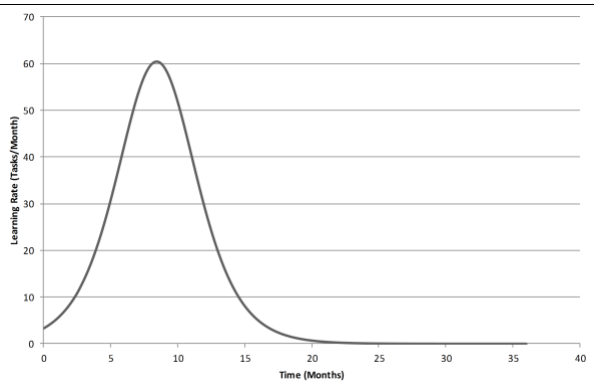

Figure 5 <Learning Rate> (Req't/Month vs. Time)

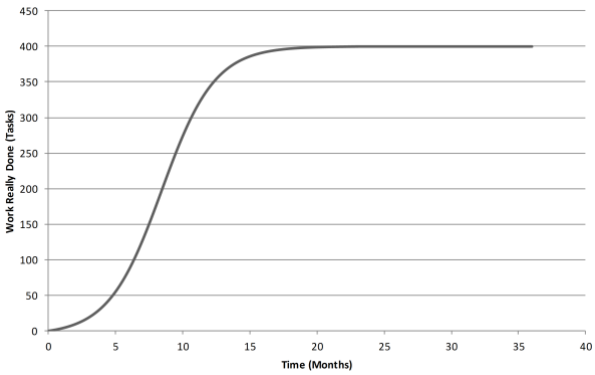

Figure 6 <Work Really Done> (No. of Req't vs. Time)

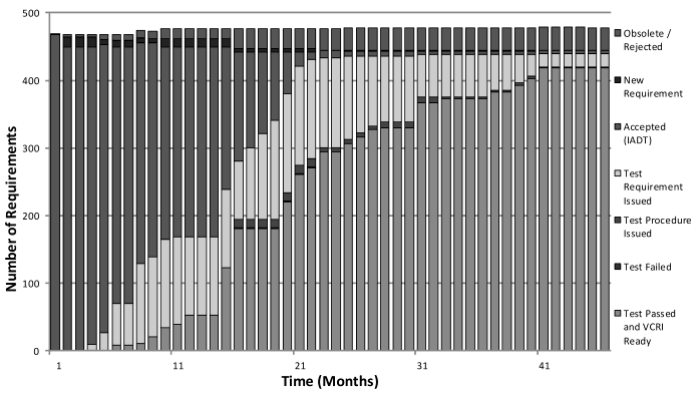

Figure 7 Historical Thales RSM Data (No. of Req't vs. Time)

Expert analysis took place with members of the senior systems engineering functional leadership. In their opinion the model was able to assist with the development of Planned Parameter Profiles via the input of simple environmental conditions. These simple environmental conditions directly relate to the characteristics of projects in practice, and the Planned Parameter Profiles produced allow comparison with actual project data. This will allow early identification of project problems.

We followed this expert analysis with verification against empirical and historical data. Our model has been successful in reproducing the expected dynamic shape suggested in the empirical work. Figure 5 shows the model generated curve of <Learning Rate> against time, and Figure 6 shows the model produced curve of <Work Really Done> against time. This shape has high similarity with both the PNR distribution and the

work by Mawby and Stupples, showing the distinctive form of the derivative of the logistic function with an extended right hand side, pushed that way due to the rework in the system. By applying our learning system, we conceive this as a slowing in the rate of learning towards the end of the project, due to either an increase in difficulty, or difficulty in interrelated elements of this learning. The shape of the data produced by our model was then compared to the limited historical data within Thales. Shown in Figure 7, this historical data shows several distinct S-curve shapes: a symmetrical S-curve at the transition from in negotiation and verification test written, reminiscent of the curves produced with a higher value of quality; and a more flattened S-curve at the transition from 'test verification' through to 'test passed/failed', reminiscent of a curve shape that exists when the PNR distribution is integrated.

Our final validation step involved examination of the causal relationships. This would allow observation of the micro-behaviour of the model, and validation of its ability to generate results expected in the real world. We did this through a sensitivity analysis of the key environmental conditions alongside narrative description of the expected behaviour. We interviewed members of the senior systems engineering functional leadership. This analysis proved to be satisfactory with the model producing results similar to those observed from actual project behaviour. The variation of outputs with the independent environmental conditions (number of requirements, level of hidden rework) was in most respects consistent with the expectations of experienced practitioners. The following section further details this analysis.

7.2 Model Behaviour

In this section we will highlight the results of some of the analysis undertaken during the final validation stage described in the Section 7.1. Figure 8 shows a set of planned parameter profiles based on varying the estimate of <Quality of Work Done>. As expected, the S-curves begin to show a marked flattening as quality decreases. This is consistent with empirical data and indicative of a longer right hand tail of the PNR distribution. One of the most interesting behaviours observed is the models prediction that a decrease in <Quality of Work Done> leads to an initial increase in rate. However, due to the increased rework inherent with a lower level of <Quality of Work Done>, the requirements take longer to reach their target value. This behaviour is consistent with expert opinion. We investigated further scenarios included the effect of reducing or increasing the <Learning Potential> available to the project. Figure 9 shows that the <Learning Rate> does not increase linearly in line with the increase of <Number of Staff>, as expected by the SE experts we interviewed. Further to this, a doubling in <Number of Staff> should not result in a doubling of the <Learning Rate>. The other behaviour that these results show is that the amount of change in <Learning Rate> gets smaller as the <Number of Staff> gets larger. SE experts recognise that in reality, the work done by a project is not proportional to the number of people involved, suggesting that this problem is related to the projects' teams' ability to share information internally. This supports the learning system conceptualisation, the idea that a project undergoes a comprehensive learning process, and that the more

people there are the slower this learning rate is likely to be (on a whole project basis) due to a surmised (N^2) scaling of the rate of information exchange.

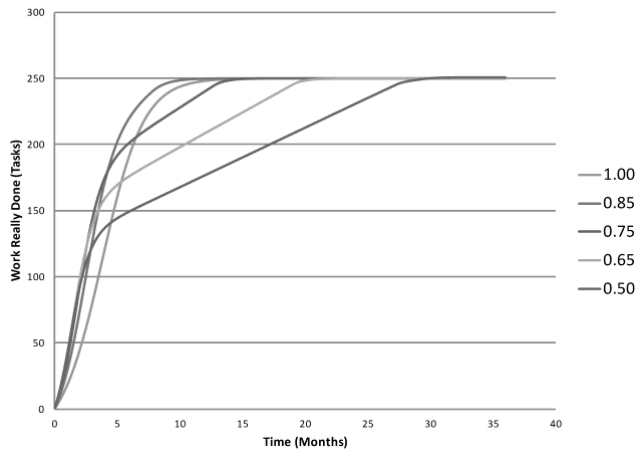

Figure 8 Varying <Quality of Work Done> (No. of Req't vs. Time)

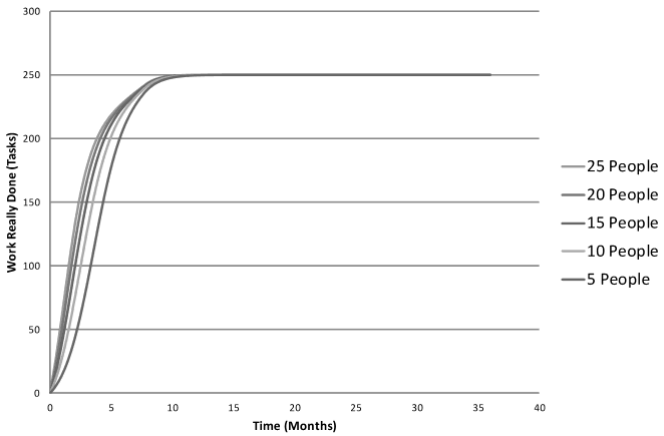

Figure 9 Varying <Number of Staff> (No. of Req't vs. Time)

We then investigated the effect of varying the three rework discovery environmental conditions to compare the actual behaviour with the expected behaviour. These results are shown in Figure 10, Figure 11, and Figure 12. The behaviour we observed, was as expected, with the <Rework Discovery Rate> exponentially decreasing across the rework discovery period (this line is not smooth due to the model preventing the value of undiscovered rework falling below zero). Due to the delay enforced by the <Urgency> value, the stock of <Undiscovered Rework> increases until the <Rework Discovery Rate> kicks in and reduces this. The amount that this reduces the <Undiscovered Rework> to by the end of the model is dependent on the environmental conditions, highlighting how rework can remain undiscovered if the correct amount of resource is not applied. The simplest value to understand is the variability of <Intensity>. This simply increases the amount of rework discovered across the defect discovery phase, whilst the start and finish points remain identical. Given the same initial environmental conditions, when <Intensity> is higher, the rework is discovered early and resolved quickly. Small amounts remain to be discovered later in the process, and this model assumes that the search is continued at the equivalent rate. <Urgency> has several affects. Primarily it changes the starting point of the <Rework Discovery Rate> curve, shifting it to

the left. It increases the length of time that rework is searched for, and as a result of this, the amount of rework that can be discovered is increased/decreased to retain the shape of the curve. The <Attention Span> input would suggest the lengthening of the curve, but that has been dealt with by the <Urgency> input. Instead, <Attention Span> is used to understand how the intensity of the rate changes over time. The higher this value the closer to a constant rate decline is seen, and the lower the value the larger the concavity of the exponential model. This is the equivalent of how long you look for defects at the same intensity.

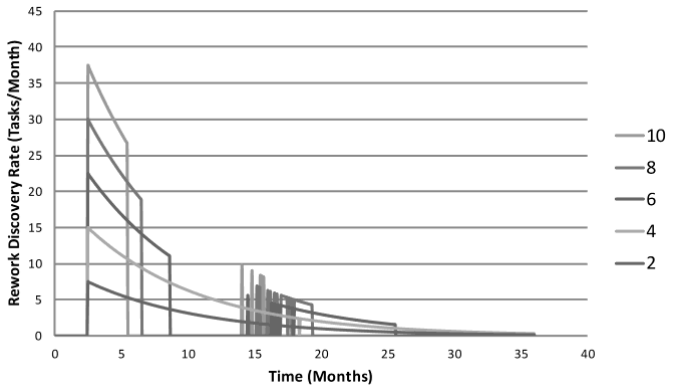

Figure 10 Varying <Intensity> (No. of Req't vs. Time)

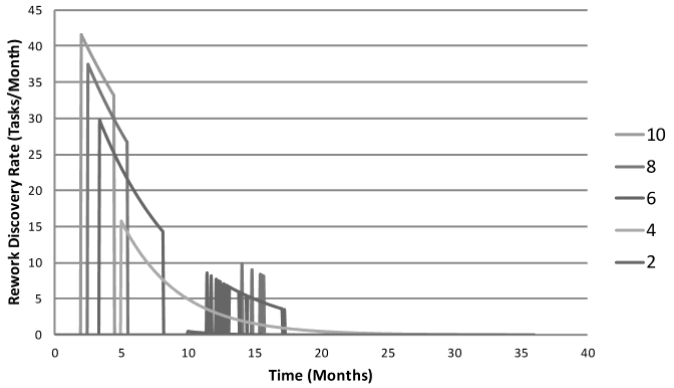

Figure 11 Varying <Urgency> (No. of Req't vs. Time)

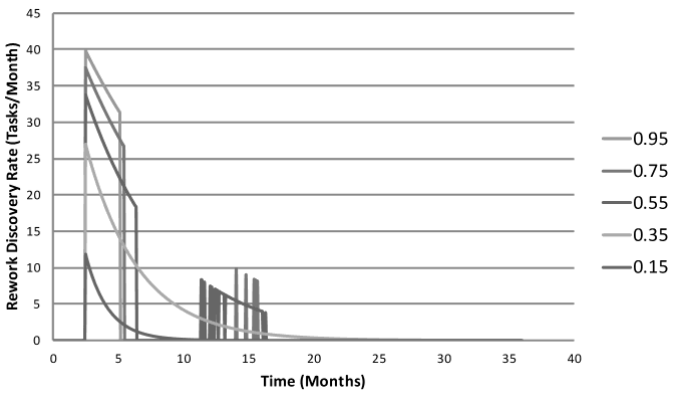

Figure 12 Varying <Attention Span> (No. of Req't vs. Time)

Finally, to ensure that the model was robust to reality checks (extreme condition testing) (Sterman, 2000, pp. 555-556) to identify the bounds within which appropriate PPP could be generated, a series of short tests were undertaken on the main environmental conditions, as shown in Table 2.

Table 2 Extreme Condition Testing

Environmental Condition	Minimum Test	Minimum Results	Maximum Test	Maximum Results
Learning Power	0	No Learning Rate, therefore No Work Done Rate (No Work Done).	1	Increased Work Done Rate (maximum value is dependent on other environmental conditions).
Number of Staff	0	No Learning Rate, therefore No Work Done Rate (No Work Done).	∞	As the Number of Staff increases, the Work Done Rate increases. This relationship

				shows exponential decay, and therefore a maximum value is reached (dependent on other environmental conditions). See Figure 9.
Quality of Work	0	No Work Done.	1	See Figure 8.
Number of Tasks	0	No Work to be Done (No Work Done).	∞	As the number of tasks increases the time taken to complete the tasks increases. This has no impact within realistic numbers on the ability of the model to function (tested to 10000).
Effort	0	No Work to be Done (No Work Done).	1	This increases the Work Rate. However since this is governed by a balancing feedback loop, this has a limited effect.
Intensity	0	Rework Discovery Rate tends to infinite (floating point exception).	1	See Figure 10.
Urgency	0	Rework Discovery Rate tends to infinite (floating point exception).	1	See Figure 11.
Attention Span	0	Rework Discovery Rate tends to infinite (floating point exception).	1	See Figure 12.

8 Summary

We were looking to create a model that could create Planned Parameter Profiles, and then help to explain the underlying relationships that can affect the shape of these Planned Parameter Profiles. Our results show that our SD model is able to perform both of these functions. The results show how the use of simple environmental conditions has enabled fast creation of planned parameter profiles with the same expected morphology as that seen in empirical and historical data, and consistent with SE expert opinion. More importantly, varying these inputs creates the same recognisable behaviour as SE experts expected. This enables the tool to be more useful in understanding the underlying mechanisms that may be creating problems on projects.

8.1 Application

This paper has demonstrated the creation of a model that can be used to create planned lines. In theory this would align with the use of leading indicators for metrics as described by Rhodes et al. (2009):

1. Planned lines are created in the context of project planning, based on environmental conditions established by the project team.
2. The lines created are then regularly reviewed to understand the trend, and deviation from plan can be established.
3. Response is taken by project, through change to an environmental condition (additional learning power, focus on quality of output etc.).

4. Outcome should see project either return to original PPP, or revise PPP based on changes in environmental conditions.

However, in order to maximise benefit, the company wanted to understand how to apply this model most effectively. In the introduction we identified the difficulty in effectively deploying standardised metrics across a culturally diverse organisation. We found this problem was amplified once we tried to parameterise the model, due to an environment where every project is different, and each project has a certain amount of noisy data already in existence. This model was therefore unable to generate exact projects curves for given environmental conditions, but could prove useful for project teams as a tool to aid their metrics (and project) planning, and as a tool to explore deviations from the original planned line. To increase the ability of the model to achieve this, we turned to another approach for dealing with complex problem situations.

8.2 Problem Structuring Methodology

The need for approaches to help structure problems with high levels of complexity is noted by Jackson, Keys, Kurtz & Snowden, and Checkland (Jackson & Keys 1984; Jackson 2003; Kurtz & Snowden 2003; Checkland 2000). The development of these approaches into what have commonly been described as Problem Structuring Methods (PSM) has been highlighted by Rosenhead, Mingers and White amongst many others (Rosenhead 1996; Ackermann 2012; Eden & Ackermann 2006; Keys 2006; Mingers & Rosenhead 2011; Mingers 2011; White 2009; White 2006). Recent work by Yearworth and White (2014) provides a generic constitutive definition for PSMs, which states that they are methods that deal with structuring action to identify improvements in the problem context, rather than attempting to solve problems directly; recognizing that wicked and messy problems are resistant to outright solution. PSMs are by nature participative and interactive, and therefore appropriate for dealing with complexity, uncertainty and ambiguity (Davis et al. 2010). The use of systems modelling to support problem structuring falls clearly within a soft-systems paradigm as defined by Checkland & Holwell (2004). Viewed from this stance, the System Dynamics model developed in our work is not a functionalist representation of the project as a system; it is modelling as a learning process focussed on discovering the feedback structures that lie behind the dynamics we have been observing (Morecroft 2007).

8.3 Model Benefits

The approach we have developed allows the exploration and structuring of the project supporting the problem solution. This exploration relies on some of the project characteristics to create the expected performance profiles which we hypothesized would allow earlier indication of project performance when compared with actual project data. This led us to a model designed to enable benefits in the following areas and answered the research aims described in the introduction:

- The model allows learning about the nature of Systems Engineering projects. It is able to demonstrate qualitatively how project characteristics affect the dynamics of the metrics traces. This should be particularly useful to illustrate the importance of minimising the amount of rework that is being created and highlight the importance of early and continued detection of hidden rework.
- The model allows early problem structuring in relation to the planning of a project. It allows project teams to establish credible performance baseline dynamics for a given project based on simplified project quality characteristics, resources, and defect detection characteristics.
- The model allows the project to learn about itself. It is able to provide possible and likely explanations for deviations from the performance baseline and allow managers to explore what-if scenarios to aid decision making about appropriate corrective action. The ability to explain possible reasons behind deviations from Planned Parameter Profiles, and then to simulate different intervention strategies and their influence on metrics traces should help projects determine the most appropriate intervention strategies.

9 Conclusions

We began this research with the understanding that Thales needed a way to generate planned parameter profiles for the technical metrics the company uses to assess technical maturity of projects. These metrics are designed to deal with the problem of understanding technical progression. Our study of best practice in that area had identified the need for these Planned Parameter Profiles. We have shown that use of existing methodologies in conjunction with each other and careful consideration of how to validate and verify models allowed the creation of an appropriate System Dynamics model. We have shown how our proposed learning system for bridging the project's 'knowledge gap', coupled with the rework cycle archetype, produces Planned Parameter Profiles for a requirements based metric. These Planned Parameter Profiles have the same characteristics as data from actual projects and that variations in the model environmental conditions produce the expected changes in output traces, both according to expert opinion and as seen in empirical data. This triangulation verifies and validates the model we have created. Our approach avoids the need for full System Dynamics development, as seen in Cooper & Lee's work, instead providing a simple and early diagnostic of Planned Parameter Profiles.

Our model does not look to accurately predict future project performance, but provides a fast development of simple Planned Parameter Profiles. This model can then be used to educate team members about the feedback structures within their project; contributing to the underlying learning process and the identification and discovery of rework within the project. We can use the model to help develop Planned Parameter Profiles as a part of a project structuring and planning process and remove the need for empirical parameter fitting from standard guidelines. During the lifecycle of the project, the model helps to understand and diagnose deviations from the plan (and assist in the required re-planning). This work

contributes to a better systemic understanding of how Systems Engineering projects work, and to better methods for measuring and controlling systems engineering projects.

10 Bibliography

Abdel-Ghaly, A.A., Chan, P.Y. & Littlewood, B., 1986. Evaluation of Competing Software Reliability Predictions. *IEEE Transactions on Software Engineering*, 12(9), pp.950–967.

Ackermann, F., 2012. Problem structuring methods "in the Dock": Arguing the case for Soft OR. *European Journal of Operational Research*, 219, pp.652–658.

Barlas, Y., 1996. Formal aspects of model validity and validation in system dynamics. *System Dynamics Review*, 12(3), pp.183–210.

Baskerville, R.L. & Wood-Harper, A.T., 1996. A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11, pp.235–246.

Braun, W., 2002. The System Archetypes.

Checkland, P., 2000. Soft Systems Methodology : A Thirty Year Retrospective a. *Systems Research and Behavioral Science*, 58, pp.11–58.

Checkland, P. & Holwell, S., 2004. "Classic" OR and "soft" OR - an asymmetric complementarity. In M. Pidd, ed. *Systems Modelling: Theory and Practice*. Chichester: John Wiley & Sons Ltd.

CMMI Product Team, 2001. *Capability Maturity Model ® Integration (CMMI SM), CMMI SM for Systems Engineering , Software Engineering , and Integrated Product and Process Development: Continuous Representation*, Carnegie Mellon.

Cooper, K. & Lee, G., 2009. Managing the Dynamics of Projects and Changes at Fluor. In *Proceedings of the International System Dynamics Conference*. Albuquerque, pp. 1–27.

Cooper, K.G., 1980. Naval Ship Production : A Claim Settled and a Framework Built. *Interfaces*, 10(6), pp.20–36.

Cooper, K.G. & Mullen, T.W., 1993. Plowshares - The Rework Cycles of Defense and Commerical Software. *American Programmer*, 6(5).

Davies, P. & Hunter, N., 2000. System Test Metrics on a Development- Intensive Project. In *Proceedings of the 2nd European Systems Engineering Conference (EuSEC 2000) Systems Engineering-a Key to Competitive Advantage for All Industries:*. Munich: Thales UK, pp. 205–212.

Davis, J., MacDonald, A. & White, L., 2010. Problem-structuring methods and project management: an example of stakeholder involvement using Hierarchical Process Modelling methodology. *Journal of the Operational Research Society*, 61(6), pp.893–904.

Eden, C. & Ackermann, F., 2006. Where Next for Problem Structuring Methods. *The Journal of the Operational Research Society*, 57(7), pp.766–768.

Elm, J.P. et al., 2008. *A Survey of Systems Engineering Effectiveness - Initial Results (with detailed survey response data)*, Pitsburgh, PA. Available at: http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=8493.

Elm, J.P. & Goldenson, D.R., 2012. *The Business Case for Systems Engineering Study : Results of the Systems Engineering Effectiveness Survey*, Pitsburgh, PA. Available at: http://www.sei.cmu.edu/library/abstracts/reports/12sr009.cfm.

Forrester, J., 1961. *Industrial Dynamics*, Massachusetts: Pegasus Communications.

Forrester, J.W., 1958. Industrial Dynamics: A major breakthrough for decision makers. *Harvard Business Review*, 36(4), pp.37–66.

Frenz, P.J., 2005. The Nuts , Bolts and Duct Tape of Establishing a System Engineering Measurement

Program. In *Proceedings of the 15th INCOSE International Symposium*. INCOSE.

Gokhale, S.S., Marinos, P.N. & Trivedi, K.S., 1996. Important Milestones in Software Reliability Modeling. In *Proceedings of Software Engineering and Knowledge Engineering (SEKE)*.

INCOSE Measurement Working Group, 2010. *Systems Engineering Measurement Primer: A basic introduction to measurement concepts and use for systems engineering*, INCOSE.

Jackson, M.C., 2003. *Systems Thinking: Creative Holism for Managers*, John Wiley.

Jackson, M.C. & Keys, P., 1984. Towards a System of Systems Methodologies. , 35(6), pp.473–486.

Jelinski, Z. & Moranda, P.B., 1972. Software Reliability Research. In W. Freiberger, ed. *Statistical Computer Performance Evaluation*. New York: Academic Press, pp. 465–484.

Keys, P., 2006. On Becoming Expert in the Use of Problem Structuring Methods. *The Journal of the Operational Research Society*, 57(7), pp.822–829.

Kurtz, C.F. & Snowden, D.J., 2003. The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Systems Journal*, 42(3), pp.462–483.

Lee, A.S. & Hubona, G.S., 2009. A Scientific Basis for Rigor in Information Systems Research. *MIS Quarterly*, 33(2), pp.237–262.

Lyneis, J.M. & Ford, D.N., 2007. System dynamics applied to project management : a survey, assessment, and directions for future research. *System Dynamics Review*, 23(2), pp.157–189.

Mawby, D. & Stupples, D., 2002. Systems thinking for managing projects. In *Engineering Management Conference*. pp. 344–349.

Mingers, J., 2011. Soft OR comes of age—but not everywhere! *Omega*, 39(6), pp.729–741.

Mingers, J. & Rosenhead, J., 2011. Introduction to the Special Issue : Teaching Soft O.R., Problem Structuring Methods, and Multimethodology. *INFORMS Transactions on Education*, 12(1), pp.1–3.

Morecroft, J.D.W., 2007. *Strategic modelling and business dynamics : a feedback systems approach*, Chichester: John Wiley & Sons Ltd.

Morrison, J.B., 2008. Putting the learning curve in context. *Journal of Business Research*, 61(11), pp.1182–1190.

Norden, P. V, 1958. Curve Fitting for a Model of Applied Research and Development Scheduling. *IBM Journal*, July, pp.232–248.

Norden, P. V, 1960. On the Anatomy of Development Projects. *IRE Transactions on Engineering Management*, March, pp.34–42.

Ormerod, R.J., 2014. The mangle of OR practice: towards more informative case studies of “technical” projects. *Journal of the Operational Research Society*, 65, pp.1245–1260.

Pickering, A., 1993. The Mangle of Practice: Agency and Emergence in the Sociology of Science. *American Journal of Sociology*, 99(3), pp.559–589.

Pickering, A., 1995. *The mangle of practice: time, agency, and science*, Chicago: University of Chicago Press.

Putnam, L. & Myers, W., 2003. *Five core metrics: the intelligence behind successful software management*, Dorset house Publishing.

Putnam, L.H., 1978. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering*, 4, pp.345–361.

Rhodes, D.H. & Valerdi, R., 2007. Enabling Research Synergies Through a Doctoral Research Network for Systems Engineering. *Systems Engineering*, 10(4), pp.348–360.

Rhodes, D.H., Valerdi, R. & Roedler, G.J., 2009. Systems Engineering Leading Indicators for Assessing Program and Technical Effectiveness. *Systems Engineering*, 12(1), pp.21–35.

Roberts, E.B., 1974. A simple model of R & D project dynamics. *R&D Management*, 5(1), pp.1–15.

Rosenhead, J., 1996. What's the problem? An introduction to problem structuring methods. *Interfaces*, 26(6), pp.117–131.

Senge, P.M., 1990. *The 5th Discipline: The Art and Practice of the Learning Organisation*, London: Random House.

Sheard, S.A. & Mostashari, A., 2009. Principles of Complex Systems for Systems Engineering. *Systems Engineering*, 12(4), pp.295–311.

Sheard, S.A. & Mostashari, A., 2013. Systems Engineering Complexity in Context. In *Proceedings of the 23nd Annual International Symposium of INCOSE*.

Sillitto, H.G., 2004. An Integrated Set of Technical Measures to Support Earned Value Management and Technical Review. In *Proceedings of the 14th INCOSE International Symposium*. INCOSE.

Sterman, J.D., 2000. *Business Dynamics: Systems Thinking and Modelling for a Complex World*, Boston: Irwin McGraw-Hill.

Van de Ven, A.H. & Poole, M.S., 2005. Alternative Approaches for Studying Organizational Change. *Organization Studies*, 26(9), pp.1377–1404.

Vennix, J.A.M., 1996. *Group Model Building: Facilitating Team Learning Using System Dynamics*, Chichester: John Wiley.

Walworth, T. et al., 2013. Early estimation of project performance : the application of a system dynamics rework model. In *Proceedings of the 7th IEEE International Systems Conference*. Orlando, USA, pp. 202–208.

White, L., 2006. Evaluating Problem-Structuring Methods: Developing an Approach to Show the Value and Effectiveness of PSMs. *The Journal of the Operational Research Society*, 57(7), pp.842–855.

White, L., 2009. Understanding problem structuring methods interventions. *European Journal of Operational Research*, 199, pp.823–833.

Yearworth, M. & White, L., 2014. The Non-Codified Use of Problem Structuring Methods and the Need for a Generic Constitutive Definition. *European Journal of Operational Research*.

Yin, R.K., 2009. *Case Study Research: Design and Methods* 4th ed., Sage Publications Ltd.