On How to Deal with Uncertainty when Architecting Embedded Software and Systems

Jakob Axelsson

School of Innovation, Design and Engineering, Mälardalen University,
SE-721 23 Västerås, Sweden
jakob.axelsson@mdh.se

Abstract. This paper discusses the topic of uncertainty in the context of architecting embedded software and systems. It presents links between complexity and uncertainty, and identifies different kinds of uncertainty. Based on this, it elaborates why uncertainty arises in the architecting of software-intensive systems, and presents ten different tactics that can be employed to deal with uncertainty and mitigate the associated risks.

1 Introduction

In many companies developing technical products, embedded systems and software play an increasingly important role. From being a small and isolated electronics-based part of a product, the embedded system has developed into a large number of computers with distribution networks and millions of lines of software. This increasing complexity leads to soaring developing costs, and many companies strive to curb this trend by reusing software and hardware between products. Often, a product line approach is applied, where the same platform is used as a basis, with modifications to fit the needs of the individual products and customers.

With a multiplicity of products and variants, the architecture is becoming very important and is a source of increasing interest for companies developing embedded systems. When developing a new platform for embedded systems, much of the architecting occurs 2-4 years before the first product is delivered. At that time, a number of key factors are unknown or known only with large uncertainty, such as the details of the software code, including its structure or execution time; the details of the hardware if new components are to be used (no prototypes available, only specifications); and the exact requirements (only an early estimate is available).

The complexity of these already complex systems increases as a result of this uncertainty, since it leaves open a larger space of possible design alternatives that the architects must consider. Many architects rarely use the tools and methods proposed by academic researchers even if they are aware of them. We have come to suspect that one reason is that many methods and tools are difficult or expensive to use in a situation where only imperfect information is available. In this paper, we therefore investigate the nature of the uncertainty in the architecting situation, and provide some ideas for how this uncertainty can be dealt with.

2 Types of Uncertainty

A development team's responsibility is to make decisions about the product. *Uncertainty* can be defined as a lack of necessary knowledge to make a decision [1]. This should be compared to *risk*, which is the probability that an unfavorable event occurs (i.e., in this context making the wrong decision). The concept of risk thus assigns a utility or value to an event, whereas uncertainty is free from such valuation. "Uncertainty causes risk which is handled by mitigation and results in outcomes" [2].

Two kinds of uncertainties are very common in engineering. One is *imprecision,* i.e. the exact value of a parameter is unknown (but a range or approximation is known). This typically occurs as a result of insufficient measurement equipment. The other common type is *variability* (or *aleatory* uncertainty) which denotes variations in parameter values between different instances due to a random factor [1].

Another dimension of uncertainty relates to what can be done about it. Sometimes there is a *reducible* uncertainty (alternatively referred to as *epistemic* uncertainty). This is an uncertainty that can be removed or at least reduced by spending an effort in collecting more information. However, there is always a balance if the effort of finding out merits the value of the added knowledge. The opposite is called *irreducible* uncertainty that cannot possibly be discovered [1]. As an example, it is an irreducible question if it will rain on a certain spot on Earth tomorrow (only prognoses are possible), but it is a reducible question if it rained there yesterday.

Variability is often described by statistical distributions, capturing properties of a large population of objects. This is sometimes referred to as the *frequentist* interpretation of probabilities. However, sometimes distributions are also used to describe one's beliefs in some event, such as our certainty that it will rain tomorrow. This is called *subjective* probabilities, and is useful in situations where statistics is meaningless (such as when talking about unique events rather than a large number of similar events). Many tools for statistical analysis that are based on probability distributions are also useful to reason about subjective beliefs.

So far, we have discussed uncertainties in the value of a known factor. Another level of uncertainty is whether all the relevant factors have actually been found, or if there are yet unknown factors that should also be considered. An example of this is the uncertainty if all relevant requirements have actually been elicited. Such factors are sometimes referred to as *unknown unknowns*.

3 Causes for Uncertainty

Architecting is characterized by many parallel activities and only few firm decisions. There are a number of factors that contribute to a certain level of inherent uncertainty. Some of them are caused by the nature of the development process and methods:

- *Sequential decision making* where the first decisions must be made under uncertainty regarding the consequences on later decisions.
- *Parallel decision making*. Several teams are refining their systems in parallel, and they must all make decisions based on uncertain knowledge about what decisions the neighboring teams will make in the future.

- *Expert judgment*. Sometimes, decisions need to be made based on experts giving imprecise statements such as "large", "fast", or "unlikely".
- *Abstract models*. It is a common engineering practice to build models of a future system. However, such models are always simplifications of reality, and factors that are believed to have minor influence are often removed.

Other uncertainties are due to the nature of the products. Three categories of noise, i.e. variability that cause quality loss, have been identified:

- *Environment*, or *outer noise,* includes changes to the environment, operating conditions, and effects of the system's interactions with different people. For architecting, only limited statistical data is available about the environment due to finite sampling. The environment available for investigation at the time of decision may vary from the one at the time of deployment.
- *Aging*, or *inner noise*. For physical products, aging leads to wear and deterioration, and this is true for the physical parts of the embedded system. For the software, architects may need to consider changes done over the system's installed life that can lead to variations.
- *Manufacturing*, or *product, noise*. Again, this is evident for the physical parts, and often there is a need to also include mechanisms in software (such as calibration or feedback) to control the effects of part-to-part variations in material and dimensions that are hard, or too expensive, to fix.

All these causes of uncertainty are general for engineering of complex systems, and are particularly strong during architecting. Two special cases of environment noise are of particular interest for architects. When the system of interest is a subsystem in a larger product, as is often the case for software, unintended interaction with neighboring subsystems often take place and cause noise that falls in this category. Also, uncertainties about the market for the products spill over on the architects, such as what functionality and variants will be requested by different customers.

4 Mitigation

We will now discuss a number of "tactics" that can be employed to reduce the unavoidable uncertainties and mitigate risks in architecting.

- *Focus on evolution rather than revolution*. When a new system is developed from scratch, everything is uncertain and remains so for a long time. By instead focusing on step-wise evolution of the architecture, only a few unknowns at a time must be dealt with, and effects are quickly seen.
- *Create feedback loops*. In architecting, feedback from downstream engineering is needed to quickly see when estimates prove to be wrong.
- *Track quality attributes*. Many architects focus on functionality, and have little data on actual values on the architecture's quality attributes. By

continuously measuring attributes, it becomes possible to base decisions on trends and extrapolation from real data on things like cost and performance.

- *Make uncertainty explicit*. Often, engineers use point data in their estimates. By instead describing estimates using probability distributions, analyses such as Monte Carlo simulation can provide a richer picture of alternative scenarios, complemented with sensitivity analyses.
- *Divide and conquer using tolerances*. To ensure that the desired state of the system-wide quality attributes are reached during subsystem development, tolerances for attributes should be made explicit for each subsystem. The tolerances can give architects early warnings of problems later on.
- *Perform cost-benefit analysis of added certainty*. Usually, reducing uncertainty comes at a cost for performing additional analysis, and it should be considered whether buying more information in this way really changes the decisions being made.
- *Use options thinking*. A difficult trade-off for architects is between product cost now, and future flexibility, and real options can be used for this.
- *Apply robust design principles*. In situations where it is not practical to reduce uncertainty, the solution should be protected through robustness.
- *Use wide sampling and saturation*. Dealing with unknown unknowns is a particular challenge. One remedy is to gather information from a wide range of sources. Saturation should be tracked over time to see when the rate of new information slows down, and investigation can stop.

5 Conclusions

As we see it, uncertainty is a fact of life for architects of complex systems. However, we also believe that a deeper understanding of the nature of this uncertainty can help reducing the associated risks and improving the efficiency of the architecting process.

Apart from new evaluation methods and models that take uncertainty into account, there is also a need to review the current practices when it comes to the architecting process and the role of the architect. Too much effort is often spent on trying to get better information about events that are by necessity occurring in the future, and this search is usually fruitless. Instead, more focus should be placed on quality attributes and their relations, and finding architectures that can handle various scenarios.

It is also important to always bear in mind when looking for a solution that all additions to the architect’s work come at a cost, and future research must focus on evaluating that the benefit is really motivating this additional spending.

References

1. Aughenbaugh, J. Managing uncertainty in engineering design using imprecise probabilities and principles of information economics. PhD thesis, Georgia Inst. of Tech., August 2006.
2. McManus, H. and Hastings, D. A Framework for Understanding Uncertainty and Its Mitigation and Exploitation in Complex Systems. In Proc. 15th Symposium of the International Council on Systems Engineering (INCOSE), July 2005.