Topic Backlog for

Certified Professional for Software Architecture®
Expert Level

2024.1-RC1-EN-20240125

Table of Contents

1. Topic Backlog

1.1. Impacts of DDD on the Business

Submitter: Hermann Woock

Status: Approved

Background

Over time, software projects typically become more cumbersome, fragile, and error-prone. The reason often lies in too many dependencies, poor cohesion, and high coupling. Domain-Driven Design (DDD) promises a solution to this problem.

However, DDD is more than just the cutting of components. Much of it is often overlooked, thus many of its strengths cannot be utilized: Deep Insight, Break Through, Supple Design, Large Structures.

Furthermore, DDD is a very agile approach, which, if implemented correctly, has a strong impact on workflows and company structures.

Task Description

The processing of the topic should demonstrate the problems that can be solved with DDD and what can be achieved with DDD.

However, the impact on the company should also be shown, or why DDD can fail. What conditions must be met for DDD to be beneficial?

The topic should be examined from the perspective of an organization, management, software developers, and users, among others.

References

Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Pearson ITP.

Vernon, V. (2013). Implementing Domain-Driven Design. Pearson ITP.

Group Size

The topic should be handled by a group of 3-5 people.

1.2. Documentation of Software Architectures

Submitter: Hermann Woock

Status: Approved

Background

Documentation is just as important in software development as it is unpopular. In the past, and under the influence of the waterfall model, often unwieldy paper deserts were created. In agile software development, on the other hand, documentation is often seen as a burden and unnecessary. New media and the recognition of the importance of clean architecture documentation shed new light on documentation.

What are the quality characteristics of good architectural documentation, and how are these insights introduced into a project? What experiences with the introduction and daily use can be passed on to other teams? What difficulties arose and how were they dealt with? Where are there still unresolved questions and problems?

Assignment

The following questions should be addressed:

- What forms are suitable for the documentation of software architectures (format, storage location, structure, used notations)?
- How is documentation used as a means of communication and not just as a repository?
- How are decisions and principles documented?
- How can the artifacts of software architecture documentation be linked with other artifacts (requirements, bug reports, test cases, etc.) (traceability)?
- How to identify the necessary contents and determine the level of detail of the documentation?

References

Zörner, S. (2021). Documenting and Communicating Software Architectures. Hanser Publishing

Starke, G. & Hruschka P. (2022). Arc42 in Action: Practical Tips for Architectural Documentation. Carl Hanser Publishing

Homepage of the ADR GitHub organization. https://adr.github.io/

Brown, S. The C4 model for visualising software architecture. https://c4model.com/

Group Size

The topic should be handled by a group of 3-6 people.

1.3. Use of Different NoSQL Databases in the Project

Submitter: Hermann Woock

Status: Released

Background

NoSQL databases are no longer a novelty. In many projects, they are standard, but in many others, they are not.

For those who have only dealt with relational databases, it's often hard to imagine how the use of NoSQL databases can be successful:

- What possibilities do these databases offer and what benefits do they bring?
- What types of NoSQL databases are there and what can they be used for?
- Are there dangers one should be aware of?

This topic is of interest to architects, developers, operations, DB administrators.

Assignment

These questions should be answered:

- What problems can be solved with NoSQL databases and what new problems might arise?
- How can one get the best use out of NoSQL databases?
- What skills and knowledge are required?
- How must NoSQL databases be considered in the architecture?

Other topics in this context include Event Sourcing, Eventual Consistency, Scalability, Fail-Safety, Licenses, or BASE.

A few successful practical examples with tips and tricks would be helpful.

References

Fowler, M., & Sadalage, P. J. (2012). NoSQL Distilled. Pearson Education.

Group Size

The topic should be handled by a group of 3-5 people.

1.4. Experiences with Event-Driven Architectures

Submitter: Hermann Woock

Status: Approved

Background

Interest in Event-Driven Architectures (EDA) didn't just start with the publication of Vernon's book "Implementing Domain-Driven Design," featuring a foreword by Eric Evans. Event-Driven Architectures offer a high degree of flexibility in software, whether in program execution or data capture. EDAs are already being successfully used in many places. Cloud providers offer solutions, and more and more frameworks are coming to the market. Monitoring, Business Intelligence (BI), and many other areas make use of EDAs.

Event-Driven Architectures help create flexible systems that can adapt to changes and make real-time decisions. Events are captured as soon as they occur from event sources such as Internet-of-Things (IoT) devices, applications, and networks, allowing event producers and event consumers to exchange status and response information in real-time. In this context, the concepts of Event Sourcing and CQRS are often discussed, presenting further challenges but also promising exciting advantages.

Assignment

These questions should be addressed:

- For what purposes is this architectural style suitable?
- What patterns and technologies can be used?
- How do the following points fit into this topic: CQRS, Event Sourcing, SAGA, Messaging, Reactive Programming, Kafka?
- How are the following problems solved?
  - Message Ordering
  - Idempotence
  - Versioning
  - Contents of Events

References

Vernon, V. (2013). Implementing Domain-Driven Design. Pearson ITP.

Domogalla, M. (2022, April 21). Software-Architektur.tv: Software Architecture with Events, Event Sourcing, and CQRS. Developer. https://www.heise.de/news/software-architektur-tv-Softwarearchitektur-mit-Events-Event-Sourcing-und-CQRS-7061126.html

Group Size

The topic should be handled by a group of 3 to 7 people.

1.5. Frontend Development in Large Projects

Submitter: Johannes Hochrainer

Status: Approved

Background

Large software systems are typically implemented by multiple development teams. While feature teams have become established in backend development, in many projects the frontend is developed by a single team. However, the concept of microservices is increasingly being adopted in frontend development as well: Independent teams work on a functional frontend component and can develop and deliver their component independently of other teams. There are various methodological and technical approaches to implementing this so-called microfrontend architecture.

Assignment

The following questions should be answered:

- How can a frontend be implemented flexibly and efficiently with various development teams? This includes, among others:
  - Micro Frontends
  - Self-Contained Systems
  - Backends for Frontends Pattern
  - Internal Organizational Component Libraries (similar to Angular Material)
  - Webpack Module Federation
- Monorepos (e.g., with Nx) *How do you convert an existing monolithic frontend into a cleanly modularized, parallel-developable frontend?

References

- Micro Frontends, https://micro-frontends.org/
- Self Contained Systems, https://scs-architecture.org/
- Backends for Frontends Pattern - Cloud Design Patterns, https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends
- Angular Material, https://material.angular.io/
- Nx Monorepo, https://nx.dev/
- Webpack Module Federation, https://webpack.js.org/concepts/module-federation/

Group Size

The topic should be handled by a group of 3 to 5 people.

ChatGPT

1.6. Machine Learning in Practice

Submitter: Hermann Woock

Status: Released

Background

Machine Learning (ML) is increasingly evolving from fiction to an indispensable technology. Those who do not engage with ML early may find themselves left behind later. However, investing in it is very risky.

The application areas are diverse: construction, automotive industry, finance, government agencies, logistics, marketing. However, implementation requires new talents and raises questions. No one should miss out on this development.

In the context of digitalization or increasing competition, new solutions are needed where ML can offer good assistance. In the field of Business Intelligence or Big Data, new methods are needed where ML can be supportive.

1.7. Assignment

The contribution should encourage finding an entry point into this topic.

There are many things to clarify:

- What problems can be solved with ML?
- What effort is involved?
- What skills and knowledge are required for implementation?
- What legal considerations must be taken into account?
- What are the success factors?

References

Group Size

The topic should be handled by a group of 3-7 people.

1.8. Migration to Microservices on Kubernetes

Submitter: Hermann Woock

Status: Approved

Background

Markets are becoming more complex, and software in the business and technology sector is getting more complicated. This often necessitates companies to seek the solution for their software projects in a Microservices Architecture (µService).

µServices are usually implemented with Kubernetes (K8s). Kubernetes is a highly complicated and complex framework. Developers are often overwhelmed with its installation, use, and operation, leading to projects that can easily fail or become unprofitable. An intense exchange between roles in requirement management, architecture, development, testing, and operation is therefore necessary. Lack of agility and rigid structures in the company make things more difficult. Initial approaches exist in DevOps, but more will be needed.

Assignment

Participants are to process the problems, approaches, and solutions from their projects and create a guideline.

References

Liebel, O. (2023). Scalable Container Infrastructures: The Handbook for Administrators (4th Edition). Rheinwerk Computing.

Group Size

The topic should be handled by a group of 6-7 persons.

1.9. Modeling Languages

Submitter: Johannes Hochrainer

Status: Approved

Background

For over 20 years, UML has been considered the standard for modeling software systems. Other modeling languages position themselves either as lightweight alternatives (e.g., C4), as complete replacements, or for a specific application area or domain (e.g., BPMN, SysML, ArchiMate).

The use of modeling languages is not always popular. They are often seen as too complicated or cumbersome. Nevertheless, large projects cannot be implemented without modeling. Often too late, it is recognized that the interrelationships in the project are no longer comprehensible, and a suitable model could have helped here. It would have been even better to start modeling early on to promote communication in the project.

Assignment

The working group should discuss the following questions and prepare the findings:

- What experiences have been made with UML and its alternatives in their own projects?
- When are informal models sufficient, and when is a formal modeling language recommended?
- How can development teams be convinced to use at least some UML?
- What are the proven practices in the use of UML?

References

- UML (Unified Modelling Language), https://www.uml.org/
- C4 model, https://c4model.com/
- FMC (Fundamental Modelling Concepts), http://www.fmc-modeling.org
- ArchiMate, https://www.opengroup.org/archimate-forum/archimate-overview
- Semantics of a Foundational Subset for Executable UML Models (fUML)
- Action Language for Foundational UML (ALF)
- Precise Semantics of UML Composite Structures (PSCS)
- Precise Semantics of UML State Machines (PSSM)

Group Size

The topic should be handled by a group of 4-7 persons.

1.10. Are SOLID Principles Still Relevant?

Submitter: Johannes Hochrainer

Status: Approved

Background

In recent years, there have been increasing voices claiming that the SOLID principles are no longer contemporary. For instance, in 2021, Dan North caused a stir when he claimed that every single SOLID principle was wrong. After some time, he followed up and proposed his so-called "CUPID-Properties" as an alternative:

- Composable
- Unix Philosophy
- Predictable
- Idiomatic
- Domain-based

Assignment

The working group should clarify the following questions:

- What are the criticisms of SOLID, and can they be scientifically validated?
- Does CUPID meet all the requirements to serve as a guide in software development?
- Are there commonalities between SOLID and CUPID?
- Are there areas of application that are particularly suitable for SOLID or for CUPID?
- Are there other collections of principles that developers can follow?

References

- Principles Of Object-Oriented Design, http://wiki.c2.com/?PrinciplesOfObjectOrientedDesign
- CUPID—the back story, https://dannorth.net/2021/03/16/cupid-the-back-story/
- CUPID—for joyful coding, https://dannorth.net/2022/02/10/cupid-for-joyful-coding/

Group Size

The topic should be handled by a group of 3-5 persons.