The Addison-Wesley Signature Series

A Vaughn Vernon Signature Book

Strategic Monoliths and Microservices

Driving Innovation Using Purposeful Architecture

Vaughn Vernon
Tomasz Jaskuła

Foreword by
Mary Poppendieck

Praise for Strategic Monoliths and Microservices

"Most books address either the business of software or the technical details of building software. *Strategic Monoliths and Microservices* provides a comprehensive approach to blending the needs of business and technology in an approachable way. It also dispels many of today's myths while offering practical guidance that any team or organization can apply immediately and with confidence."

—*James Higginbotham, Executive API Consultant, Founder of LaunchAny, and author of* Principles of Web API Design

"Digital Transformation cannot succeed as a 'grass roots' effort. Vaughn and Tomasz offer C-level execs a roadmap to software excellence that includes establishing the culture necessary to foster and sustain software innovation. Written with real-world understanding, Vaughn and Tomasz help the reader to appreciate that moving software development from a cost center to a profit center involves tradeoffs that need not sacrifice innovation. A must-read for decision makers."

—*Tom Stockton, Principal Architect, MAXIMUS*

"In this book, Vaughn Vernon and Tomasz Jaskuła use their extensive experience with DDD to present a comprehensive guide to using the many different aspects of DDD for modern systems development and modernization. It will be a valuable guide for many technical leaders who need to understand how to use DDD to its full potential."

—*Eoin Woods, software architect and author*

"There are common misconceptions and roots of failure around software engineering. One notable example is neglecting the rugged trek towards digital transformation. Such an endeavor comprises breakthrough innovations, failure culture, emphasis on the role of software architecture, as well as on the importance of efficient and effective inter-human communication. Fortunately, the authors offer the necessary help for mastering all hurdles and challenges. What I like most about this book is the holistic view it provides to all stakeholders involved in digital transformation and innovation. Vaughn Vernon and Tomasz Jaskuła introduce a clear path to successful innovation projects. They provide insights, tools, proven best practices, and architecture styles both from the business and engineering viewpoint. Their book sheds light on the implications of digital transformation and how to deal with them successfully. This book deserves to become a must-read for practicing software engineers, executives, as well as senior managers. It will always serve me as a precious source of guidance and as a navigator whenever I am entering unchartered territories."

—*Michael Stal, Certified Senior Software Architect, Siemens Technology* decisions made over the long haul. An ADR provides a document template that is used to capture each important architectural decision made, along with its context and consequences.

Each ADR should be stored along with the source code to which it applies, so they are easily accessible for any team member. You might protest that #agile doesn't require any documentation, and that the current code should be self-explanatory. That viewpoint is not entirely accurate. The #agile approach avoids *useless* documentation, but allows for any documentation that helps technical and business stakeholders understand the current context. More to the point, ADRs are very lightweight.

A number of ADR templates are available, but Michael Nygard proposed a particularly simple, yet powerful, one [Nygard-ADR]. His point of view is that an ADR should be a collection of records for "architecturally significant" decisions—those that affect the structure, nonfunctional characteristics, dependencies, interfaces, or construction techniques. Let's examine the structure of this template:

- ***Title:*** The self-explanatory title of the decision.
- ***Status:*** The status of the decision, such as proposed, accepted, rejected, deprecated, superseded, etc.
- ***Context:*** Describe the observed issue that has motivated this decision or change.
- ***Decision:*** Describe the solution that was chosen and why.
- ***Consequences:*** Describe what is easier or more difficult due to this change.

The next section provides an example of the ADR of a team within NuCoverage.

Applying the Tools

NuCoverage must determine whether it should continue to run its business with the existing Monolithic application, or to use a different architecture to support their new white-label strategy. Its senior architects recommend using a Microservices architecture. It's unwise to make such a decision hastily. The business and technical stakeholders decided to use the Cynefin framework to gain a better understanding of the situation by means of a decision-making tool that fosters thorough analysis. Table 2.2 summarizes what they came up with.

Table 2.2 *Cynefin at Work to Determine the Risks and Complexities of Migrating from a Monolithic Architecture to a Microservices Architecture*

	Current Context	Dangers to Migration	Response to Dangers
Clear Known Knowns	Monolith is well modularized. Splitting modules from in-process to network calls is not a problem. Microservices common practices are well known to every member of the team.	Complacency and comfort. Team has a strong desire to dig into the distributed systems. No deep analysis made of the current state. Universal common practices don't exist.	Recognize the value and limitations of "common practices." Don't assume it's just a simple transition to Microservices. Don't oversimplify the solution.
Complicated Known Unknowns	Monolith may not be well modularized, but it's currently unknown for the whole scope of application. Common practices for Microservices aren't established in the community. The team has no experience in the migration from Monoliths to Microservices.	New approaches to migration are ignored by experts. Experts are overconfident in their own solutions or in the efficiency of past solutions. Views of experts are conflicting and they cannot agree on a common approach to migration.	Encourage external and internal stakeholders to challenge experts' opinions. Use experiments and scenarios-based analysis to force people to think differently and in a different way about the response.
Complex Unknown Unknowns	Monolith is not well modularized, and certainly it won't be easy to replace in-process communication by means of the network. No one agrees on best practices for migrating from the Monolith to Microservices. The team has no experience in the migration to Microservices, and it is difficult to identify real experts for the task.	Desire for accelerated resolution of the migration problem from stakeholders. Temptation to force a given decision on the rest of the stakeholders. Authoritarian response model to migration.	Be patient and allow the patterns to emerge through experimentation. Learn from failures and see which practices work and which don't work.
Chaotic Unknowables	Monolithic application is difficult to deploy and to restart on a daily basis. Many production bugs cause teams to take extra emergency support time to make the business run.	Relying too long on local practices that were discovered through use. No innovation. Relying too much on the "cult of the leader" architect to keep the business system running.	Set up parallel teams to work on the same domain problem. Challenge the current point of view and encourage innovation. Shift from this context to a complex one.

After significant analysis, all stakeholders involved agree that NuCoverage is in the *Complex* domain, because no one is confident that the current Monolithic application can be easily transformed to a Microservices-based distributed architecture. There appears to be no broad agreement on the common practices either in the community or among known Microservices experts.

One implication of using a Microservices architecture is that messages must be exchanged between each of the services. To make an architectural decision around this requirement, discussions around various mechanisms are held, which lead to the decision to initially use REST-based messaging. This decision is captured in the ADR shown in Listing 2.1.

Listing 2.1 *ADR That Captures the REST Message Exchange Decision*

```
Title: ADR 001: REST Message Exchange

Status: Experimental; Accepted

Context: Feed event messages to collaborating subsystems

Decision: Remain technology agnostic by using Web standards

Consequences:
Advantages: HTTP; Scale; Inexpensive for experiments
Disadvantages: Performance (but unlikely)
```

The current thought for the best way forward is found in the following points:

- Ensure an environment of experimentation with the option to fail, free from reprisals.
- Limit the experimentation scope to the current Monolithic application.
- Engage two experts from whom the team can learn, and to help avoid the likelihood of failure.

There is a sense of urgency in regard to establishing software architecture and design patterns that can be used by NuCoverage for the safe transformation of its Monolith to a Microservices architecture style, all in support of the new white-label insurance strategic initiative. Even so, the teams are making good progress and look forward to the journey ahead.

Summary

This chapter presented multiple strategic learning tools, including culture as a success enabler. Consider these essential for any business to achieve its strategic goal through differentiation and innovation. Making informed decisions is vital because the outcomes of ad hoc decisions are completely unreliable. Applying context to and forming insights into decisions is essential. To reinforce this, culturally safe experimentation and controlled failure are critical to better decision making, because Conway's Law is unforgiving of the inferior. As such, partitioning a problem space into smaller chunks feeds understanding, and using well-formed modules is essential to that effort. The recognition of business capabilities as modular divisions within and across which operations occur is core to every business that is expected to lead in revenue generation. Goal-based decisions are better than feature-based decisions, and Impact Mapping helps teams make strategic decisions on purpose. Some tools, such as the Cynefin framework, help with decision making. Others, such as ADRs, enable decisions along with long-term tracing.

The most salient points of this chapter are as follows:

- Understanding when decisions are most appropriate is essential to responsible decision making.
- The results of experimentation are an important source of knowledge for informed decision making.
- Beware of organizational culture and how it affects the safe use of experimentation and controlled failure as a decision-making tool.
- Recognizing business capabilities leads to applying modularity for better understanding and problem solving.
- Tools such as Cynefin and ADRs can help with decision making and long-term traceability.

The next chapter peers into events-first experimentation and discovery tools, which enables rapid learning and exploration that leads to innovations.

References

[Brooks] Frederick P. Brooks, Jr. *The Mythical Man-Month*. Reading, MA: Addison-Wesley, 1975.

[Cohn] Mike Cohn. *User Stories Applied: For Agile Software Development*. Boston, MA: Addison-Wesley, 2004.

[Conway] http://melconway.com/Home/Committees_Paper.html

[CT] "Book Reviews and Notes: *Teaching Thinking Skills: Theory and Practice*. Joan Baron and Robert Sternberg. 1987. W. H. Freeman, & Co., New York. 275 pages. Index. ISBN 0-7167-1791-3. Paperback." *Bulletin of Science, Technology & Society* 8, no. 1 (1988): 101. doi:10.1177/027046768800800113. ISSN 0270-4676.

[DrDobbs] Mary Poppendieck. "Morphing the Mold." August 1, 2003. https://www.drdobbs.com/morphing-the-mold/184415014.

[Hollnagel] Erik Hollnagel. "The ETTO Principle: Efficiency–Thoroughness Trade-Off or Why Things That Go Right Sometimes Go Wrong." https://skybrary.aero/bookshelf/books/4836.pdf.

[Impact] Gojko Adzic. *Impact Mapping: Making a Big Impact with Software Products and Projects*. https://www.impactmapping.org/.

[LA] James O. Coplien and Gertrud Bjornvig. *Lean Architecture: for Agile Software Development*. Hoboken, NJ: Wiley, 2010.

[LogFal] https://www.logicalfallacies.info

[Miller] "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two.

[Nygard-ADR] https://github.com/joelparkerhenderson/architecture_decision_record/blob/master/adr_template_by_michael_nygard.md

[Org-Culture] "Organizational Culture." https://en.wikipedia.org/wiki/Organizational_culture.

[Pisano] Gary P. Pisano. "The Hard Truth about Innovative Cultures." https://hbr.org/2019/01/the-hard-truth-about-innovative-cultures.

[Poppendieck] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Boston, MA: Addison-Wesley, 2003.

[TT] Matthew Skelton and Manuel Pais. *Team Topologies*. Portland, OR: IT Revolution, 2019.

[Tuckman] Bruce W. Tuckman. "Developmental Sequence in Small Groups." *Psychological Bulletin* 63 (1965): 384–399.

[TW-ICM] https://www.thoughtworks.com/radar/techniques/inverse-conway-maneuver

Chapter 3

Events-First Experimentation and Discovery

Extroverts thrive on communication, and many business executives, sales and marketing people, and personnel with other customer-facing roles match that description. They tend to be energized and reinvigorated by social interactions. Many extroverts find that their best insights are the result of unreserved expression and active listening. Software developers and programmers are often the extreme opposites—introverts with a strong impulse for silence and solitude, while also welcoming of the opportunity to work through challenging puzzles. For these professionals, large numbers of people, noisy conversations, and social requirements to participate can be draining.

Yet contrary to popular opinion, neither of these stereotypes is necessarily accurate. In fact, a few decades ago, it was estimated that among 450 CEOs, 70% of those top executives were introverts. That didn't often help those leaders in articulating their business visions and models in environments where "communicating is the staff of life" [Fortune-Campbell]. Whatever personality characteristics are considered strengths or weaknesses, individuals are what they are, and most are capable of rising above to meet complex tasks head-on. The ability to overcome weaknesses, even those that are born from strengths—extroversion that leads to impatience and overbearing dominance while engaged in knowledge sharing—lies within nearly everyone [Inverted-U]. It is imperative to do so because collaborative communication among distinctively intelligent and respectful people is an undeniable agency in strategic innovation and differentiation, which distinguishes one business from all others.

The question is, how can the broad range of personality types overcome communication challenges to achieve deep dives into collaborative communication, learning, and improved software construction? That's the topic tackled in this chapter.