Supporting Organizational Qualities Through Architectural Patterns

Fritz Solms
Department of Computer Science, University of Pretoria, Pretoria, South Africa

Keywords: Organizational Architecture, Architectural Patterns, Blackboard, Microkernel, Pipes and Filters, Organizational Quality.

Abstract: The structure of an organization is known to affect an organization's qualities. Commonly variations of the hierarchical patterns are used to constrain the structure of organizations. Examples include functional and divisional hierarchies, and organizations based on a matrix structure. In either case the depth of the hierarchies can be varied. For software architectures a much wider set of architectural patterns is considered and patterns are combined in such a way as to support desired system qualities. This paper suggests that many of these patterns are applicable also for organizational architectures and proposes a conceptual framework for the selection and application of architectural patterns in order to improve the quality attributes of an organization.

1 INTRODUCTION

A *quality attribute* of a system is the realization of a non-functional requirement (NFR). Examples of quality attributes include scalability, reliability, flexibility, integrability, performance, and cost. In software architecture design it is common to select and combine architectural patterns to specify an infrastructure within which it is easier to satisfy required quality attributes (Bachmann et al., 2005; Harrison and Avgeriou, 2007).

Functional and divisional hierarchies as well as matrix structures are commonly used in organizational architecture design (Tran and Tian, 2013; Harris and Raviv, 2002). These are based on the hierarchical pattern. Organizational architecture optimization involves activities like optimizing, grouping into clusters, specifying depth (tallness or flatness of the hierarchy), choosing and specifying the level of decentralization and standardization (Harris and Raviv, 2002).

This paper suggests that many of the architectural patterns used in software architecture design are equally applicable to organizational architecture design. In particular, the *Microkernel* pattern can facilitate service provider flexibility and integrability, the *Blackboard* pattern innovatability and the *Pipes and Filters* pattern process flexibility and low cost. It is suggested that architectural patterns should be combined to support required qualities across an organization.

2 RELATED WORK

Weber's early work on organizational theory and the bureaucratic model (Weber, 1958) included a study of the impact of responsibility and accountability localization within a hierarchical structure. Both, functional and divisional hierarchies were considered. The former are based on a segregation of activities resulting in good skills separation. However, changes to product scope or business processes require changes across the organization which are not well managed within a functional hierarchy. Divisional hierarchies base the segregation on some other variable like product or geography. Whilst changes to product scope and business processes are better managed within divisional hierarchies, they commonly lead to duplication of resources and poor product integration.

Matrix structures (Mee, 1964; Galbraith, 1971) were introduced to realize management and coordination across functional and divisional domains with individuals reporting to multiple managers. Benefits include more efficient resource utilization and wider information flow resulting in increased innovation and knowledge distribution. However, multiple reporting lines may introduce conflicts and unhealthy competition for resources.

We have seen a trend from narrow and tall organizations toward flat organizations with wider spans of control (Docherty et al., 2001), i.e. to hierarchies with less layers. Benefits are reduced specialization, loosened hierarchies of control, rapid and less centra lized decision making, closer contact between man-

Solms, F.
Supporting Organizational Qualities Through Architectural Patterns.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 2*, pages 594-599
ISBN: 978-989-758-187-8

agement and workers enabling the latter to gain a better understanding of the strategic goals of the organization, improved innovation and knowledge distribution and lower cost. Disadvantages of flat organizations include possible quality degradation in the context of reduced control, difficulties around scaling to large organizations and handling complex processes. Also, flat organizations put strain on management and lack of precise definition of employee roles may lead to uncertainty. (Dalton et al., 1980) have shown that hierarchical depth, degree of specialization, span of control, the level of centralization, administrative intensity and the level of formalization and standardization impact not only on frequency and duration of labour strikes and the level of absenteeism, but also on the turnover of the organization.

This leads one to model the design of an organizational architecture more formally as an optimization problem (Harris and Raviv, 2002; Tran and Tian, 2013). (Harris and Raviv, 2002) modeled the impact of changes to an organization's structure on its basic activities of producing, designing and marketing products, its coordination activities and managerial costs. They found the optimal structure follows a life cycle as the company grows in size and complexity. In particular, a company tends to evolve from flat structure to a highly centralized structure to a divisional hierarchy to a functional hierarchy and finally to either a matrix structure or a highly decentralized flat structure. They also found that firms that do not face high resource constraints, highly regulated firms and firms operating in stable environment tend to have decentralized organizational structures.

Moving beyond hierarchical structures, (Mintzberg, 1979) introduced the *Structure of 5*. Here an operational core is managed by the middle-line (middle management) reporting to the strategic APEX of the organization with support from technostructure and support staff. Technostructure covers areas which are not involved in operations but may observe, design and optimize them, e.g. research and development, business analysis, operational research and strategic planning.

(Kolp et al., 2004) study the ease with which NFRs can be addressed within different organizational styles. They include *Structure of 5*, *Pyramid*, *Matrix*, *Chain of Value* and the *Bidding Style*. (Gómez and Ortiz, 2013) extend the analysis of Kolp and Tung Do to include further qualities including security and availability.

(Berzisa, 2015) proposed *Capability Driven Development* (CDD) for integrating organizational and information system development. They suggest reusing organizational patterns, but these refer to operational patterns and not structural patterns.

3 STRUCTURAL PATTERNS

In this section the hierarchical pattern widely used in organizational architecture design is compared to structural patterns used in software architecture design but currently not explicitly in organizational architecture design. It is shown how patterns can be combined to enhance different qualities in different parts of an organization.

3.1 The Hierarchical Pattern

Functional and divisional hierarchies, flat structures (single level hierarchies) and matrix structures are all based on the hierarchical pattern

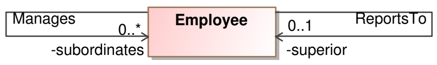

Figure 1: A UML class diagram representing the structure of the hierarchical pattern. Employees have a communication channel to their superior as well as to their subordinates. Exceptions are the top (CEO) and bottom of the reporting hierarchy.

Figure 1 shows the reporting lines specified by the simple hierarchical pattern. The pattern introduces specialized roles requiring a person to manage a particular domain of responsibility at a particular level of accountability as is more explicitly depicted in Figure 2. The natural human resourcing approach is thus job based human resourcing.

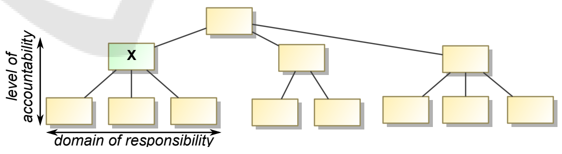

Figure 2: A UML object diagram depicting an instance structure. It shows the domain of responsibility and level of accountability assigned to employee X.

Accountability hierarchies improve reliability and role based human resourcing around defined domains of responsibility at defined levels of accountability result in standard skills requirements which simplifies scalability.

On the other hand, the reduced communication channels and fixed ring-fenced domains of responsibility make it more difficult to achieve high levels of innovatability and learnability. Changing processes within hierarchical organizations is often more difficult as processes typically cut across domains of responsibility and business units. Finally, deep hierarchies incur a significant cost-overhead.

The hierarchical pattern is particularly applicable in areas where control and accountability hierarchies are desirable. It's use should be minimized in areas of the organization where innovatability and flexibility are important.

3.2 The Blackboard Pattern

The blackboard architectural pattern (Buschmann et al., 1996) has been applied to a wide range of problems including speech recognition (Lesser et al., 1975), distributed resource allocation planning (Han et al., 2014), and decision support for stock trading (Luo et al., 2002). It provides an infrastructure for specialized processing units (experts) to collaborate in developing a possibly partial or approximate solution to a difficult problem for which no deterministic solution strategies are known.

The pattern obtained its name from an analog of mathematicians collaboratively solving a difficult problem using a blackboard as communication mechanism. An observable blackboard hosts both the problem specification as well as the current state of the solution. One mathematician may try one approach to a solution and get stuck leading to another expert perhaps continuing on the approach or alternatively arguing why the approach will not work and removing the contributions made by the first mathematician from the blackboard. Note that the experts auto-orchestrate a process amongst themselves.

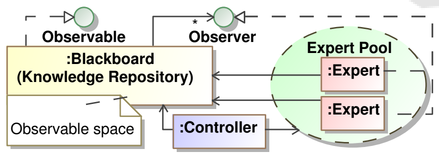

Figure 3: A UML object diagram depicting the structure of the blackboard pattern.

In the blackboard pattern (see Figure 3) a controller manages a pool of experts and feeds problems into a blackboard which is observed by them. Operating as a self-organizing team, experts decide when to process information or requests published on the blackboard. The output of any processing is fed back into the blackboard and may trigger further activities by experts. The controller manages the scope of the problem and decides when a solution or a sufficient approximation to it has been obtained. The natural human resourcing for blackboard based architectures is skills based human resourcing (Lawler and Ledford, 1992) and not role-based human resourcing.

The strengths of the blackboard pattern include *innovatability* through auto-orchestrated processes as well as *learnability and knowledge distribution* (Jurado et al., 2012). The blackboard provides an integration infrastructure improving *integrability* (Jurado et al., 2012) and *auditability*. *Scalability* can be achieved by adding further experts for concurrent processing within a processing grid (Dong et al., 2005). One can dynamically add and remove experts resulting new processes being immediately auto-orchestrated across the new skills set. This increases *flexibility*. *Security* is enforced through access control on the blackboard (Ortega-Arjona and Fernandez, 2008). Weaknesses include reduced *predictability and reliability* as the success of auto-orchestrated is not guaranteed. Communication overheads and auto-orchestration may lead to performance concerns (Schwartz, 1995). Due to redundancy and non-stream-lined processes, the *blackboard* pattern is not usually suited for organizations competing on cost.

Even though the blackboard pattern is currently not used explicitly in organizational architecture design, one finds organizational structures which are implicitly based on this pattern. In open source software development organizations the version control repository represents the blackboard, developers the experts, and the project lead the controller. Other examples include Wikipedia, the bidding style discussed by (Kolp et al., 2004) and even a board of directors. One should consider the blackboard pattern when innovation, learnability, extreme flexibility are required, e.g. for R&D, marketing and software development teams.

3.3 The Microkernel Pattern

The microkernel pattern was introduced for operating systems to separate a minimal set of core functionality for which flexibility is sacrificed for efficiency and reliability from pluggable extension code which requires a higher level of flexibility (Liedtke, 1995). It has also been used for application servers (Goebel and Nestler, 2004) and is the base pattern for Services Oriented Architectures (SOA) with the Enterprise Services Bus (ESB) representing the microkernel (Solms, 2012).

Figure 4 depicts the structure of the microkernel pattern. The microkernel and internal servers are efficient, reliable implementations which evolve only slowly whilst external servers evolve more rapidly to address evolving client requirements. The microkernel provides a communication infrastructure decoupling the various servers. Clients access external servers through an adapter.

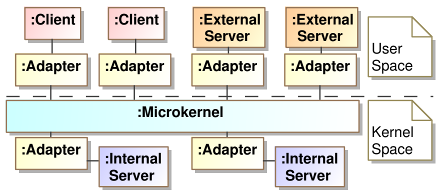

Figure 4: A UML object diagram depicting the structure of the microkernel pattern.

The main purpose of using a microkernel pattern is usually *integrability* (Solms, 2012) which, in turn, improves *reusablity* and *flexibility* through decoupling and pluggability. *Auditability* is achieved by logging all messages traversing the microkernel.

On the other hand, decoupling via a microkernel introduces *performance* and *cost* overheads (Solms, 2015; Sun et al., 2013) and the isolation of the various responsibility domains (servers) reduces opportunities for *innovation* and *learning*.

An example for implicit use of the microkernel pattern with organizational architecture design is that of an internal post-office routing messages between internal departments as well as to and from external organizations to the appropriate internal business units represents a microkernel. Also, corporate and merchant banks can be seen to be implicitly based on the microkernel pattern. Client faced business units like credit derivatives and foreign exchange operate like independent organizations. These are the external servers which need to be flexible and innovative, evolving their service offering to address changing client needs and environmental changes. Back-office services like transaction processing and regulatory reporting which need to provide their services reliably and efficiently are the internal servers which change infrequently. The communication between client faced external servers and the internal servers providing back-office services is commonly facilitated through an *Enterprise Services Bus* (ESB). Finally, clients are provided unified access channels across external servers in the form of a web interface and call center for human access and web-services and messaging adapters for system access. This unified interface represents the adapter of the microkernel pattern.

3.4 The Pipes and Filters Pattern

The *pipes and filters* pattern was originally used for compiler design (Meunier, 1995). The pattern is widely used for processing pipelines like signal or video processing, *Unix* command pipelines and for workflow systems (Scheibler et al., 2010).

Filters represent processing units which process

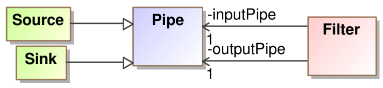

Figure 5: A UML class diagram depicting the structure of the pipes and filters pattern.

inputs received from an input pipe and deliver outputs to an output pipe (see Figure 5). Pipes have the responsibility of transporting artifacts between filters. Processing units are decoupled through pipes and defined pipes and filters processes can be aggregated into higher-level, reusable filters.

The main strength of the pipes and filters pattern are process *flexibility* including the ability to quickly assemble new processes from the available processing components (filters) (Scheibler et al., 2010). Weaknesses include low innovatability and learnability due to the isolation of processing units and the lack of overall *control*. The pattern does, however, incur only *low cost* overheads as the structure is focused on value-generating processing components.

The pipes and filters pattern is the base pattern for the *Chain of Value* style discussed by (Kolp et al., 2004). It is also commonly used to achieve flexibility in manufacturing organizations. The filters are machines performing processing steps on the product being built and pipes (e.g. conveyor belts) transport the product between processing steps.

4 LEVELS OF GRANULARITY

When designing an infrastructure for an organization, one is unlikely to use a single pattern. Instead one may constrain the high-level structure for the system as a whole (the first level of granularity) with a particular pattern, whilst constraining the infrastructure of lower level organizational components through other patterns.

The infrastructure design of an organization can be driven by a recursive decomposition into business units requiring qualities and basing the selection of the pattern constraining its infrastructure on the ease with which the required qualities are realized within the structure prescribed by the pattern as discussed in section 3 and summarized in Table 1. Generally one will want to select a pattern which supports the most important quality attribute(s) for the respective business unit whilst not negatively impacting on other relevant qualities.

For example, an infrastructure based on the *hierarchical* pattern could be chosen if *reliability/control* are important and the *microkernel* or *blackboard* for *integrability* and *pipes and filters* for *process flexibil-*

Table 1: Impact of architectural patterns on organizational qualities. An up/down-arrow indicates that the pattern makes it easier/more difficult to realize a quality attribute, whilst the tilde symbol indicates that it has no significant effect on a quality attribute.

Pattern	**Quality Attribute**								
Quality	Innovatability	Learnability	Flexibility	Reliability	Integrability	Reusability	Scalability	Performance	Auditability
Microkernel	~	~	↑	~	↑	↑	~	↓	↑
Blackboard	↑	↑	↑	↓	↑	↑	↑	↓	↑
Hierarchcal	↓	↓	↓	↑	↓	↓	↑	~	~
Pipes & Filt	↓	↓	↑	~	~	↑	~	~	~

ity and *low cost*. The blackboard pattern is the only pattern which provides infrastructural support for innovation.

As an illustrative example consider designing the infrastructure for a hypothetical corporate and merchant bank. Assume the organization has a large number of specialized trading units operating as virtually independent business units with their own accounting base (e.g. credit derivatives, foreign exchange and equities) which compete in the market by constructing innovative hedging products for their clients. These trading units consume back-end services like transaction processing, regulatory reporting and procurement from back-office units. These back-office units need to not only provide their services efficiently, but must also be able to rapidly and at low cost change their business processes to adapt to continuously changing laws an regulations. The organization as a whole is to be controlled by a relatively shallow management structure.

The most important quality requirement for the small trading business units is innovatability. Hence one may consider either basing their infrastructure on the blackboard architectural pattern or use that pattern within those business units as incubator for innovation.

The back-office services could provide their streamlined services within a pipes and filters infrastructure which allows for rapid and efficient process changes driven by regulatory requirements. Selecting the *microkernel* pattern as integration infrastructure between the loosely coupled client facing and back-office business units enhances flexibility and the ability to easily add further business units. The back-office units represent the internal servers of the microkernel pattern, the trading units the external servers and a unified client interface the adapter. The microkernel is responsible for routing requests from the client adapter to the appropriate trading units and between the trading and back-office units. The concrete manifestation might be an *Enterprise Services Bus* (ESB) within a service-oriented software architecture.

For the control infrastructure for each of the business units one could consider a single layer hierarchy which in turn reports to a relatively flat *hierarchy* for overall control and accountability of the organization. The above combination of pattern is chosen to facilitate innovation, flexibility, integrability and accountability and control where these respective qualities are important.

5 CONCLUSIONS AND FUTURE WORK

Organizational architecture design has predominantly focused on structures based on the hierarchical pattern including functional and divisional hierarchies, matrix structures and flat hierarchies. Hierarchical structures are strong in control and accountability, but achieving qualities like innovatability, flexibility, learning, and knowledge distribution is difficult within such structures.

This paper argues that one should also consider other architectural patterns like blackboard, hierarchical and pipes and filters to achieve qualities like flexibility, innovatability, and integrability. One should select and combine patterns in such a way that they facilitate the required qualities across organizational components. Furthermore, viewing enterprise architecture as an aggregation of organizational and software architecture makes it attractive to use the same concepts across both these sub-domains. Such an approach may make it easier to achieve alignment of software architecture and organizational architecture – it is across both of these that an organization needs to realize its desired quality attributes.

Future work will include include empirical studies investigating the correlation between the use of architectural patterns and organizational qualities. The work of Harris and Raviv (Harris and Raviv, 2002) can be extended to provide a formal optimization framework for selecting and combining architectural patterns.

Whist the selection of architectural patterns provides an infrastructure within which it is easier to realize certain qualities, it is the selection of architectural tactics like load balancing, resource reuse, active and passive redundancy which concretely realize these qualities (Solms, 2012). Future work will aim to include the selection of architectural patterns and tactics within a comprehensive method for organizational architecture design.

REFERENCES

Bachmann, F., Bass, L., Klein, M., and Shelton, C. (2005). Designing Software Architectures to Achieve Quality Attribute Requirements. *IEE Proceedings - Software*, 152(4):153–165.

Berzisa, S. e. a. (2015). Capability driven development: An approach to designing digital enterprises. *Business & Information Systems Engineering*, 57(1):15–25.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley, Chichester, UK.

Dalton, D. R., Todor, W. T., Spendolini, M. J., Fielding, G. J., and Porter, L. W. (1980). Organizational structure and performans: A critical review. *Academy of Management Review*, 5(1):49–64.

Docherty, J. P., Surles, R. C., and Donovan, C. M. (2001). *Textbook of Administrative Psychiatry*, chapter Organizational Theory, pages 33–43. ASM Zahidul Islam, second edition edition.

Dong, J., Chen, S., and Jeng, J.-J. (2005). Event-based blackboard architecture for multi-agent systems. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 379–384 Vol. 2.

Galbraith, J. R. (1971). Matrix organization designs: How to combine functional and project forms. *Business Horizons*, 14(1):29 – 40.

Goebel, S. and Nestler, M. (2004). Composite component support for ejb. In *Proceedings of the Winter International Synposium on Information and Communication Technologies*, WISICT '04, pages 1–6. Trinity College Dublin.

Gómez, O. and Ortiz, J. (2013). Modeling the organizational style structure in five. *Procedia Technology*, 7:384 – 390.

Han, X., Mandal, S., Pattipati, K., Kleinman, D., and Mishra, M. (2014). An optimization-based distributed planning algorithm: A blackboard-based collaborative framework. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 44(6):673–686.

Harris, M. and Raviv, A. (2002). Organization design. *Manage. Sci.*, 48(7):852–865.

Harrison, N. B. and Avgeriou, P. (2007). Leveraging architecture patterns to satisfy quality attributes. In Oquendo, F., editor, *Software Architecture, First European Conference, ECSA 2007, Aranjuez, Spain, September 24-26, 2007, Proceedings*, volume 4758 of *Lecture Notes in Computer Science*, pages 263–270. Springer.

Jurado, F., Redondo, M. A., and Ortega, M. (2012). Blackboard architecture to integrate components and agents in heterogeneous distributed elearning systems: An application for learning to program. *Journal of Systems and Software*, 85(7):1621 – 1636. Software Ecosystems.

Kolp, M., Do, T. T., and Faulkner, S. (2004). Analysis styles for requirements engineering: An organizational perspective. In Chang, S., editor, *Handbook Of Software Engineering And Knowledge Engineering*, volume 3, chapter 1. World Scientific.

Lawler, E. and Ledford, G. (1992). A skill-based approach to human resource management. *European Management Journal*, 10(4):383–391.

Lesser, V., Fennell, R., Erman, L., and Reddy, D. (1975). Organization of the hearsay ii speech understanding system. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):11–24.

Liedtke, J. (1995). On micro-kernel construction. *SIGOPS Oper. Syst. Rev.*, 29(5):237–250.

Luo, Y., Liu, K., and Davis, D. (2002). A multi-agent decision support system for stock trading. *Network, IEEE*, 16(1):20–27.

Mee, J. F. (1964). Matrix organization. *Business Horizons*, 7(2):70 – 72.

Meunier, R. (1995). The pipes and filters architecture. In *Pattern Languages of Program Design*, pages 427–440. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Mintzberg, H. (1979). *The structuring of organizations: a synthesis of the research*. Theory of management policy series. Prentice-Hall.

Ortega-Arjona, J. L. and Fernandez, E. B. (2008). The secure blackboard pattern. In *Proceedings of the 15th Conference on Pattern Languages of Programs*, PLoP '08, pages 22:1–22:5, New York, NY, USA. ACM.

Scheibler, T., Leymann, F., and Roller, D. (2010). Executing pipes-and-filters with workflows. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, pages 143–148.

Schwartz, D. (1995). Blackboard systems. In *Cooperating Heterogeneous Systems*, volume 299 of *The Springer International Series in Engineering and Computer Science*, pages 27–42. Springer US.

Solms, F. (2012). What is software architecture? In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, SAICSIT '12, pages 363–373, New York, NY, USA. ACM.

Solms, F. (2015). A Systematic Method for Architecture Recovery. In *10'th International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 215–222, Barcelona, Spain.

Sun, H., Wang, X., Yan, M., Tang, Y., and Liu, X. (2013). Towards a scalable paas for service oriented software. In *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, pages 522–527.

Tran, Q. and Tian, Y. (2013). Organizational structure: Influencing factors and impact on a firm. *American Journal of Industrial and Business Management*, (3):229–236.

Weber, M. (1958). *From Max Weber: Essays in Sociology*. Oxford University Press, USA.