Requirements Uncertainty in a Software Product Line

Andy J. Nolan¹, Silvia Abrahão², Paul Clements³ and Andy Pickard¹

¹Rolls-Royce
SIN C-3, Rolls-Royce plc, PO Box 31
Derby DE24 8BJ, England
{Andy.Nolan,
Andrew.C.Pickard}@Rolls-Royce.com

²ISSI Group, Department of Computer Science
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia,
Spain
sabrahao@dsic.upv.es

³Software Engineering Institute
Carnegie-Mellon University
4500 5th Avenue, Pittsburgh, PA
15213-2616, USA
(412) 268-5800
clements@sei.cmu.edu

Abstract— A complex system’s requirements almost always remain uncertain late into its software development. In gas turbine engine control systems at Rolls-Royce, for a traditional project (non-product line) typically 50% of requirements will change between Critical Design Review and Entry into Service. Requirements uncertainty is particularly relevant when defining the scope of a Software Product Line. If the core asset team fails to recognise or accommodate requirements uncertainty, changes will manifest later in the product line. If the core asset team over- compensates by adding too much functionality or variability to account for a wide range of uncertainty, they will invest effort that may never be required. The optimal balance can be found through an application of requirements uncertainty analysis and understanding the balance between the impact of risk and mitigation effort. This paper first describes the use of requirements uncertainty analysis technique used at Rolls-Royce for traditional (non-product line) software development and then explains how this technique works in the context of a software product line.

Keywords: Requirements, Uncertainty, Software Product Lines

I. INTRODUCTION

A common reason for project problems is insufficient management of changing requirements during all stages of the project life. One of the root causes for changing requirements is requirements uncertainty [6] i.e., requirements changing unexpectedly that can cause impact through rework. In fact, requirements uncertainty has long been recognized as a major risk factor for software development projects (e.g., [5], [9], [7]). Therefore, it is vital to manage these uncertainties through the project life and to either eliminate the uncertainty or allow for it in cost-effective ways.

Requirements uncertainty has also been identified as a key ingredient in the strategic planning of transition towards product lines [14]. Software product lines (SPLs) to date have been promoted for their improvement in cost, time to delivery, and quality [4], [13]. SPLs are also supposed to increase the complexity of management, process, and communication overhead [3].

However, in this paper, we show how SPL can actually relieve technical management of requirements uncertainty. It is tempting to believe that uncertainty cannot be identified because, after all, it’s uncertain. However, from studies conducted at Rolls-Royce, we have shown that for a traditional (non-product line) project, over 80% of requirements uncertainty can be anticipated at project launch making uncertainty a factor that can be anticipated and managed.

This paper presents an experience report on the application of risk analysis techniques used at Rolls-Royce for managing requirements uncertainty and shows how this technique can be used with SPLs. The technique will help architects better define the scope of product lines by analysing not only the risks associated with the known variations but also the risks associated with the less probable variations in a product line.

Measurements show that the analyses technique described in this paper adds around 1 minute extra effort for analysing each requirement but has shown to reduce Scrap & Rework (work that is either thrown away (Scraped) or reworked which has an impact on cost) on a traditional (non product line) project from an average of 50% to below 5%. The return on investment for uncertainty analysis and mitigation can be between 100:1 and 500:1, making it one of the most cost-effective improvements a project can apply.

This paper is organized as follows. Section II presents the context in which the technique for requirements uncertainty analysis presented here has been developed. Section III provides an explanation about the importance of managing requirements uncertainty at Rolls-Royce. Section IV presents the principles of the requirements uncertainty analysis technique as well as some observations gathering from its use. This section also presents four attributes used for predicting the uncertainty of a requirement. Section V describes how SPL principles can be used to mitigate some sources of risk. Finally, section VI presents conclusions and future work.

II. CONTEXT: ROLLS-ROYCE ENGINE CONTROL SYSTEMS

Although we believe the results in this paper are broadly applicable, the supporting data for our conclusions was gathered in a specific context, which is described in this section.

The Control Systems department at Rolls-Royce is responsible for the Engine Electronic Controllers (EECs) for a range of small and large gas turbine engines for the
aerospace industry. The EEC contains a significant amount of software that is designed to control the engine, as directed by the pilot, in a way that is safe for the engine, safe for the aircraft, fuel-efficient, component life efficient, and environmentally efficient. EEC software is developed to DO-178B Level-A standards for safety-critical software.

The development of a new engine can take up to 5 years and will be highly evolutionary. The electronics, the engine, and the airframe will evolve and mature through the life of a project causing new functionality and changes to emerge. Historical data shows that between the point of Critical Design Review (a system concept review gate) and Entry into Service, a project will spend approximately 50% of its cost on evolutionary work rather than new product development. The evolutionary work will arise in the form of formal change requests raised either by customers or by the Control Systems department. Dealing with changing requirements and uncertainty is therefore an important issue.

III. AN OVERVIEW OF REQUIREMENTS UNCERTAINTY AT ROLLS-ROYCE

A Six Sigma project at Rolls-Royce intended to improve estimation capability found that requirements uncertainty and scope creep were ranked as the two most dominant reasons for poor estimation, which shows up as overspending relative to the initial estimate.

Data at Rolls-Royce shows that uncertainty changes over time, not just in magnitude but also in form. In general, at the start of a project there is more uncertainty than near the end – but not always. Even late into a project’s life there are residual uncertainties that manifest as surprises and late changes.

Fig. 1 shows the percentage of requirements change at key project phases. The shaded area represents the full measured range of requirements uncertainty a project can expect at each milestone. The data is based on 10 aerospace projects and contains both systems and software requirements.

Fig. 2 is based on data from Rolls-Royce and shows the average relative cost to make a software change at various phases of a project. All costs are shown relative to the cost to find and remove an error during the review process. A similar chart for hardware design shows an order of magnitude change in cost with each key project phase.

By combining Fig. 1 and Fig. 2 we get Fig. 3 which shows that unmitigated uncertainty causes an average net impact of 50% of the cost of a project, in the form of Scrap & Rework.

Fig. 3. Percentage of Scrap & Rework for a range of Engine Controllers over the last 10 years.
Whereas Fig. 3 represents the costs for a traditional project, how might this chart look for a SPL? If the requirements change can be accommodated by a built-in variation mechanism of a core asset, the change may look more like the green line on Fig. 4. If the core assets and architecture do not lend themselves to the change and there is wide spread impact too many projects then we might see a cost closer to the red line. How this differs from a traditional project will depend on the extent of the change and if it is special case (limited to a single engine) or common (affecting all engines).

Fig. 4 shows the impact for failing to accommodate (mitigate) uncertainties in both requirements and implementation within a project. An SPL asset is designed to accommodate cross project variation and this paper proposes that SPL assets can therefore more effectible cope with (mitigate) changes in requirements that lie within the assets variation mechanism. This paper proposes that the solutions required to accommodate cross project variation can also be used to mitigate many aspects of inter-project uncertainty. These concepts have been used on software development for many years. Architects will anticipate points of growth or variation and will develop functionality to accommodate. For example, auto-code generators or data configurable functions have been in use at Rolls-Royce for nearly 20 years to accommodate uncertainties in engine design and aircraft interfaces. SPL makes more explicit the analysis of variation and building assets to accommodate an anticipated level of variation in the requirements.

The exact costs of change have yet to be measured but we hypothesize that SPL principles should help reduce the impact of requirements uncertainty for both traditional projects and SPLs.

B. Managing Uncertainty

It is tempting to believe that most uncertainty is driven by factors outside the project’s control; for example, changes introduced by the customer. However, an analysis of the source of change for engine control systems (all civil and defence projects for the last 16 years) shows that only 16% of change is driven by the customer. Also, of the 16% customer driven changes, a proportion could have been anticipated by either Rolls-Royce or by the customers themselves. This is an interesting observation for SPL. It may be tempting to develop functionality to accommodate the known, visible requirements and to play victim to unexpected changes. However, the analysis from Rolls-Royce has shown that most uncertainty can be anticipated – if you invest enough effort to find it and talk to the right people.

If 16% of change is driven by the customer, this means that 84% of changes were self-generated. Bell Labs and IBM conducted studies that determined that 80% of all product defects were incorporated during the requirement definition phase [8]. This is in line with the results obtained by Mawby and Stuples [11] who found that project overruns are usually caused by rework generated within the project.

If an organisation has a higher level of unexpected customer changes, that may be a sign that it has not done its systems engineering and requirements gathering correctly. Being close to the customer can help anticipate where requirements are likely to change or new requirements appear.

Technical Risk Management [12] is one of the methods that can be used by project managers to identify and potentially mitigate the impact of requirements uncertainty. Given that there is a high level of uncertainty in a Rolls-Royce project and that the effects of this uncertainty causes up to 50% Scrap & Rework, we would expect to see this expressed in the risk logs. An analysis of the risk logs in 2006 for a range of projects across the business showed that less than 4% of risks were technical in nature. The remaining 96% of the risks tended to focus on project and business risks.

Further analysis showed an absence of risk identification and mitigation processes used by the engineers. The engineers were not encouraged to identify and express their uncertainty – there was an implicit message in the business that engineers must always be certain, and that uncertainty was a sign of failure rather than a normal and expected occurrence to be managed.

A range of projects was studied where the projects all had the same maturity (all their products were in service). They had approximately the same processes, the same team, and each project underwent a modification of a similar size. In the case where the projects did not apply any Technical Risk Management to understand and anticipate uncertainty, they experienced a “normal” 50% Scrap & Rework rate. Those projects that actively apply Technical Risk Management (to requirements and design) had a Scrap & Rework rate below 5%. The cost benefit from Technical Risk Management was 100:1. A similar study in hardware engineering revealed a return on investment of 500:1.

The analysis from Rolls-Royce showed that one of the biggest root causes for Scrap & Rework, at the time, was that we did not manage our uncertainty.
IV. THE PRINCIPLES OF REQUIREMENTS UNCERTAINTY ANALYSIS AND SOME OBSERVATIONS

A. Uncertainty Identification

The risk analysis technique (described in section IV-D) involves applying Technical Risk Management principles to each individual requirement (or logical group of requirements) to understand the uncertainty in the customer (and supplier) requirements in order to control and minimize the cost of late changes. The technique involves the use of the standard risk principles by assigning probability and impact values to each requirement, such as L=Low, M=Medium, and H=High.

The analysis is performed initially by the systems engineer then reviewed independently by the requirement reviewer. The probability of change is estimated by the systems engineer (based on subjective experience, data from past projects, and his/her understanding of the system). The impact of change can be estimated by the designers and implementers of the system but Rolls-Royce tends to let experienced systems engineers do this. In the case of SPL, this analysis should be performed during domain analysis.

Fig. 5 presents an example of a probability impact diagram showing the results of the uncertainty analysis for a project’s requirements. The diagram shows the net uncertainty for each system function of an engine controller, but could just as well show the results per requirement. The colour coding conveys the impact to a project: green has a low impact, amber has some impact, and red has the greatest impact.

![Figure 5. Probability impact diagram showing the volatile function. Each function shown is the aggregate of the uncertainty and impact for each requirement within that function.](image)

B. Mitigating the Uncertainty

The objective of a systems engineer is to move a function (or requirement) out of the top right area (red) by reducing its uncertainty (moving it to the left across the diagram in Fig. 5), minimising the impact (moving it down the diagram in Fig. 5) or a combination of both approaches. The selection of a particular approach will depend on the type of risk (uncertainty) the function is exposed to. Rolls-Royce catalogs recurring risks and their mitigations. For example, the risk of “Immature or late customer requirements” has a number of mitigation options.

- More customer engagement
- Does the customer have a "design style" - can you predict the requirements from previous projects working with this customer?
- Get the customer to do a Technical Risk assessment
- Propose requirements to the customer
- Look at other projects – what did they have
- Build robustness into the architecture
- Delay- wait for the requirements
- Proceed but factor late change into plans & budgets
- Model system and present to the customer
- Find the range outside which it will hurt

It is not always necessary to remove the uncertainty (risk). It is possible to develop products that are “robust” to accommodate a reasonable level of uncertainty - which is common practice in SPL. This amounts to moving a function down the diagram in Fig. 5. For example, the performance data used in the control laws of an engine is separated from the functionality in the design of the EEC, allowing the performance engineers to evolve the engine parameters with minimal impact to the software.

One technique to reduce the uncertainty of a requirement (moving a function to the left in the diagram in Fig. 5) is to “soften” the requirement. For example, a “single-point” requirement may say that a temperature probe for an engine must operate at 1000 degrees. This requirement is precise but almost certain to change. The requirement can be expressed as a range; for example, “the probe must operate at temperatures in the range of 900 to 1200 degrees; or “the probe must operate at temperatures over 900 degrees,” and so on. In these cases, by relaxing the precision of a requirement, we will reduce the uncertainty (probability that the real requirement will lie outside the range). In other words, we can cope with the uncertainty by being certain about our uncertainty and expressing this in a way to allow designers to develop a solution to meet the required range. For a core asset designer, such a requirement is a clear signal that a variation point is needed.

In those cases where there is uncertainty but no clear mitigation available, the systems and software architecture can bound the requirements (or function) and isolate it from the rest of the system until the requirements become more mature. The SPL technique of “component replacement” (replacing a component with one that satisfies an interface but provides a specifically desired behaviour or quality attribute via its implementation) may be usable once the requirement is available.

Requirement uncertainty analysis can be used to help influence the planning process. In the example of a high uncertainty function, a project can ask for time to help mature the requirements before proceeding with development. Once the requirements uncertainty attributes (described in section IV-D) are calibrated to a project domain, the information can be used to estimate Scrap & Rework, the impact of late changes, the contingency a project must carry, and so on.
C. Risk / Mitigation Tradeoff Analysis

Risk to a project is calculated as probability a risk will occur multiplied by the impact that the risk manifest. Once a risk has been identified, the systems engineer (or domain expert) identifies the mitigation to be used. If the cost of the mitigation exceeds the cost of the risk (probability * Impact), then a project should not mitigate the risk. For example, the pressures required for an engine actuator may not be known at the start of a project. The cost to develop a variable actuator may add too much weight and cost. It may actually be better to redesign the actuator once the engine characteristics are understood.

D. Requirements Uncertainty Attributes – the method

This section describes the four attributes used at Rolls-Royce for predicting the uncertainty of a requirement and its impact to the project. Although we apply these to each requirement, they could be applied to a group of requirements such as for a system function.

The following attributes are added to our requirements database:

- **Volatility** represents the engineer’s best judgment as to the probability the requirement will change through the course of the project. This is based on subjective judgement, past data and knowledge of the system under development. This attribute was also suggested in other studies, e.g., Lam et al. [10], who manage volatility by process control, suggest that volatility classification should capture the domain-specific nature of change in order to facilitate change estimation and reuse.

- **Impact** accounts for the degree that a change in the requirement will negatively affect a development program. Rather than the systems engineer selecting this attribute, it may be better for the developers of the system to “estimate” the impact.

- **Precedence** incorporates multiple variables to indicate Rolls-Royce’s heritage in providing solutions that address this attribute for a similar application, environment, and context of use. Experience from within Rolls-Royce has shown that even if you cannot anticipate the risk, if it is a novel system requirement, there will be a lot of evolution required to mature it.

- **Time Criticality** provides sensitivity to time critical requirements. This compliments the prioritization of work to allow the engineer to work on items critical for the current phase as opposed to items not required until later phases of the project. This attribute challenges the view that all requirements do (and have to) mature at the same rate. This attribute informs an architect that they need to isolate the impact of this late maturing requirement.

E. Uncertainty Analysis

Based on the above attributes, the following uncertainty analyses can be performed:

- **Risk Index (RI)**: For technical risk evaluation, risk index collects the volatility, impact, time criticality, and precedence values into a number that illustrates the overall risk of each requirement to the function and subsequently the subsystem and program. Each requirement should have an associated RI value. These numbers can then be averaged to produce a function RI value between 1 and 9.

\[
RI = \frac{V * I * P * TC}{729}
\]

where,

\[V = \text{Volatility score (1 to 9)}\]
\[I = \text{Impact score (1 to 9)}\]
\[P = \text{Precedence score (1 to 9)}\]
\[TC = \text{Time Criticality score (1 to 9)}\]
729 = 9 * 9 * 9 which returns a RI in the range 1 - 9

- **Maturity Index (MI):** The maturity index is a value associated with a system function, not with each individual requirement. A function that is 100% mature is defined as mature and free from any further anticipated changes.

\[
MI = \frac{\sum (V \times TC)}{\text{Total # Requirements} \times 81} \times 100 \tag{2}
\]

where,
- \(V\) = Volatility score
- \(TC\) = Time Criticality score
- \(81 = 9 \times 9\) (the maximum score for V and TC giving a value for M in the range 0 – 1)

and the sum is for the product of these factors for all of the requirements in the document.

- **Proportional Risk Index (PRI):** For an understanding of the risk of an individual requirement, the proportional risk index is recommended. PRI is intended to prioritize the engineer’s tasks in regard to the document.

\[
PRI = \frac{R_I}{\sum R_I} \tag{3}
\]

**F. Example Analysis Results**

This section illustrates some examples from the application of requirements uncertainty analysis and applying the equations shown in the previous section.

- **Plotting Proportional Risk:** Fig. 10 shows an example of plotting requirements risk. Requirements system function for a single project are listed across the X-axis of the chart and the uncertainty of each on the Y-axis. The systems engineer could annotate this chart as shown with bands of uncertainty to present their risks to project management.

- **Reporting Requirement Maturity Index.** The Maturity Index (Fig. 11) was calculated for a range of system functions. It shows that there are some functions which can be started at little risk and other functions requiring more Technical Risk Management to improve their overall uncertainty. Alternatively, the maturity index can be tracked over time and plotted as shown in Fig. 12.

**G. The effort to Apply Technical Risk Management to Requirements Uncertainty**

It may seem daunting to apply requirements uncertainty analysis to a project that may have thousands of requirements. Four years of using this technique has shown that it will take an engineer between 1 and 2 minutes (but typically around 1 minute) per requirement to apply (not counting the additional effort required to mitigate the uncertainty identified).

When we performed an experiment to test the concepts, we applied the technique to the hardware requirements of an existing engine (a double blind experiment), the results show that the engineer was able to correctly identify 80% of the uncertainty and that the risk identification and mitigation approach would have yielded a return on investment of 500:1. Other studies in software have shown that when used by experts, 90% of the uncertainty can be identified and mitigated with a 100:1 return on investment.
H. Lessons Learned

When performing requirement uncertainty analysis on traditional projects, we found that the high-risk requirements were often already recognized; this analysis did not help identify these issues. A good engineer will typically know the problem areas and will have flagged these issues to the project.

Interestingly, the sum of the medium-risk requirements added up to 80% of the uncertainty. That means that overall, the high-risk requirements had an impact on a project but not nearly as much as all the smaller risks added together. If project management is only interested in the high-risk requirements (as is often the case) then 80% of the uncertainty (and source of Scrap & Rework) will go ignored by the project. This is consistent with the findings earlier in this paper in which we showed that 84% of the Scrap & Rework is self generated and also that only 4% of risks registered on the risk log were technical.

This technique of requirements uncertainty analysis helps to identify both high-risk and medium-risk requirements, which can then be managed and mitigated using Technical Risk Management techniques.

V. MANAGING REQUIREMENTS UNCERTAINTY IN SPLS

A. Introduction

This section describes a classification of the sources of change used for a traditional project development and explains how it applies to software product lines.

Fig. 13 shows the classification of the sources of change for a traditional project development at Rolls-Royce. It classifies the change into four types of risk:

- The Known-Known quadrant represents known functionality but with defects during implementation, which are typically mitigated through Verification & Validation. These issues will also arise in SPL.
- The Known-Unknown quadrant shows changes caused by known risks that we did not mitigate resulting in late changes. In an SPL, this quadrant expresses the risks over functionality that may be required but were not accommodated in the core asset variation mechanisms.
- The Unknown-Known quadrant referred to as “corporate memory loss” and represents changes that manifest that could have been produced if we had involved the right domain experts or historic lessons. A SPL is designed to capture the corporate memory in its core assets and therefore is intended to mitigate this quadrant.
- The Unknown-Unknown quadrant represents the changes that appear as surprises i.e. emergent behaviour that arise during integration or unforeseen customer requests. An analysis of this quadrant showed that although there risks were unforeseen by the development team, many of these changes were Known-Unknown (risks) to someone else and could have been identified and moved into the Known-Known or Known-Unknown quadrants.

<table>
<thead>
<tr>
<th>Known</th>
<th>Unknowns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known knowns (28% of uncertainty arises here). We failed on implementation.</td>
<td>Known unknowns (29% of uncertainty arises here). We know we have risk.</td>
</tr>
<tr>
<td>Unknown knowns (30% of uncertainty arises here). We knew but forgot.</td>
<td>Unknown unknowns (13% of uncertainty arise here). Surprises</td>
</tr>
</tbody>
</table>

Figure 13. The four main sources of uncertainty

The role of SPL and uncertainty analysis can be understood in the context of this diagram. Domain Analysis is intended to move any uncertainty from the bottom row (Unknown-X) into the upper row (Known-X).

Uncertainty analysis then complements domain analysis by classifying and mitigating (when cost effective to do so) any risks in the Known-Unknown quadrant i.e., to move uncertainty from the X-Unknown column to the X-Known column.

The combination of domain analysis and uncertainty analysis is therefore intended to migrate uncertainties into the Known-Known quadrant, where the risks are accommodated through variation in the core assets. However, it is not always practical to reduce the uncertainty in the three unknown quadrants to zero as the mitigation effort may be higher than the expected cost of the risk. At Rolls-Royce the decision point to implement is when the mitigation effort is less that 1/3rd the expected cost of the risk. Although Fig. 14 shows an exponential mitigation cost, the mitigation may be:

- Flat (no cost increase). For example, extending the range of a library function.
- Linear (as we add more functionality, the cost grows consistently). For example, adding more decision points to the implementation
- Stepped (there are stepped cost increases as we switch between mitigation solutions). For example, providing more component replacement options.

Figure 14. An illustration of the trade-off between risk reduction and mitigation effort. Although this diagram shows an exponential curve for the mitigation and risk, the reality may have different curves depending on the sensitivity or risk and mitigation effort.
B. Identifying the Scope of an SPL

This section describes how the principles of requirements uncertainty analysis can be used to help identify the scope (and boundary) of an SPL. Consider Fig. 15 below. The box represents the full range of the domain, the green circles represent known projects within the domain and the amber shaped area represents the scope of a SPL. This paper proposes a technique for identifying the most economical shape (and size) for the amber area – the scope of the SPL.

Fig. 16 shows some scenarios of the SPL boundary relative to known projects. If the scope is defined too far away from any known requirement (example B in Fig. 16) i.e., there is more functionality then required, then the project will incur a cost with no benefit. In safety-critical software this is a very expensive option.

If the scope is set too close to the known requirements (example A in Fig. 16), the project runs a risk of needing to make costly modifications late in the product line life. However, in safety-critical software, with the certification activities, it is tempting to define the boundary of the SPL as close as possible to the known functionality required.

To determine the “sweet spot” for the scope of the SPL (example C in Fig. 16), the principle of risk/mitigation trade-off analysis can be used. The “sweet spot” can be calculated as the ratio between the impact to a SPL from a new emergent requirement (defined here as risk) and the costs to mitigate this risk through the initial core asset design (mitigation).

Figure 17. A theoretical cost-benefit trade between risk and mitigation costs. The figure shows a theoretical functional “sweet spot” when the business achieves maximum return-on-investment.

There are some cases when it is economical to move from scenario B (Fig. 17) to scenario A. For example, the “starting” function of an engine can be very specific to an engine type and the airframe. The range of functionality and options is so large (scenario B) that it is economical to create several smaller core assets (scenario C) or to develop bespoke solutions for each engine/customer combination (scenario A).

The boundary of any SPL functionality should be based on both anticipated requirements changes and business opportunities that may exist beyond the boundary. Implicit within requirements uncertainty analysis is sensitivity analysis. There may be some requirements that can be extended with little or no mitigation costs; whereas, there may be requirements which have a high sensitivity to change or discontinuities in the impact. For example, in a factory that builds cars, the painting facility could be extended to accommodate 6 colours rather than 5 with little or no mitigation effort. But to extend the production plant to accommodate trucks as well as cars may be a significant impact. The risk analysis proposed in this paper can accommodate both types of risk.

As an example, consider the selection of a temperature probe used on an engine. The operating temperature of an engine may not be known precisely at the start of a project. Different probes can be used dependent on the temperature required but some probes, those for higher temperatures, can be heavier and more costly. However, a more expensive probe can meet a wider range of engine temperatures. There is a direct trade between flexibility and net cost to the business. Risk analysis can be used to determine the probability the engine temperature will exceed a probes capability and, therefore, if it is better to adopt a more expensive probe now, or later when the true temperatures are determined.

As late changes can be expensive to an engine and to an SPL, these analysis techniques are intended to determine if it is better to take a risk and develop the functionality now, or to wait and develop the functionality later when the impact
of change is higher. If the risk (probability or impact) is high enough, it may be better to develop the capability up front. If the risk is low enough, it may be better to wait for it to manifest.

VI. SUMMARY

This paper presented the results from an extensive Six Sigma study into requirements uncertainty conducted at Rolls-Royce and shows how the technique can be used for managing requirements uncertainty of a software product line.

The main conclusions of the study are as follows:

- Uncertainty is certain but if no effort is made to control uncertainty, then it will manifest later as Scrap & Rework.
- Most Scrap & Rework (or a significant amount of it) can be self-generated by not managing your inherent uncertainty.
- Contrary to expectations, changes in customer requirements are not a major driver of scrap and rework - most is internally generated by the development team.
- Systems Engineering and Risk Management are critical in understanding and controlling the sources of Scrap & Rework.
- Requirements uncertainty analysis and risk/mitigation trade-off can help a software product line: (1) to find the economical boundary (sweet spot) of the SPL functionality and (2) to minimise the functionality required whilst also minimising the risk for future updates to an SPL as new requirements emerge.
- SPL principles can be used to mitigate 3 out of the 4 main sources of uncertainty i.e., known-unknowns (risks), unknown-known’s (capturing domain knowledge in the SPL), and unknown-unknowns (looking for anticipating or accommodating uncertainties when economical to do so).
- The return on investment for the use of Requirements Uncertainty Analysis can be between 100:1 and 500:1.
- The SPL principles used to accommodate variation across projects is the same technique that can mitigate uncertainty within a project. It is therefore proposed that the return-on-investment of applying uncertainty analysis will also apply to software product lines.

REFERENCES

[16] Software Considerations in Airborne Systems and Equipment Certification, EUROCAE ED-12B/RTCA/DO-178B.