Volume XXVII 2024
ISSUE no.2
MBNA Publishing House Constanta 2024

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Structurizr DSL: A Tool for Expressive Architectural Documentation in Maritime Software Projects

To cite this article: Andrei Băutu, Anca Atodiresei and Elena Băutu, Scientific Bulletin of Naval Academy, Vol. XXVII 2024, pg. 220-224

Submitted: 29.03.2024
Revised: 28.11.2024
Accepted: 05.12.2024

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-24-I2-026

Structurizr DSL: A Tool for Expressive Architectural Documentation in Maritime Software Projects

Andrei Băutu[1], Anca Atodiresei[2] and Elena Băutu[3]

[1] Romanian Naval Academy, Romania, andrei.bautu@anmb.ro
[2] Romanian Naval Academy, Romania, anca.atodiresei@anmb.ro
[3] "Ovidius" University, Romania, ebautu@univ-ovidius.ro

Abstract. This article discuses the usage of Structurizr Domain-Specific Language (DSL) as a tool for generating expressive architectural documentation in maritime software projects. By defining system architecture using code, software engineers can maintain consistency, version control, and automate the generation of architecture diagrams. The article provides an overview of the Structurizr DSL and outlines advantages and limitations of using it during development of maritime software.

1. Introduction

Architectural documentation plays an essential role in the development and maintenance of software systems, ensuring clarity of communication between the parties involved and facilitating system adaptation and expansion. In the field of maritime software, the complexity and specificity of applications require well-structured and up-to-date architectural documentation capable of clearly describing the interactions between components, data flows and compliance with maritime standards. This type of documentation becomes indispensable for the development of robust and secure systems, such as those used in automatic navigation, ship monitoring or fleet management.

Domain-Specific Languages (DSL) represent an innovative approach to generating architectural documentation. Unlike general-purpose programming languages, a DSL is designed to directly model domain-specific structures and concepts [1]. Using a DSL for architectural documentation, such as Structurizr DSL, allows the description of the system architecture through code, thus offering numerous advantages. These include maintaining documentation consistency, version control, and automating diagram generation.

Structurizr DSL was designed to simplify and improve the architectural documentation process, making it a suitable tool for marine software projects where accuracy and clarity are critical. As a DSL that easily integrates with popular programming languages and development platforms, Structurizr DSL enables engineers to build expressive and precise architectural documentation based on precise and maintainable specifications [2][3].

Many software architecture tools are currently available to software engineering teams [4][5]. The purpose of this article is to explore in the capabilities of Structurizr DSL in the context of maritime software projects. The article will provide an overview of the syntax of this language and how it can be used to effectively document the architecture of maritime applications. During the presentation, the advantages of using a DSL dedicated to architectural documentation will be discussed, as well as the limitations and potential directions for improvement.

The results presented in this article are intended to demonstrate how the use of Structurizr DSL contributes to the development of accurate and relevant architectural documentation in the context of maritime projects, providing a scalable and easy-to-maintain solution to the challenges of this field.

The following sections discuses the principles of architectural documentation and the challenges faced in the maritime industry. Section 3 presents a brief overview of the syntax and features of Structurizr DSL, illustrating its capabilities in modeling complex systems. In Section 4, we analyze the benefits and limitations of using Structurizr DSL within the context of maritime software development.

2. Domain Specific Languages in the context of architectural documentation

In complex software projects, maintaining consistent and clear architectural documentation is an ongoing challenge. Domain-Specific Language (DSL) were developed to provide domain-specific solutions, simplifying tasks that, with general-purpose languages, would have been time-consuming and error-prone. A DSL is essentially a language designed to model domain-specific concepts and structures, making it easy to express intents and structures in an intuitive and efficient way for users of that domain.

In architectural documentation, using a DSL brings several significant advantages:

- Clarity and expressiveness: A DSL can capture domain specificities, allowing users to directly describe architectural structures without ambiguity. For example, a DSL dedicated to architectural documentation will have terms and constructs that reflect concepts such as components, relationships, and data flows, making the documentation more intuitive and easier to understand for those who use it.
- Documentation automation and consistency: Compared to manual methods, using a DSL allows automation of documentation generation, which ensures consistency and eliminates common errors. Architectural frameworks can be updated automatically and documentation stays in sync with the source code, which is essential in long-running software projects.
- Version control and maintenance: Being code-based, a DSL can be managed using version control systems such as Git. This allows changes in the architecture to be tracked over time, facilitating audit and maintenance processes. Maintenance also becomes simpler, as architectural changes are reflected in the documentation almost instantly.
- Integration with other tools: Most DSLs are designed to integrate with other technologies and languages, making them easier to use within broader software development processes. Structurizr DSL, for example, can automatically generate diagrams and can be used in conjunction with general-purpose programming languages such as Java or C# to provide detailed and customizable documentation.

In the context of maritime software projects, architectural complexity is amplified by domain-specific requirements: safety standards, integration with navigational equipment, and compliance with maritime regulations. Maritime software applications such as ship monitoring, maritime traffic management and weather forecasting require architectures to facilitate system security, interoperability and scalability.

A dedicated DSL for architectural documentation allows designers and engineers to model the structure of marine software in a way that meets these requirements and facilitates architectural compliance analysis. For example, such a language can help clearly model interactions between safety-critical components, describe data flows, and identify integration points with other external systems, such as sensors and communication systems.

Structurizr DSL is an example DSL designed to support the creation of expressive and structured architectural documentation. Unlike other traditional approaches, such as UML diagrams [6] or manual description models, Structurizr DSL enables the automatic generation of diagrams and the description of architectural structures through code. This method offers the advantage of consistency

and automation, and in the maritime field it proves to be a suitable choice due to its ability to describe the complexity and particularities of maritime applications.

3. Structurizr DSL

Structurizr DSL (Domain-Specific Language) is a domain-specific language designed to enable clear and concise architectural documentation through code. Developed as part of the Structurizr platform, this DSL provides developers and architects with an efficient way to describe the architecture of software systems by automatically generating diagrams that illustrate the structure and component interactions. In this way, Structurizr DSL combines the precision and flexibility of domain-specific languages with the advantages of automation, consistency, and integration with version control.

Structurizr DSL is designed to be easy to understand and use, even by those without advanced programming knowledge. Its syntax enables the rapid definition of architectural components, their relationships, and levels of abstraction, as well as the generation of multiple views of the same architecture.

Among the essential concepts of Structurizr DSL are:

- System Context: Allows the software system to be described in a holistic way, highlighting interactions with external systems and users.
- Containers: Structurizr DSL uses the concept of containers to delineate major subsystems or modules within the architecture. For example, a container can represent a web application, a database service, or a data processing module.
- Components: Within each container, Structurizr DSL allows the definition of individual components, such as classes, services, or libraries, highlighting their roles and how they work together to perform system functionality.
- Relationships: Relationships between components, containers, and external systems can be clearly specified, indicating data flow and architectural dependencies.

The simplified code structure allows a concise definition of the architecture. For example, the following Structurizr DSL code snippet illustrates a basic architectural description for a maritime software system:

```
workspace {
    model {
        user = person "Maritime operator" {
            description "Uses the software for ship monitoring."
        }

        system = softwareSystem "Maritime monitoring system" {
            description "Monitors in real time the status of ships."

            webapp = container "Web application" {
                description "Allows operator to view monitoring data."
                technology "ReactJS, Node.js"
            }

            database = container "Data base" {
                description "Stores data for each ship."
                technology "PostgreSQL"
            }

            user -> webapp "Access monitoring data"
            webapp -> database "Query and update data"
        }
    }
```

```
    views {
        systemContext system {
            include *
            autolayout lr
        }

        container system {
            include *
            autolayout tb
        }

        theme default
    }
}
```

The code above describes a simple marine monitoring system, which includes a user (the marine operator) and a marine monitoring system (composed of a web application and a database). In addition, the Structurizr DSL allows defining the relationships between these components, making it easier to understand how users and systems interact.

Structurizr DSL can be used with popular programming languages such as Java, C# or Python due to its ability to generate documentation independent of the system's implementation language. This allows developers to integrate Structurizr DSL directly into their workflows and use version control tools to manage and track architectural changes.

An important aspect of integration is the ability of Structurizr DSL to be used in continuous development (CI/CD) environments, so that architectural documentation is automatically generated and updated with each architecture change. This automation ensures that the documentation is always in sync with the source code and significantly reduces the risk of discrepancies between code and documentation.

4. Applications of Structurizr DSL in maritime software projects

Structurizr DSL offers a set of advantages that make it ideal for use in maritime software projects:

- Accuracy and clarity of documentation: Structurizr DSL enables the description of architecture at the system, container, and component levels, facilitating a deep understanding of the structures and interactions between different parts of the system. In maritime software projects, where architectures can include complex modules such as navigation systems and monitoring sensors, this clarity is crucial.
- Automation of diagram generation: Architectural documentation can be automatically generated, eliminating the need for manual diagramming and ensuring constant updating. In the maritime field, where safety regulations and requirements can evolve rapidly, this feature becomes valuable, reducing the time and resources needed to update documentation.
- Flexibility in change management: Structurizr DSL enables easy management of architectural changes, and version control provides clear traceability of changes over time. Thus, when new requirements or changes in regulatory rules arise, the documentation can be efficiently adjusted, maintaining the history of changes and facilitating the audit of the architecture.

Some limitations also need to be taken into account when planning to use this DSL:

- The Learning Curve: Structurizr DSL is relatively easy to use, but teams with no experience using DSLs or documenting architecture through code may experience initial difficulties. This learning curve can be an impediment to rapid adoption of this tool, especially in projects that require a rapid transition to a new documentation methodology.
- Limitations in Documenting Operational Details: Structurizr DSL focuses on describing the architectural structure and relationships between components, but does not provide detailed support for documenting operational aspects such as specific network configurations,

performance parameters, or monitoring system activity. In marine projects, where safety and performance are critical, this limitation may require supplementing the architectural documentation with other tools or additional descriptions to cover these details.

- Lack of Flexibility for Custom Views: Some projects may require highly customized architectural visualizations to meet specific presentation requirements. Structurizr DSL does not always allow fine-grained control over the styling and customization of visualizations.

Therefore, Structurizr DSL represents a valuable tool for architectural documentation in maritime software projects, combining the advantages of domain-specific languages with the automation and flexibility required in this dynamic domain.

5. Conclusions

This article explores the use of Structurizr DSL as an architectural documentation tool in maritime software projects, highlighting both the benefits and limitations of this approach.

Using Structurizr DSL significantly improves the efficiency of the architectural documentation process. By modeling system architecture in an expressive way and automatically generating diagrams, teams can reduce the time and effort required to update documentation. The ability to integrate architectural changes in an efficient way is crucial in the dynamic environment of software development. Structurizr DSL enables teams to quickly respond to changes while keeping documentation up-to-date and relevant.

Although Structurizr DSL has many advantages, it is important to recognize its limitations, such as the learning curve and lack of operational detail. These limitations must be considered when evaluating its use in specific projects.

References

[1] Fowler, Martin. Domain-specific languages. Pearson Education, 2010.

[2] Chiu, Kevin, Sean Marquez, and Sharanabasaweshwara Asundi. "Model Based Systems Engineering with a Docs-as-Code Approach for the SeaLion CubeSat Project." Systems 11.7 (2023): 320.

[3] Nicacio, Jalves, and Fabio Petrillo. "An approach to build consistent software architecture diagrams using devops system descriptors." Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. 2022.

[4] Rashidi, Hassan, Zahra Rashidi, and Zeynab Rashidi. "Software Architecture Tools-A Classification and Survey with Recommendation for an Organization." Journal of Computing and Security 10.2 (2023): 61-81.

[5] Darvas, Adam, and Raimund Konnerth. "System architecture recovery based on software structure model." 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE, 2016.

[6] Fowler, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2018.