Software Architecture: Organizational Perspectives

Paul L Bannerman
NICTA and School of Computer Science and Engineering, University of NSW, Sydney, Australia
paul.bannerman@nicta.com.au

Abstract

This position paper examines software architecture through the lenses of four organizational perspectives and identifies opportunities for research to improve the positioning and support of the discipline within organizations. It concludes that while software architecture may rest on solid technical foundations, its position in the organization is not as firm.

1. Introduction

As a functional discipline, software architecture (SA) recognizes that its effectiveness depends on non-technical factors as well as technical competences. For example, the support of the organizational ecosystem is recognized as essential in establishing, orchestrating and sustaining SA and software architects [7]. This paper reverses the perspective and argues that how SA is viewed, valued and supported by the organization – that is, how it establishes it value to the organization – also critically determines its success. The paper examines SA externally, from four basic organizational perspectives: management and leadership; governance; capabilities; and strategy. Based on these perspectives, it finds that there are opportunities for SA to improve its position within the organization. Areas for further research are proposed.

Section 2 presents the analyses through the four organizational lenses and Section 3 integrates the findings and illustrates effective embedded SA with a case study, before conclusions are drawn.

2. Organizational perspectives of software architecture and related challenges

Software architecture is a diverse collection of practices. The capabilities of the software architect are described by the profession as both broad and deep, encompassing technology, design, standards, solution determination, application domains, decision-making, software development processes, programming, non-functional requirements, communication, mentoring, strategy, management, leadership, negotiation, politics, diplomacy, consulting, risk management and quality assurance [8, 9, 11, 18]. While it is not explicitly expected that all these capabilities are vested in every architect (indeed, many are shared with developers), there is a strong implication that they are all necessary to competently fulfill the range of duties and challenges of SA in software developing organizations.

Furthermore, there is great diversity in the application of SA because of differences in the contexts of software-developing organizations. For example, product suite developers will have different architectural challenges to shrink-wrapped software, customized application, and component developers. This makes it difficult to generalize about architecture practices.

However, looking beyond this diversity, considering software architecture as an organizational function from an organizational perspective, there are three major challenges that SA faces in gaining visibility, acceptance and support from the organization. First, software architecture is a technical specialty that operates at a low level within the organization, which means that its contribution is not readily visible or demonstrable. Linking architecturally significant business drivers (such as quality attributes) to realized systems via architectures is important for business value creation [13], but this often has limited visibility in practical terms at the business level. For example, goals such as reduce costs and improve quality, processes and market position attract contributions from all organizational activities. Therefore, it can be difficult for SA to demonstrate its specific contribution to business-level outcomes and its value to the firm.

Second, software architects do not have sole claim to architecture within all software organizations. As much as software architects might disagree, similar technical outcomes can often be achieved within traditional software development methods by software design and coding practices (after all, a common view is that good developers make good software architects). Furthermore, multiple layers of architecture practice often apply within organizations, from software to the application/system and enterprise levels. This can make it more difficult for the distinctive contributions of SA to stand out.

Third, the effectiveness and success of software architecture are largely dependent upon the adoption of SA outputs (architectures, architecture principles, rules and practices) by other developers. Therefore, the benefits of SA do not flow solely from the SA function but also from other product development activities.

Individually and combined, these three factors place a heavy onus on SA to lay claim to organizational recognition and support. This is not easy because, fundamentally, the benefits SA provides are manifested indirectly through other, complementary activities and outputs, making it difficult to trace them back. (At a higher organizational level, this same problem faces information technology at large.) This implies a need for research and practice to develop ways of breaking through this visibility barrier by demonstrating the path to value to which SA contributes.

With the aim of providing insights and possible solution paths to these issues, the remainder of this section considers software architecture from four organizational and business-oriented perspectives. In each case, the organizations' interests are outlined and applied to software architecture. Areas for future research are also suggested.

2.1. Management and Leadership

The leadership and management skills of software architects are key interests of the LMSA workshops. Here, I consider leadership and management of SA as a function and discipline within organizations.

From an organizational perspective, management is a formal position-based role responsible for planning, organizing, coordinating, controlling, resourcing and motivating one or more organizational functions and/or activities within a particular domain. This is distinct from the more common use of 'management' as, for example, a software architect exercising discipline in fulfilling his/her responsibilities in an orderly and timely manner. By contrast, leadership is a role, which may or may not be formal or position-based, that challenges, communicates and motivates people to achieve high performance in a particular activity.

Management and leadership are both applicable to SA as a specialist activity. However, to date, they have not attracted specific research attention. This, perhaps, implies a view that there is nothing unique or different in the case of SA as an organizational activity. Since management and leadership of software architecture are likely to vary with the organizational context, there is an opportunity for research to investigate common issues and develop general principles that might apply in most (if not all) firms. This can contribute to establishing the distinctiveness of the function and perhaps apply more structure to its diverse practices.

Without clear differentiation, in practice, software architecture may only attract dedicated management if scale or circumstance dictate that architects be coordinated independently of other software developers and testers. Otherwise, it is likely to be more efficient for software architects to be managed by the software engineering or development manager, together with other specialist development roles. The challenge here is to determine the conditions under which the interests of SA might be better served by separate management and how the overlap in designing, coding, testing and architecture practiced by the other developers might be managed. One common mechanism organizations use for this type of situation is the matrix structure, in which (as applied to this context) each developer reports to both a development and an architecture manager. However, such structures can introduce as many problems and risks as they solve [1].

As well as differentiating itself horizontally, another challenge for SA is determining how to integrate and/or differentiate software architects in an organization that has a hierarchy of architecture roles that require different technical skills (such as software architects; application, system or 'solution' architects; enterprise architects; and chief architect). Governance (discussed in the next section) can contribute to solving this problem, as well as capability development policies that map career structures and progression pathways. However, the solution is also likely to require an arrangement that permits some level of centralized management oversight of architecture principles and standards with decentralized technology adoption and implementation (such an arrangement is described, for example, in [20]).

Software architecture leadership is relevant in three main ways, viz., to motivate best practice, exemplary architecture, and high performance (a) within the SA discipline, (b) across development/test teams and related areas, and (c) within the organization at large.

In many ways, architecture leadership is a greater challenge than management. In practice, leadership is notoriously difficult to develop. Rather, leaders tend to emerge. First, there is the perennial challenge of leadership of any endeavor, namely, to find the right person (with the right motives and the right skills) to arise and unite action through a common vision of architecture practice in a motivational way [14, 15]). Second, there is the difficulty of finding and/or developing the socio-behavioral skills required for leadership in technicians. Third, the socio-cultural processes of institutionalizing SA leadership within the organization and propagating a reputation for that capability within the marketplace are not well understood. Consequently, developing and sustaining a leadership capability provides rich opportunities in practice and for research.

2.2. Governance

While structures such as architecture boards can be found in practice, and governance increasingly appears as a focus area in architecture frameworks (such as [7, 19]), a governance perspective of SA is absent from current research. Arguably, consideration of architecture governance precedes leadership and management because a central role of governance is to enact a framework within which functions such as SA are established, operated and controlled [2]. Governance also positions and integrates the function within the value chain of related activities to ensure coordinated outcomes for the organization. This integration is both horizontal, linking related software development specialties, and vertical, linking the function to higher order functions and to the strategic center of the organization. Therefore, representation in a firm's governance framework is critical. It flags the activity or function as sufficiently important to the organization that its management and operations must be formally scrutinized.

Interest has tended to focus on information technology (IT) governance (e.g., [3, 12, 21, 22]). A notable exception is *The Open Group Architecture Framework* (TOGAF) [19]. However, its focus is on enterprise, rather than software, architecture. By contrast, an emergent *Architecture Competence Framework* by the Software Engineering Institute (SEI) focuses on SA at the project level [7]. Here, I briefly introduce how organizational governance might apply to software architecture as an organizational function, and identify future research opportunities.

Governance is a broad concept, encompassing dimensions of authority, accountability, compliance, legitimacy, oversight, capability, risk management and control. It is responsible for establishing the structures, processes and relational mechanisms that establish and legitimate an organizational function. This includes assigning, monitoring and administering decision rights, responsibilities and accountabilities, as well as establishing the framework of inter-related boards, councils, and/or committees that are needed to provide the required oversight and control. In this regard, governance is different to leadership and management. Effectively, it manages the leadership and management of the function and its activities rather than directly performs the leadership, management and work of the function [2]. Ultimately, governance aims to achieve good order and workable arrangements within the function. Fundamentally, effective governance can establish and institutionalize SA as a key organizational competency.

TOGAF identifies several key success factors for architecture governance that are equally relevant to software architecture [19]:

- Establish and apply best practices for the submission, adoption, reuse, reporting and retirement of architecture policies, procedures, roles, skills, organizational structures and support services
- Establish correct organizational responsibilities and structures to support the architecture governance processes and reporting requirements
- Manage criteria for the control of architecture governance processes, dispensations, compliance assessments, service level agreements and operational level agreements
- Meet requirements for the effectiveness, efficiency, confidentiality, integrity, availability, compliance, and reliability of architecture governance-related information, services and processes.

A particular research challenge is establishing a governance framework that enables and coordinates the independent authorities of software architecture, software development, infrastructure management, and project management within the organization's IT governance arrangements. At present, this need tends to be fulfilled in practice by various home-grown arrangements that have evolved over time. These include informal project-level mechanisms as well as formal design review boards, architecture boards or general IT steering committees. Governance research can help both establish a distinctive role for software architecture and position it within the organization's framework of key IT accountabilities.

2.3. Capabilities

Some research has been done by SEI on individual SA competences [6, 10], resulting in a draft *Architecture Competence Framework* [7], but there is nothing on software architecture as an organizational capability. Viewing it in this way would directly raise the question of the importance of SA to the business. Recognition and support of a function as a capability is dependent upon its value to the business [5].

Broadly speaking, two categories of capabilities are distinguished in the literature: those that can provide a distinctive advantage to the firm; and those that are common to all (or most) firms [5]. The former are called core or strategic capabilities while the latter are called ordinary or operational capabilities. Successful organizations invest greatly in building and maintaining their core capabilities but source ordinary capabilities on the basis of least cost or expedience.

Therefore, for an organization to make a significant commitment to software architecture as one of its core capabilities, it needs to have a clear vision of its distinctive value to the software development business. It would also need to be deeply embedded in the organization so that it could not be easily purchased or replicated by other organizations (competitors) or substituted by other capabilities to the same effect.

This is likely to present a particular challenge for software architecture as much of its distinctive value is vested in the tacit skills of individual architects – to analyze and synthesize alternatives technologies and architectures. Within any particular domain, architects are highly mobile and tradable on the open market. Such people-dependent capabilities (sometimes called competences) are therefore available to all firms (albeit, at a price) so they offer little sustainable (long term) advantage for a firm.

Consequently, the degree to which SA can attract the attention and support of the organization is dependent upon how deeply and distinctively the competences that are not highly people-dependent are embedded in the organization, in its practices, processes, systems, structures, culture, values, know-how and technologies.

Distinctive capabilities in SA are likely to shape and be shaped by the type of business the organization is engaged in. For example, a developer of customized solutions with well-architected systems that retain their central integrity over time is likely to attract business because of this capability. Preserving and enriching it becomes an organizational imperative. Similarly for a product developer that can save costs through transitioning a system across different technologies and platforms over time through good SA.

Current research on developing competence frameworks is making a critical contribution in defining a minimal set of operational capabilities that an organization is likely to need to practice effective SA. What is yet to be researched is methods for identifying, measuring and developing distinctive SA capabilities. That is, firm-specific capabilities that have the potential to deliver strategic benefits.

The challenge, therefore, for organization-based SA research, is to (a) develop a clear path to value of the role of SA in the firm; (b) identify the organization's distinctive SA capabilities and opportunities; (c) develop strategies for SA capability development within the function and development teams, and; (d) build tools and techniques to enable these analyses to be done and strategies to be applied at the level of individual organizations, to take account of firm-specific differences.

2.4. Strategy

The role and importance of SA to the organization will vary according to its place in the organization's strategy. Strategy is a broad and complex field. Here, the discussion is limited to consideration of SA within the context of four generic strategic organizational types found in the literature: defender, prospector, analyzer and reactor [16].

Defender. Defenders operate in stable, focused product-market domains, applying an established set of technologies and developing expertise in improving efficiency in existing operations. These organizations invest heavily in designing their systems, usually based on a dominant technology, to minimize variability and uncertainty rather than in continuously seeking out new technologies and opportunities. They seek the most efficient solution to a specific problem through technological efficiency and optimized processes. Subsequent investment is limited to updating current technology and processes to maintain efficiency. A challenge for defender SA is in striking a balance between maintaining adherence to proven standards and incrementally transitioning to new technologies and architectures when it become necessary to do so. The defender's strengths are more in the routineness of its operations than (necessarily) in the smartness of its technologies or people.

Prospector. These are the industry leaders, focused on product and market innovation and creating change and uncertainty for their competitors. However, they are not always operationally efficient. Prospectors offer many opportunities for SA. Technology choices are not limited to a dominant type or current capabilities, and solution design actively avoids being locked in to any single technology or process. Flexibility (agility) in design and development processes is favored over standardized procedures and efficiency. However, agility in responding to changing environmental contexts makes for a high degree of efficiency in the ability to innovate. Solution prototyping is common. The prospector's strengths are in the knowledge and skills of its people to apply technology in innovative ways. The technologies may change but the creativity and dedication of its people are indispensable. A weakness of the prospector is that it cannot develop maximum operational efficiencies (that is, a low cost base) because of the commitment to rapid response and multiple technologies.

Analyzer. Analyzers operate in two domains: one stable and the other dynamic. In the stable domain they operate routinely and efficiently through formalized arrangements. In the dynamic domain, they watch the industry and competitors for new ideas to adopt and/or opportunities to which they can respond. The analyzer has to manage conflicting demands for technological stability and flexibility. It is the so-called ambidextrous organization [17]. This is typically achieved by some structural solution that enables the conditions of the defender organization (stable operations and technologies) to coexist with the conditions of the prospector organization (flexible operations and multiple technologies) within the same firm (usually in separate divisions). This permits the possibility of one side (such as the defender-type function) being more dominant than the other. Regardless, analyzers remain a compromise of the pure defender and prospector types, never being completely efficient or effective.

Reactor. Each of the above types has an established pattern of behavior to adapt to its business environment, through seeking to develop greater efficiency and/or openness to new opportunities. By contrast, reactors are unable to respond effectively to perceived industry changes and uncertainties until they are forced to. They lack consistent response mechanisms to apply to changing environmental conditions. This mostly stem from inadequate organizational governance and management, making it difficult for them to prosper or survive in the long term.

Software architecture is likely to have a diminishing role and importance to organizations of a particular strategic type in the following order (from highest to lowest): prospector, analyzer, defender, reactor.

3. Integration and case illustration

It has been argued that software architecture is challenged in making its value and contribution to the organization's systems and product development functions visible because of its low level of application, shared responsibility, dependency on others, and indirect effects. This means that in addition to fulfilling its technical charter, software architecture must (a) establish a profile of its role and contribution to the organization's strategic and operational outcomes; (b) develop effective ways to influence other specialty domains (such as non-architect developers) to accept and follow its technical leadership; and (c) actively seek out ways to reduce costs or increase revenue. SA also faces the challenges of vertically integrating with higher order architecture disciplines and differentiating its own unique contribution within this hierarchy.

Building a research-based view of SA in organizational (as well as technical) terms will greatly facilitate progress against these challenges. In particular, viewing SA as a distinctive, valuable, firm-specific resource (a core capability) that must be fostered and controlled through the organization's governance framework, positions it in the organization's portfolio of strategic resources. This will contribute significantly to giving it the visibility that it needs to attract organizational attentions and support. In practical terms, it provides a platform for software architecture's contribution to business performance to become transparent to the executive, and for it to more readily participate in the goals and evolving opportunities of the firm. In turn, this also positions SA to attract great focus and support from the organization to extend its role and improve its effectiveness in the organization's 'business as usual' activities as well as align it to the organization's strategic orientation and initiatives.

One organization that is applying this theory in practice is NCW Systems (a pseudonym), a subsidiary of a major aerospace/defense contractor [4]. The defense division's expertise, as a lead systems integrator in government contracts for large software-enabled defense systems, is viewed as a core capability that is governed through an elaborate framework. Engineering governance falls under discipline-based Capability Managers and Process Councils. The former provide leadership as well as ensure that the processes, people, technologies, tools and environments needed by the organization to apply that capability are in place when and where they are needed. Process councils ensure that adequate processes are maintained, projects comply with process standards, and necessary process improvements are made. Software architecture is guided by an internally developed Software Engineering Reference Framework (SERF) and a Strategic Architecture Reference Model (SARM) that is consistent with the Department of Defense Architecture Framework (DoDAF). It also deploys tools such as SEI's Architecture Tradeoff Analysis Method (ATAM). Software architecture analysis and definition practices emphasize identifying and representing quality characteristics to stakeholders; maintenance and reuse of design patterns; and communicating architectures through stakeholder-relevant views. The company is developing its own capability assessment framework to assess the maturity of its existing capabilities. This is then analyzed against the outputs of sophisticated strategic planning to identify (and subsequently fill) gaps in their existing and projected future capability requirements.

4. Conclusions

This position paper has examined software architecture through the lenses of four fundamental organizational perspectives and identified opportunities for research to improve the positioning, acceptance and support of the discipline within organizations.

The overall conclusion of the analyses is that while software architecture may rest on firm technical foundations, its position in the organization is not as solid as it could be. Focused research on the organization-based enablers and inhibitors of software architecture is likely to improve both the discipline and its standing within organizations. Software architecture has much to contribute to business goals as a technical and organizational capability. This potential value will only be realized as the field develops in both its technical and non-technical dimensions.

5. References

[1] Bannerman, P.L., "Risk Implications of Software Project Organization Structures", *Proceedings of the 20th Australian Software Engineering Conference*, Gold Coast, April 2009.

[2] Bannerman, P.L., "Software Development Governance: A Meta-management Perspective", Software Development Governance 2009 (an ICSE'09 workshop), 2009.

[3] Bannerman, P.L., "IT Governance as a Necessary Evil", Working Paper (submitted to ECIS 2009), 2009.

[4] Bannerman, P.L., and Staples, M., "Capabilities of Software-Developing Organizations", Working Paper, NICTA, 2008.

[5] Barney, J.B., and Clark, D.N., *Resource-based Theory: Creating and Sustaining Competitive Advantage*, Oxford University Press, Oxford, 2007.

[6] Bass, L., Clements, P., Kazman, R., and Klein, M., *Models for Evaluating and Improving Architecture Competence*, Software Engineering Institute, CMU/SEI-2008-TR-006, March 2008.

[7] Bass, L., Clements, P., Kazman, R., Klein, J., Klein, M., & Sivily, J., *A Workshop on Architecture Competence*, Software Engineering Institute, Technical Note, October 2008.

[8] Bredemeyer, D., and Malan, R., "The Role of the Architect", White Paper, 2006, accessed December 2008, http://www.bredemeyer.com/pdf_files/role.pdf.

[9] Brown, S., "The Role of the Software Architect in Successful Projects", C5 Alliance, 2008, accessed December 2008, http://static.codingthearchitecture.com/presentations/20081113-role-of-the-architect.pdf.

[10] Clements, P., Kazman, R., Klein, M., Devesh, D., Reddy, S., and Verma, P., "The Duties, Skills and Knowledge of Software Architects", *Proceedings of the Working IEEE/IFIP Conference on Software Architecture* (WICSA'07), 2007.

[11] Eeles, P., "Characteristics of a Software Architect", http://www.ibm.com/developerworks/rational/library/march06/eeles, March 2006, accessed December 2008.

[12] ITGI, *COBIT 4.1*, IT Governance Institute, 2007.

[13] Kazman, R., and Bass, L., *Categorizing Business Goals for Software Architectures*, Software Engineering Institute, CMU/SEI-2005-TR-021, December 2005.

[14] Kotter, J.P., "Leading Change: Why Transformation Efforts Fail", *Harvard Business Review*, 85(1), 2007, pp96-103.

[15] Malan, R., and Bredemeyer, D., *Leadership: Architect Competency Elaboration*, 2002, accessed January 2009, http://www.bredemeyer.com/pdf_files/LeadershipCompetency.PDF.

[16] Miles, R.E., and C.C. *Snow, Organizational Strategy, Structure, and Process*, Stanford Business Classic, Stanford, 2003 (originally published in 1978 by McGraw-Hill).

[17] O'Reilly III, C.A., and M. L. Tushman, "The Ambidextrous Organization", *Harvard Business Review*, 82(4), 2004, pp. 74-81.

[18] SEI, "What are the Duties, Skills and Knowledge of a Software Architect?", accessed December 2008, http://www.sei.cmu.edu/architecture/arch-duties.html.

[19] TOGAF, "Architecture Governance", The Open Group Architecture Framework (TOGAF), accessed January 2009, http://www.opengroup.org/architecture/togaf8-doc/arch/chap26.html.

[20] VA, "Enterprise Architecture: Strategy, Governance, & Implementation", Department of Veterans Affairs, August 2001, accessed January 2009, http://www.enterprise-architecture.info/Images/Documents/DVAEAGov.pdf.

[21] Van Grembergen, W., and De Haes, S., *Implementing Information Technology Governance: Models, Practices, and Cases*, IGI Publishing, Hershey, 2008.

[22] Weill, P., and Ross, J.W., *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*, Harvard Business School Press, Boston, 2004.

8. Acknowledgements

This paper benefits from comments by Liming Zhu. NICTA is funded by the Australian Government via the Department of Broadband, Communications and Digital Economy and the Australian Research Council.