A Model for Prioritization of Software Architecture Effort

Eoin Woods[1][0000-0001-7858-1552] and Rabih Bashroush[2][0000-0002-9610-0027]

[1] Endava Limited, 125 Old Broad Street, London, EC2N 1AR, UK
eoin.woods@endava.com
[2] University of East London, University Way, London E16 2RD, UK
r.bashroush@uel.ac.uk

Abstract. As part of our software architecture research and practice we have found that a common difficulty for new architects is knowing where to focus their effort to maximise their effectiveness. This led us to wonder whether successful experienced architects have reusable heuristics or guidelines that they follow to help them prioritise their work. To investigate this we have performed a study using semi-structured interviews to explore how experienced software architects prioritise their activities in order to maximise their effectiveness. From the primary data collected through the interviews we have synthesised a simple model that organises and explains the heuristics that we found to be common across a number of experienced software architects.

Keywords: software architecture, software architecture decision making, software architect effectiveness.

1 Introduction

In our research and practice in the field of software architecture, we have noticed and experienced how complex it is for software architects to prioritise their work. The software architect's responsibilities are broad and in principle they can be involved in almost any technical aspect of a project from requirements to operational concerns.

However we observe that successful software architects appear to be very good at focusing their effort effectively, which led us to wonder how they achieve this. They may use generic time management techniques (like [1]) but we were interested in commonly used, role-specific, heuristics which could be taught to new architects.

We decided to investigate this via a questionnaire-based study of a group of experienced architects. We discovered that there are common heuristics which experienced architects use to prioritise their work and we have created a model to capture them.

In this paper, we explain the approach we took and present the model that we created from the results that we obtained. The contribution of our work is not specifically the heuristics in our model, indeed most of them are quite familiar to experienced practitioners, but rather the organisation of the heuristics and the validation that they are used by experienced practitioners to guide their work. We believe that this makes the model

potentially useful as a reminder for experienced practitioners and as a teaching aid for new architects who are learning how to fulfil the role.

2 Related Work

When we started investigating this topic, we were primarily interested in how practitioners really worked however we also performed a literature search to find related work from the research community.

We did not find any studies investigating our specific topic, but an architectural method which helps architects to direct their effort is Risk and Cost Driven Architecture (RCDA) [12]. This method transforms the architect's approach from defining finished architectural structures at the start of a project, to use the risk and cost of open decisions to prioritise the architect's work throughout the project. Another practitioner oriented approach that stresses the importance of risk in guiding architecture work is [6]. We were also interested to find some very specific advice from a very experienced architect and researcher [10] that architects should spend 50% of their time on architecting, 25% on inbound communication and 25% on outbound communication.

In the research domain, we found a research community interested in prioritisation of requirements [3, 7], but this only addresses part of an architect's work.

Finally, there is a large amount of mainstream business and self-help literature on time management (such as the well-known [1] and [9] and some more focused on software engineering such as [5]) however we were interested in architecture specific approaches and heuristics rather than more general advice.

3 Research Method

When planning this research, we selected a qualitative research approach because we needed to explore the "lived-experiences" of expert practitioners by asking them questions to encourage reflection and insight [13] rather than assessing performance or alignment with specific practices via quantitative means.

We chose to gather our primary data using semi-structured interviews, where we provided the interviewees with a written introduction to the question we wanted to answer and then some specific questions to start their thought processes.

The analysis of the primary data was performed using a simple application of Grounded Theory as it is a suitable method for theory building, to understand the relationships between abstract concepts [4], which described our situation and needs very closely. We performed initial coding on the primary data and then refined this with a focused coding exercise. As suggested in [13] the process of collection and analysis was a parallel, iterative process, rather than a linear one with fixed phases.

This exercise produced a set of themes that classify the heuristics that the architects use, as well as the heuristics themselves. A heuristic had to be mentioned by at least three of the participants (which represented a third of them) for us to consider it significant enough to be included in the model. We combined the themes and heuristics to form a simple model of how experienced architects go about prioritizing their effort.

4 The Study

Our primary data gathering was performed using a semi-structured, face-to-face survey of 8 experienced software architecture practitioners working across 4 countries.

We found the participants by approaching suitable individuals from our professional networks. We were looking for practitioners who had a minimum of 10 years' professional experience and who worked as architects in the information systems domain (rather than architects from – for example – embedded systems).

We focused on the information systems domain because we know from experience that working practices differ between professional domains like information systems and embedded systems. Hence, we thought it more likely that we could create a useful model if we limited ourselves to one broad domain, at least initially.

We deliberately selected candidates that we knew differed from each other in organisation, specialisation and geography to get a reasonably diverse population and avoid obvious sample bias (we discuss the threat of sample bias further in Section 6).

Some characteristics of the participants in the study are summarised in the graphs in Fig. 1. As can be seen they represent a range of experience, role type and country.

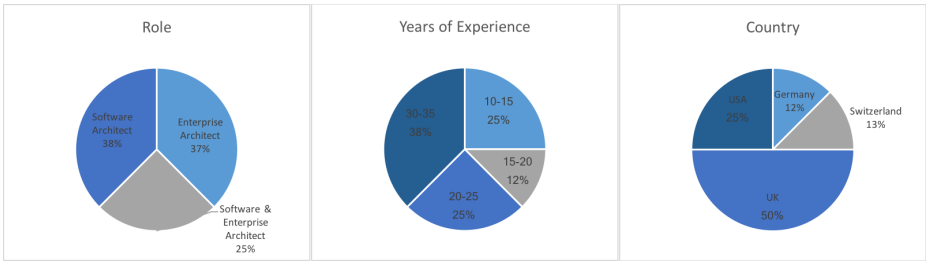

Fig. 1. Study Participants (8 in total)

We used a semi-structured interview format with a written introduction to the question which each interviewee read before being asked a standard set of open ended questions which explored how they went about prioritisation of architecture work and any specific factors that they used to guide them.

The question we asked during the interview was "how can architects concentrate their attention so that they are most effective?" The more specific questions we asked the interviewees to stimulate thought were:

- How do you go about this in your work?
- What factors do you consider when prioritising your attention?
- Do you consider what to focus on? Or what not to focus on?
- For example, how do you prioritise architectural governance compared to other aspects of the project?

The interviewer asked additional questions to understand the answers fully or to encourage the interviewee to add more detail or fill in ambiguous aspects of the answer.

The process of initial coding of the primary data resulted in 25 items, which could be associated with at least one of the interviews. A further focused coding process revealed that there were 9 underlying heuristics which appeared to be significant to the participants in the study and then a further analysis iteration lead to the identification of three categories of prioritisation heuristic which we use to structure our model.

5 A Model for Prioritising Architectural Effort

5.1 The Model

Our heuristic model for focusing architectural effort is shown in Fig. 2.

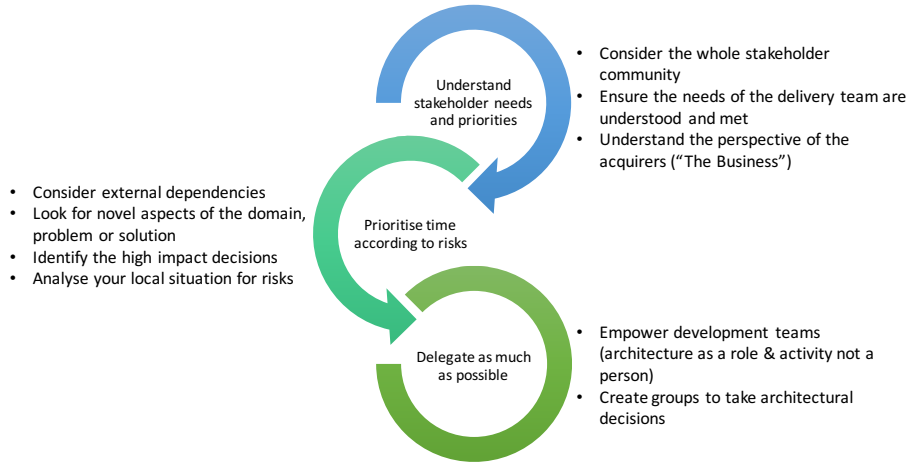

Fig. 2. Model for Focusing Architectural Attention

The three categories of heuristic that the study revealed were firstly, the need to focus on stakeholder needs, secondly, the importance of considering risks when deciding on where to focus effort, and finally the importance of spending time to achieve effective delegation of responsibilities. These categories form the structure of our model, and remind the architect of the general ways in which they should prioritise their efforts. The categories and heuristics are explained in more detail in section 5.2.

It is important to understand the nature of this model and how it should be used. It is not a prescriptive process for architects to follow or a process for developing an architecture. This model is an aide memoire to organise a set of heuristics that experienced practitioners appear to find useful when prioritising their work. While we believe this to be a useful model to teach trainee architects, and a useful reminder for experienced architects, it is necessary to apply the model in a context sensitive manner, within whatever method that the architect is using to develop software architectures.

5.2 Content of the Model

Understand the Stakeholder Needs and Priorities.
The first theme which emerged strongly in our study was focusing on the needs and priorities of the stakeholders involved in the situation. The principle that architecture work involves working closely with stakeholders is widely agreed [2, 14] and this theme reinforces that. Architects need to focus significant effort to make sure that stakeholder needs and priorities are understood to maximise focus on the critical success factors for a project and maximise the chances of its success. Three specific heuristics to achieve this which emerged from the study are:

- Consider the whole stakeholder community. Spend time understanding the different groups in the stakeholder community and avoid the mistake of just considering obvious stakeholder groups like end-users, acquirers and the development team. As the architecture methods referenced above note, ignoring important stakeholders (like operational staff or auditors) can prevent the project meeting its goals and cause significant problems on the path to production operation.
- Ensure that the needs of the delivery team are understood and met. Spend sufficient time to ensure that the delivery team can be effective. What is the team good at? What does it know? What does it not know? What skill and knowledge gaps does it have? These areas need attention early in the project so that architecture work avoids risks caused by the capabilities of the team and that time is taken to support and develop the the team to address significant weaknesses.
- Understand the perspective and perceptions of the acquirers of the system. Acquirers are a key stakeholder group who judge its success and usually have strategic and budgetary control, so can halt the project before delivery if they are unhappy. Specifically addressing this group's needs, perceptions and concerns emerged as an important factor for some of the experienced architects in our study. Acquirers are often distant from the day-to-day reality of a project and need clear communication to understand their concerns and ensure that they have a realistic view of the project.

Prioritise Effort According to Risks (Driven by Impact x Probability).
During a project, an effective approach to prioritising architectural attention is to use a risk driven approach to identify the most important tasks. If the significant risks are understood and mitigated then enough architecture work has probably been completed. If significant risks are open then more architecture work is needed. The specific heuristics to consider for risk assessment are:

- Consider external dependencies. Understand your external dependencies because you have little control over them and they need architectural attention early in the project and whenever things change.
- Look for novel aspects of domain, problem and solution. Another useful heuristic, from the experience of our study participants, is to focus on novelty in your project. What is unfamiliar? What problems have you not solved before? Which technology is unproven? The answers to these questions highlight risks and the participants in our study used them to direct their effort to the most important risks to address.

- Identify the high impact decisions. Prioritise architecture work that will help to mitigate risks where many people would be affected by a problem (e.g. problems with the development environment or problems that will prevent effective operation) or where the risk could endanger the programme (e.g. missing regulatory constraints).
- Analyse your local situation for risks. Consider the local factors unique to your situation, which you will be aware of due to the knowledge you have of the domain, problem and solution. It is impossible to give more specific guidance on this heuristic as every situation is different, but the participants in our study noted the importance of "situational awareness" [15] that allows the architect to find and address the risks specific to the local environment (perhaps due to organisational factors, specific technical challenges, domain complexities or business constraints).

Delegate as Much as Possible.

Delegation was an unexpected theme that emerged in the study. The architects who mentioned this theme viewed themselves as a potential bottleneck in a project and delegation and empowerment of others was a way to minimize this. Delegation was also seen as a way of freeing the architect to focus on the aspects of the project that they had to focus on rather than all the other aspects that they could possibly get involved in.

The general message of this theme is to delegate as much architecture work as possible to the person or group best suited to perform it, to prevent individuals becoming project bottlenecks, allow architects to spend more time on risk identification and mitigation, and to spread architectural knowledge through the organisation. The heuristics that were identified to help achieve this are:

- Empower the development teams. To allow delegation and work sharing, architects need to empower (and trust) the teams that they work with. This allows governance to become a shared responsibility and architecture to be viewed as an activity rather than something that is only performed by one person or a small group. This causes architectural knowledge, effort and accountability to be spread across the organisation, creates shared ownership, reduces the load on any one individual and prevents reliance on a single individual from delaying progress.
- Create groups to take architectural responsibilities. A related heuristic is to formalise delegation somewhat and create groups of people to be accountable for specific aspects of architectural work. For example, in a large development programme, an architecture review board can be created to review and approve significant architectural decisions. Such a group can involve a wide range of expertise from across the programme and beyond, so freeing a lead architect from much of the effort involved in gathering and understanding the details of key decisions, while maintaining effective oversight to allow risks to be controlled and technical coherence maintained. Similarly, a specific group of individuals could be responsible for resilience and disaster recovery for a large programme, allowing them to specialise and focus on this complex area, and allowing a lead architect to confidently delegate to them, knowing that they will have the focus and expertise to address this aspect of the architecture.

6 Threats to Validity

There are potential limitations to any qualitative study, including our work, and there are potential threats to its validity. Although we do not believe that any of these seriously threaten the usefulness of our study, it is important to acknowledge them.

There are four main types of threat to the validity of a study like this, namely construct, internal, external and conclusion validity as defined in [11].

Construct and *internal* validity relate to the effectiveness and integrity of the implementation of the research methodology adopted. In our case, sample bias could affect the study due to the small size, specific experience, and regional distribution of our sample of practitioners and author bias could be a problem because the authors of this study are involved in software architecture research and practice. We addressed the former by deliberately inviting a fairly diverse set of practitioners to participate in the study and we plan to address this weakness more robustly by validating the model with a much larger group. Author bias was addressed by careful construction and execution of the interviews, to avoid leading the participants to any specific answers.

External validity ensures the applicability of the results of the study beyond its initial scope and *conclusion* validity ensures the validity of the conclusions drawn. In our case, we believe we avoid these risks through the reasonable diversity we achieved in our participants and because we did not have any preconceived ideas of likely answers and did not suggest any answers to the participants in the written material or verbally and, we drew our conclusions (i.e. the model) using grounded theory and we believe that this process will largely address risks to our conclusion validity.

7 Future Work

We have created a candidate heuristic model for guiding architects through the process of prioritising their effort to maximise their effectiveness. The next step in the work is to construct a simple questionnaire to explore the usefulness and credibility of the model for a much larger number of practitioners. We believe that this will provide us with a strong validation or refutation of it.

If the model proves to be valid and found to be useful by a significant majority of a larger study group then we would aim to publicise it in practitioner circles via conference sessions and short articles in practitioner-oriented web and print publications.

8 Conclusion

Our experience and informal discussion with architects over many years suggested that they find it difficult to decide how to focus their effort to maximise their effectiveness. We were interested in how practitioners solved this problem and if there were commonly used heuristics. To investigate this, we used a semi-structured interview process with eight experienced practitioners and used Grounded Theory to analyse the results.

The conclusion of our initial study is that there are some shared heuristics which practitioners use, but that the community is not aware that the heuristics are widely known. We found that the heuristics clustered into three groups: focus the architects attention on stakeholders, use their time to address specific risks and delegate as much as possible, in order to give them as much time for architecture work as possible.

These findings are not completely unexpected and many of the heuristics are familiar. However, neither the participants or ourselves knew that these were the key heuristics before we undertook the study, so we believe that the model we have created will have value as a teaching aid and as a reminder to experienced practitioners. This is preliminary work based on a small study, so to validate its usefulness, we plan to continue this work with a much wider, questionnaire based study to find out whether a larger group of practitioners finds the model useful and credible.

References

1. Allen, D.: Getting Things Done: The Art of Stress-free Productivity. 2nd edn. Piatkus (2015).
2. Bass L., Clements, P., Kazman, R.: Software architecture in practice. 3rd edn. Addison Wesley, Upper Saddle River, NJ (2012).
3. Berander, P., Andrews, A.: Requirements prioritization. In: Aurum, A., Wohlin, C. (eds) Engineering and managing software requirements., pp. 69-94. Springer, Heidelberg (2005).
4. Charmaz, K.: Constructing grounded theory: A practical guide through qualitative analysis. Sage, London (2006).
5. De Marco, T.: Slack: Getting Past Burn-out, Busywork, and the Myth of Total Efficiency. Dorset House, New York, NY (2001).
6. Fairbanks, G.: Just Enough Software Architecture, A Risk Driven Approach. Marshall & Brainerd, Boulder, CO (2010).
7. Herrmann, A., Daneva, M.: Requirements prioritization based on benefit and cost prediction: an agenda for future research. In: Tetsuo, T. (ed) International Requirements Engineering, 2008. RE'08. 16th IEEE. IEEE, 2008.
8. Karlsson, J., Ryan K.: A cost-value approach for prioritizing requirements. IEEE Software 14(5) 67-74 (1997).
9. Koch, K.: The 80/20 Principle: The Secret of Achieving More with Less. Nicholas Brearley Publishing (2007).
10. Kruchten, P.: What do software architects really do?, The Journal of Systems and Software, 81(12), 2413–2416 (2008)
11. Matt, G.E., Cook, T.D.: Threats to the validity of research synthesis. Cooper, H., Hedges, L.V. (eds) The Handbook of Research Synthesis, pp. 503-520. Russell Sage Foundation, New York. 503-520 (1994)
12. Poort, E.R., van Vliet H.: RCDA: Architecting as a risk-and cost management discipline. Journal of Systems and Software 85(9) 1995-2013 (2012).
13. Reimer, F.J., Quartaroli, M.T., Lapan, S.D.: Qualitative Research: An Introduction to Methods and Designs. Wiley, London (2012).
14. Rozanski, N., Woods, E.: Software systems architecture, working with stakeholders using viewpoints and perspectives, 2nd edn. Addison Wesley, Upper Saddle River, NJ (2011).
15. Wikipedia, Situational Awareness, https://en.wikipedia.org/wiki/Situation_awareness, last accessed 2017/04/10.