Empirical evidence of using C4 design method in development of edge-cloud systems

Jānis Grabis[1,*,†], Jānis Kampars[1,†]

[1]Information Technology, Riga Technical University, Zunda krastmala 10, Riga, Latvia

Abstract

The paper reports the findings of case studies on the application of the C4 software architecture development method. It focuses on development of edge and cloud systems and aims to analyze peculiarities of collaborative software design and practices of designing the edge and cloud systems. Three applied research projects are considered on the development of smart systems. The C4 method proved valuable in establishing a common understanding on the development of distributed systems. However, there were significant differences among the projects in a way the method was applied and a level of details represented in software architecture. That is at least partially influenced by the organizational structure of development teams. The research is envisioned to lead to a set of recommendations and best practices for software modeling, applying the C4 method and the collaborative development of distributed systems in general.

Keywords

software architecture, collaborative modeling, edge and cloud architecture

1. Introduction

Many modern information systems are characterized by a high level of complexity due to their distributed nature, data heterogeneity, scalability requirements and other factors. Systems architecture provides an overview of the envisioned technical solution by defining key components and their relationships. It helps developers to conceptualized and reason about technical challenges and plays an important role in communication of the solution among stakeholders, allocation of work to development teams, analyzing non-functional requirements and evaluation of design choices.

Architecture has been recognized as a way to coordinate software development activities [1]. The coordination is of major importance in development of distributed systems. Edge and cloud systems are inherently distributed and increasingly find applications in areas requiring advanced sensing capabilities, for example, in smart manufacturing [2].

Visualization of software architecture facilitates collaboration and comprehension [3]. Infrastructure as a code specifications such as Docker compose can be transformed in architecture diagrams for their specific application domain [4]. Collaboration and support for domain specific languages are among the requirements the teams have towards model driven development [5]. The C4 method [6] has emerged as a lightweight method for development of system architecture. It provides a simple to follow methodological guidance for step-wise elaboration of the architecture and the Structurizr development tool for on-line collaborative work [7].

We have applied the C4 method in several applied research projects conducted in collaboration between a university and industry. These projects deal with development of software systems for sensing enterprises and involve using IoT, big data and edge and cloud technologies. The development work is divided between the university and industry partners (more than one industry partner in some cases). This paper aims to summarize experiences accumulated during the system architecting phase and on usage of the C4 method in particular.

BIR-WS 2025: BIR 2025 Workshops and Doctoral Consortium, 24th International Conference on Perspectives in Business Informatics Research (BIR 2025), September 17-19, 2025, Riga, Latvia

*Corresponding author.

†These authors contributed equally.

grabis@rtu.lv (J. Grabis); janis.kampars@rtu.lv (J. Kampars)
0000-0003-2196-0214 (J. Grabis); 0000-0003-0045-5593 (J. Kampars)

The overall goal of this research is to understand challenges and best practices of development of architecture of distributed systems and edge and cloud systems in particular. The evaluation is performed with respect to key features and components of the edge and cloud system architectures reported in literature. In future research, that would lead to a set of recommendations one could use to guide the system development process in the collaborative environment. Case study research [8] is adopted as a research method. System architecture documentation as well as messages exchanged among the developers during the development process are analyzed from three applied research projects.

The rest of the paper is organized as follows. Section 2 briefly reviews the C4 method and analyzes existing work on architecture of edge and cloud systems to identify typical features of this architecture. The case study is reported in Section 3, and Section 4 concludes.

2. Background

2.1. C4 Method

C4 is a method for development of software architecture [6]. It allows for a gradual decomposition of the system, defining the main components and designing and detailing the specific services sequentially. The C4 method is an "abstraction-first" approach to diagramming software architecture, and it attempts to mimic a way software architects and developers think about software design and development [6]. It uses a limited set of constructs (i.e., person, software system, container and component) arranged in four levels of elaboration. The design process starts with a context diagram (a landscape diagram can be added as an optional Level 0 diagram), proceeds with a container diagram and a set of component diagrams for each container. A detailed design can be elaborated at Level 4 using variety of software engineering techniques such as UML. This paper concerns levels one to three.

C4 starts with identification of the core software system to be built and defines its context in the Context diagram (Level 1). This diagram shows users of the software system and key related systems. The Container diagram (Level 2) splits the software system into applications and data stores required to have a running system. It is useful for showing the overall allocation of responsibilities to specific parts of the software system. The component diagram (Level 3) is created for each container and shows major building blocks and their interactions. A component is a grouping of related functionality and has a clear access interface. Technology choices and implementation details can be represented at this level.

The C4 Domain specific language (DSL) allows to describe software architecture following the model-as-code principle. Elements of the software architecture and their relationships that define the software model are declared. These elements are subsequently arranged into views displayed as diagrams. Various visual aids can be added to the diagrams.

The C4 method was selected primarily because of its ability to clearly allocate responsibilities to parties involved as well as due to its light-weight approach and quick ramp-up. Other advantages of the method are a collaborative on-line workplace, version management, getting ready methodological guidance and DSL.

2.2. Edge and Cloud System Architecture

Edge and cloud computing combines an ability to process data at their origin with capabilities of cloud computing. It finds applications in IoT solutions, Industry 4.0 and smart systems. It follows a distributed computing paradigm requiring integration of different data processing technologies. The key requirements towards the edge and cloud architecture include proactive control in real time, decoupling of control from physical devices, interoperability, visualization, data availability, scalability, operational/long-term data storage and efficient usage of computational resources [2]. The review of such edge computing architectures as FAR-Edge, ECC and ICC [9] reveals that their main features are Connectivity and communication, Device management, Data collection, analysis and performance, Scalability, Compliance with standards and Security. Key components of the reference architecture can be allocated to IoT, Edge and Business Solution (Cloud) layers [10]. The IoT layer includes sensors, actuators and controllers and concerns data collection and initial processing. The Edge layer includes an edge server for local processing and an edge gateway for data integration. The Cloud layer performs advanced data processing and provides front-end service. The layers are connected by a vertical security layer. At the component level, there are best practices to ensure resilience of edge and cloud systems [11]. These include redundancy, rerouting, verification and access control.

3. Case Study

A case study research is conducted to observe application of the C4 method in practice. The cases selected are three university and industry joint applied research projects. All projects deal with development of smart systems. The university and companies involved have their own separate development teams with separate management, and the chief system architect was from the university's team. Different companies were involved in the projects while the core university's team was the same although two persons served as the chief architect in these projects.

3.1. Research Question

The following research questions are formulated:

- Do system architects follow best practices of design of edge and cloud systems?
- What are peculiarities of using the C4 method?
- Does using the C4 method facilitate collaboration among the teams?

The first research question compares the models created with the reference model reported in literature with focus on types of components used. The second question aims to identify differences and commonalities of using the C4 method across the projects. Finally, one is interested into utility of using C4 or similar methods, and the third question aims to provide initial insights on value of the method.

3.2. Cases and Data Collection

One completed and two on-going projects are selected for exploration. **Project 1**: The project aims to create a platform for ensuring a safe working environment that integrates advanced information and communication technologies and biotechnologies. The platform combines business continuity planning, IoT, computer vision, machine learning and wastewater analytics technologies for comprehensive Covid-19 and other infections risk assessment, mitigation and prevention in workplaces where the nature of work limits remote working options, such as shift-based manufacturing companies. **Project 2**: The project concerns work-in-progress inventory management in the production environment. A production operator receives information about production orders what includes a list of materials required to produce an end-product. Sensing technologies such as computer vision and RFID provide data on location of the materials. These data are aggregated and processed to create a materials pick-up route, which is executed to set-up a production step. The production step yields an intermediate product, which should be placed in a suitable location on the shop-floor for the next production step. A software system is being developed as a service, which can be integrated with the existing ERP system used by the manufacturing company. **Project 3**: The aim of the project is to develop an edge-cloud based AI-driven power resource management and balancing platform for office buildings, which would be compatible with the building's local systems and external data sources, and would maximise the use of renewable resources, minimise electricity costs while ensuring the desired level of user convenience. Although software systems are being developed in different fields of application, they all rely on intelligent processing of sensor data and combine edge and cloud capabilities for efficient data processing.

Complete software architecture documentation was available for all three projects. That included diagrams and descriptions created as well as the complete version history of updates made in the models and diagrams during their continuous elaboration. A drawing tool based on templates was used in Project 1, while Structurizr on-premise installation was used in Project 2 and Project 3.

Project 1 is completed and Project 2 and Project 3 are currently entering the implementation phase.

Table 1
Identifiable edge and cloud components in projects' software architecture.

Component	Project 1	Project 2	Project 3
Edge devices	Large variety, explicitly specified	Three types, explicitly specified	Abstracted as BMS
Edge server	Limited data processing at edge	Video stream processing	Central edge system
Edge gateway	ChirpStack server	Edge API	Central edge system
Operational storage	Redis cache server	Edge data storage	X
Data broker	Kafka is used for exchange with external system	Fully-fledged message broker	
Long-term storage	MongoDB	Postgre	Database management system
Cloud analytics	Custom build using NodeJS	Prediction and optimization ML models	
Front-end	A part of the legacy application	Dedicated web application	Front-end system

3.3. Results and Discussion

Level 1 to Level 3 diagrams were elaborated in all projects. An optional Level 0 landscape diagram was created for Project 3 because the participants required a more detailed understanding of the application context. The architecture was strongly influenced by a three tier client-server system with easily recognizable data, application and presentation layers (Fig.1). That can be explained by having a legacy time management application at its core. The architecture is also characterized by a wealth connections with external systems. The architectures in Project 2 and Project 3 start with explicit separation of edge and cloud systems (Fig.2) and are influenced by the microservice approach.

Structurizr was not used in Project 1. Although the main syntax requirements were observed, diagrams also included additional pictograms and variety of explanatory comments. This approach was possible because the chief architect centralized the architecture management. A more distributed approach was taken in other projects and following DSL allowed to keep diagrams consistent and readable. Typical edge and cloud architecture components can be identified in all three projects (Table 1). Project 2 has the most advanced edge processing and limited number of edge devices, and the sensor data flow is also handled using the edge server using dedicated components for each sensor type. Project 3 stores and updates device profiles in the edge and has a central edge system responsible for data gathering and processing for a specific location (i.e., building). The cloud system is responsible for advanced analytical features using machine learning. Many technology choices are identified in the software architecture for Project 1, while Project 3 mainly uses generic terms, e.g., user interface wise just an abstract Front-end system component is provided. All three projects use different approaches to showing relationships. Technical terms specifying interfaces and protocols are used in Project 1 (Fig.3). Project 2 operates with a limited set of verbs and nouns to describe exchange of messages in style close to REST. Textual descriptions are in used in Project 3 (Fig.4). Project 3 also has the largest number of connections. Many connections were defined in the model and subsequently excluded from the views to improve readability. Many of the connections were made to service components such as logging and there were discussions of the most suitable ways to represent such service components. It is observed that diagrams are becoming more verbose as one moves to the component level. That is especially noticeable in elaboration of analytical features what might be explained by limited DSL expressiveness or uncertainty about solutions to be developed. Detailed descriptions were added to elements and connections in the diagrams in Project 3. The development company used subcontractors in this project and there very several additional rounds of reviews and approvals. None of the software architectures created explicitly accounted for security management what is an essential part for edge and cloud systems. Perhaps, the hierarchical structure of C4 is not well suited to representation of cross-cutting features. Nevertheless, Project 1 and Project 2 included components for device management in the

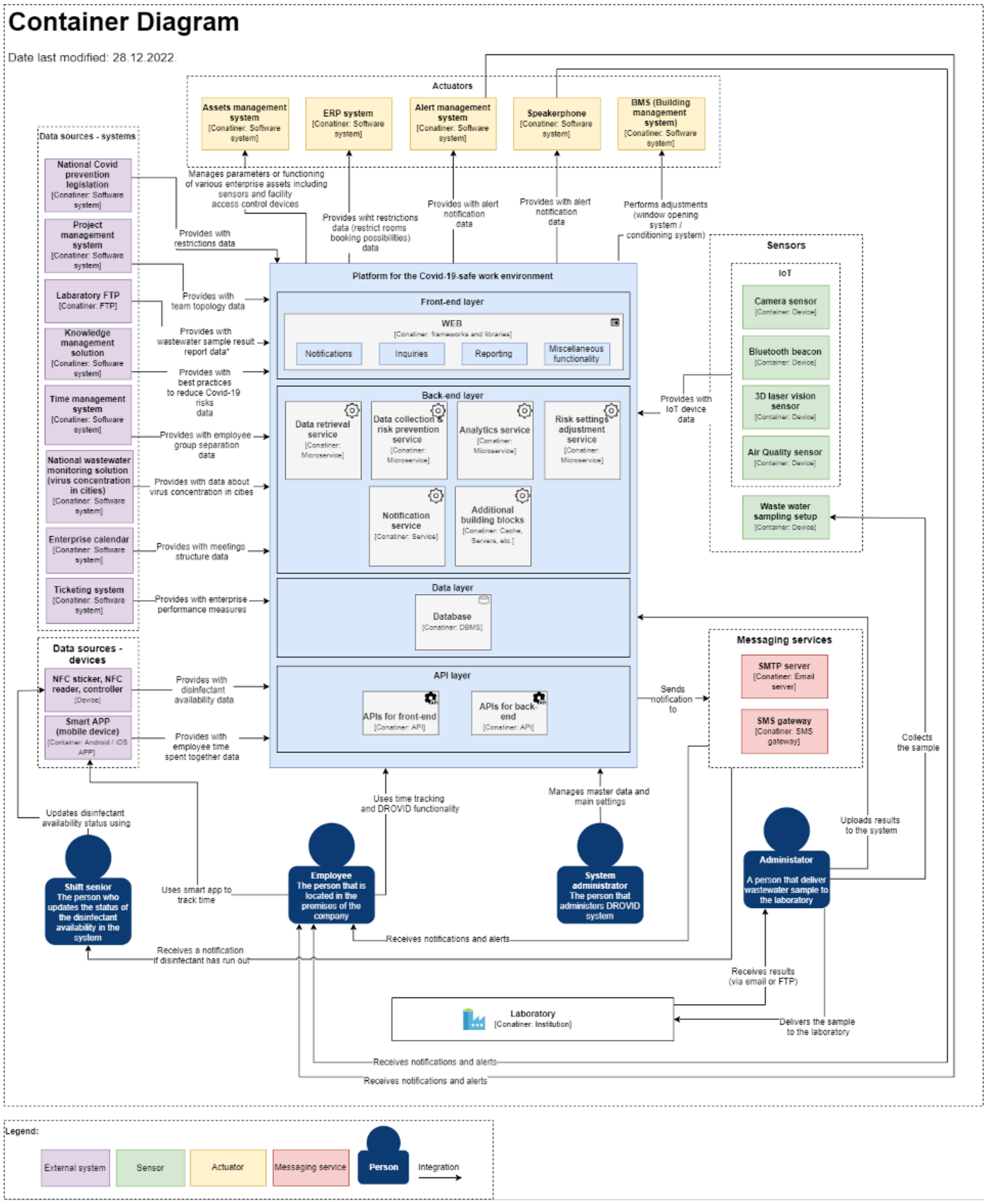
Container Diagram
Date last modified: 28.12.2022
Actuators
Assets management system [Conatiner: Software system]
ERP system [Conatiner: Software system]
Alert management system [Conatiner: Software system]
Speakerphone [Conatiner: Software system]
BMS (Building management system) [Conatiner: Software system]
Data sources - systems
National Covid prevention legislation [Conatiner: Software system]
Project management system [Conatiner: Software system]
Labaratory FTP [Conatiner: FTP]
Knowledge management solution [Conatiner: Software system]
Time management system [Conatiner: Software system]
National wastewater monitoring solution (virus concentration in cities) [Conatiner: Software system]
Enterprise calendar [Conatiner: Software system]
Ticketing system [Conatiner: Software system]
Data sources - devices
NFC sticker, NFC reader, controller [Device]
Smart APP (mobile device) [Container: Android / iOS APP]
Manages parameters or functioning of various enterprise assets including sensors and facility access control devices
Provides wiht restrictions data (restrict rooms booking possibilities) data
Provides with alert notification data
Provides with alert notification data
Performs adjustments (window opening system / conditioning system)
Provides with restrictions data
Provides with team topology data
Provides with wastewater sample result report data*
Provides with best practices to reduce Covid-19 risks data
Provides with employee group separation data
Provides with data about virus concentration in cities
Provides with meetings structure data
Provides with enterprise performance measures
Provides with disinfectant availability data
Provides with employee time spent together data
Platform for the Covid-19-safe work environment
Front-end layer
WEB [Conatiner: frameworks and libraries]
Notifications
Inquiries
Reporting
Miscellaneous functionality
Back-end layer
Data retrieval service [Conatiner: Microservice]
Data collection & risk prevention service [Conatiner: Microservice]
Analytics service [Conatiner: Microservice]
Risk settings adjustment service [Conatiner: Microservice]
Notification service [Conatiner: Service]
Additional building blocks [Conatiner: Cache, Servers, etc.]
Data layer
Database [Conatiner: DBMS]
API layer
APIs for front-end [Conatiner: API]
APIs for back-end [Conatiner: API]
Sensors
IoT
Camera sensor [Container: Device]
Bluetooth beacon [Container: Device]
3D laser vision sensor [Container: Device]
Air Quality sensor [Container: Device]
Waste water sampling setup [Container: Device]
Provides with IoT device data
Messaging services
SMTP server [Conatiner: Email server]
SMS gateway [Conatiner: SMS gateway]
Sends notification to
Collects the sample
Uploads results to the system
Updates disinfectant availability status using
Uses time tracking and DROVID functionality
Manages master data and main settings
Shift senior The person who updates the status of the disinfectant availability in the system
Uses smart app to track time
Employee The person that is located in the premises of the company
System administrator The person that administers DROVID system
Administator A person that deliver wastewater sample to the laboratory
Receives notifications and alerts
Receives a notification if disinfectant has run out
Receives results (via email or FTP)
Laboratory [Conatiner: Institution]
Delivers the sample to the laboratory
Receives notifications and alerts
Receives notifications and alerts
Legend:
External system
Sensor
Actuator
Messaging service
Person
Integration

Figure 1: Container diagram for Project 1

cloud.

Getting started materials provided by C4 and a brief introduction sessions proved sufficient for companies' developers to pick-up the core principles and concepts of the method. The choice of the light-weight approach proved to be helpful because industry architects and developers were much less used to structured modeling methods and were primarily relying on free-form visualizations or low-level of abstraction forms.

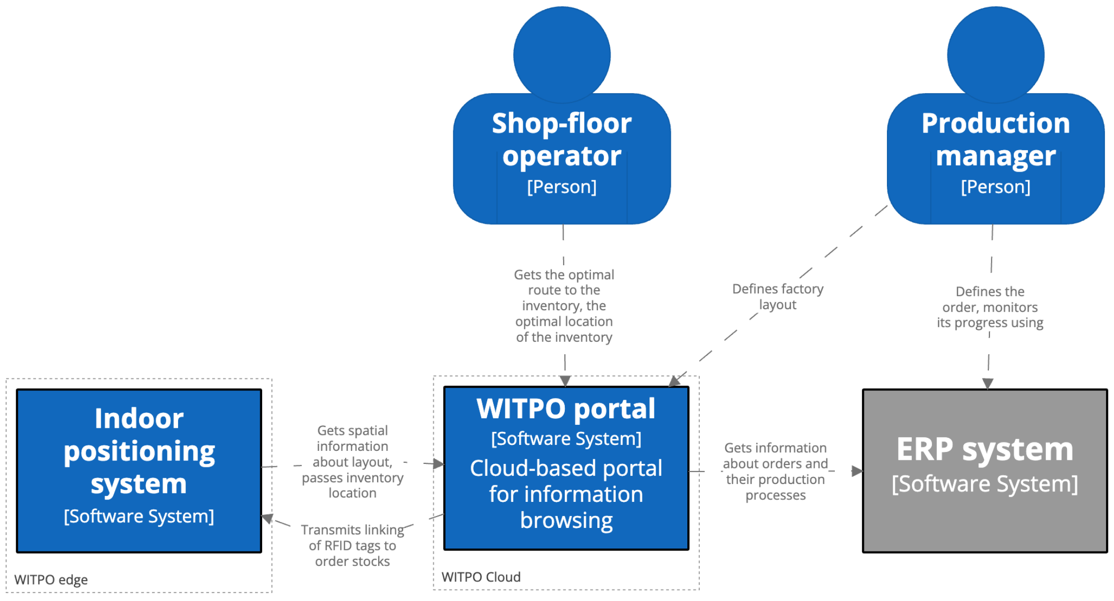
Shop-floor operator
[Person]
Production manager
[Person]
Gets the optimal route to the inventory, the optimal location of the inventory
Defines factory layout
Defines the order, monitors its progress using
Indoor positioning system
[Software System]
WITPO edge
Gets spatial information about layout, passes inventory location
Transmits linking of RFID tags to order stocks
WITPO portal
[Software System]
Cloud-based portal for information browsing
WITPO Cloud
Gets information about orders and their production processes
ERP system
[Software System]

Figure 2: Context diagram for Project 2

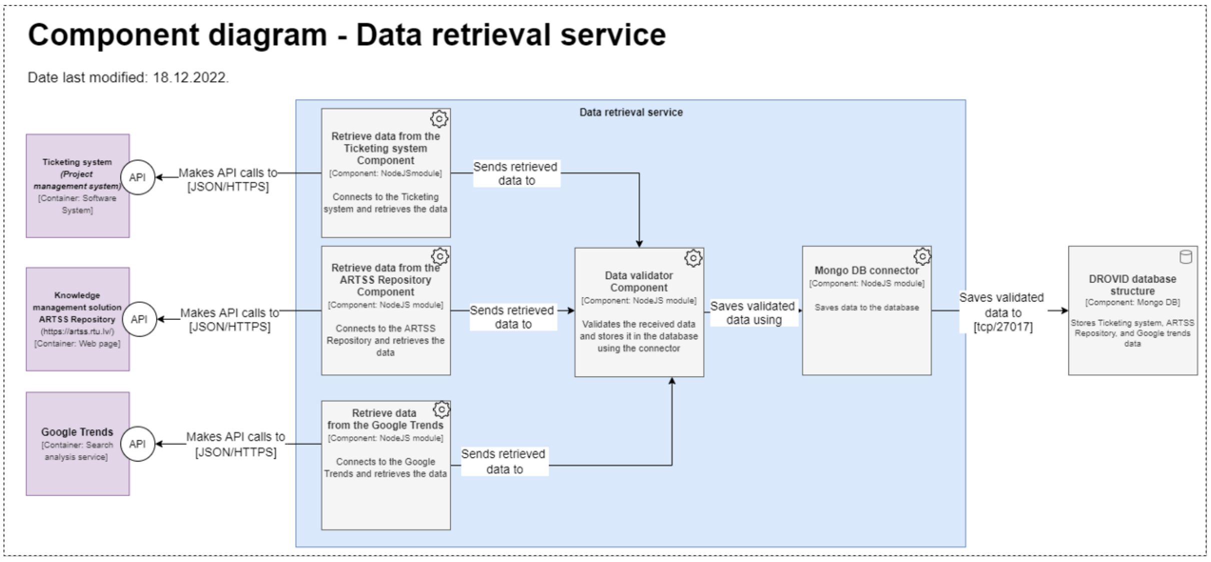
Component diagram - Data retrieval service
Date last modified: 18.12.2022.
Data retrieval service
Ticketing system (Project management system) [Container: Software System]
API
Makes API calls to [JSON/HTTPS]
Knowledge management solution ARTSS Repository (https://artss.rtu.lv/) [Container: Web page]
Google Trends [Container: Search analysis service]
Retrieve data from the Ticketing system Component [Component: NodeJSmodule] Connects to the Ticketing system and retrieves the data
Retrieve data from the ARTSS Repository Component [Component: NodeJS module] Connects to the ARTSS Repository and retrieves the data
Retrieve data from the Google Trends [Component: NodeJS module] Connects to the Google Trends and retrieves the data
Sends retrieved data to
Data validator Component [Component: NodeJS module] Validates the received data and stores it in the database using the connector
Saves validated data using
Mongo DB connector [Component: NodeJS module] Saves data to the database
Saves validated data to [tcp/27017]
DROVID database structure [Component: Mongo DB] Stores Ticketing system, ARTSS Repository, and Google trends data

Figure 3: Data retrieval service component diagram for Project 1

4. Conclusion

The paper has explored empirical experiences on using the C4 software architecture method in three applied research projects on development of edge and cloud systems. The exploration was guided by a brief literature review on typical characteristics of edge and cloud systems. The overall observation is that application of the C4 method achieved its stated objective of coordinating software architecture development among development teams involved.

Although the typical features of the edge and cloud system are present in all diagrams, significant variations were identified among the projects. The software architecture is more detailed in Project 1, and it was used to guide the development process without much of an additional work on detailed design. It is possible that models in Projects 2 and 3 will be further elaborated as the projects are

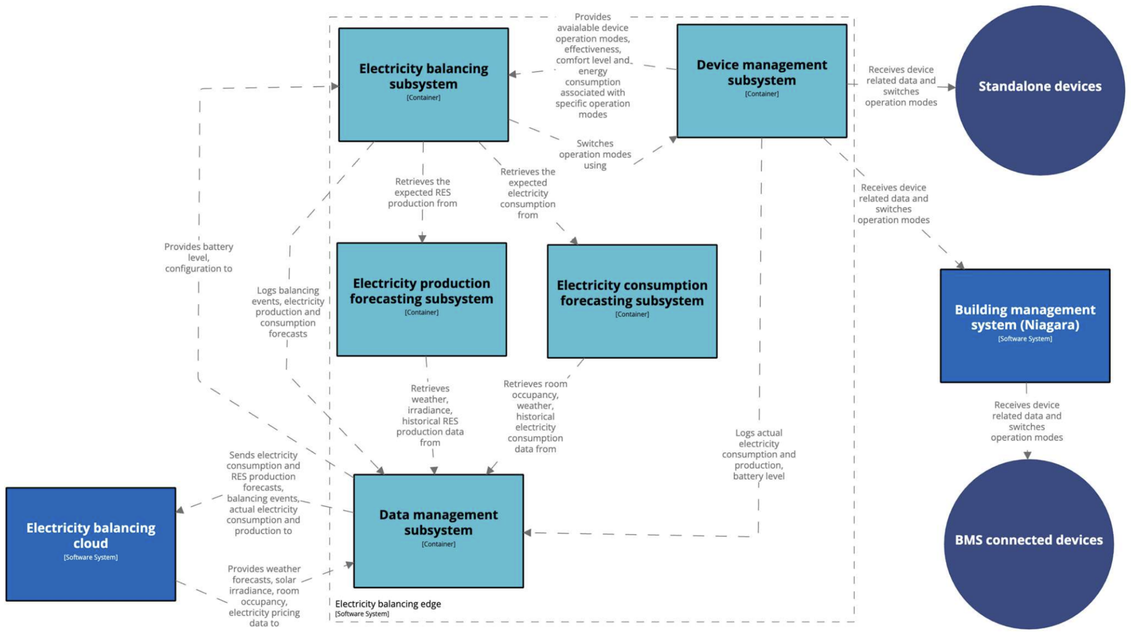
Electricity balancing subsystem
[Container]
Provides avaialable device operation modes, effectiveness, comfort level and energy consumption associated with specific operation modes
Device management subsystem
[Container]
Receives device related data and switches operation modes
Standalone devices
Switches operation modes using
Retrieves the expected RES production from
Retrieves the expected electricity consumption from
Receives device related data and switches operation modes
Provides battery level, configuration to
Logs balancing events, electricity production and consumption forecasts
Electricity production forecasting subsystem
[Container]
Electricity consumption forecasting subsystem
[Container]
Building management system (Niagara)
[Software System]
Retrieves weather, irradiance, historical RES production data from
Retrieves room occupancy, weather, historical electricity consumption data from
Receives device related data and switches operation modes
Logs actual electricity consumption and production, battery level
Sends electricity consumption and RES production forecasts, balancing events, actual electricity consumption and production to
Electricity balancing cloud
[Software System]
Data management subsystem
[Container]
BMS connected devices
Provides weather forecasts, solar irradiance, room occupancy, electricity pricing data to
Electricity balancing edge
[Software System]

Figure 4: Container diagram for the edge system for Project 3

entering the implementation phase. The project's organizational structured influenced the way the C4 method was used and models were prepared. The degree of centralization and the way reviews were organized had significant impact of appearance of the diagrams. The variations also suggest that software architects need additional guidance on using the C4 method.

The paper reports only analysis of the models created. However, it would be beneficial to survey projects' stakeholders about their views on efficiency of the development process. That would yield a more comprehensive answer to the third question. The software architecture was implemented as planned in Project 1, while retrospectives for Project 2 and Project 3 will be performed in due time.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] P. Ovaska, M. Rossi, P. Marttiin, Architecture as a coordination tool in multi-site software development, Software Process Improvement and Practice 8 (2003) 233–247. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-14344262656&doi=10.1002%2fspip.186&partnerID=40&md5=85209a2cbef41c1ae9a5fdec6f1d707f. doi:10.1002/spip.186, cited By 65.

[2] J. Vater, L. Harscheidt, A. Knoll, A reference architecture based on edge and cloud computing for smart manufacturing, volume 2019-July, Institute of Electrical and Electronics Engineers Inc., 2019. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85073154392&doi=10.1109%2fICCCN.2019.8846934&partnerID=40&md5=25dd203a2bd907a47f2a5480db857fa1. doi:10.1109/ICCCN.2019.8846934, cited By 19.

[3] K. Gallagher, A. Hatch, M. Munro, Software architecture visualization: An evaluation framework and its application, IEEE Transactions on Software Engineering 34 (2008) 260–270. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-42549103058&doi=10.1109%2fTSE.

2007.70757&partnerID=40&md5=4f9aaffa06fc13ae162e01ea15e5aad5. doi:10.1109/TSE.2007.70757, cited By 27.

[4] J. Nicacio, F. Petrillo, An approach to build consistent software architecture diagrams using devops system descriptors, Association for Computing Machinery, Inc, 2022, pp. 312–321. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85142937707&doi=10.1145%2f3550356.3561567&partnerID=40&md5=41972b7c0507d2505d95c48fbbe45927. doi:10.1145/3550356.3561567, cited By 0.

[5] I. David, K. Aslam, I. Malavolta, P. Lago, Collaborative model-driven software engineering — a systematic survey of practices and needs in industry, Journal of Systems and Software 199 (2023). URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85147541731&doi=10.1016%2fj.jss.2023.111626&partnerID=40&md5=4e0f3f4937e5442860b3d976c8e02f9b. doi:10.1016/j.jss.2023.111626, cited By 8.

[6] S. Brown, The c4 model for software architecture, http://c4model.com/, 2018. Accessed: 2024-07-26.

[7] A. Vazquez-Ingelmo, A. Garcia-Holgado, F. Garcia-Penalvo, C4 model in a software engineering subject to ease the comprehension of uml and the software, volume 2020-April, IEEE Computer Society, 2020, pp. 919–924. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85087897329&doi=10.1109%2fEDUCON45650.2020.9125335&partnerID=40&md5=85b2c782c089c273796350235a4c028d. doi:10.1109/EDUCON45650.2020.9125335, cited By 22.

[8] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2009) 131–164. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-61849169018&doi=10.1007%2fs10664-008-9102-8&partnerID=40&md5=6483f0b4e116b2744cb34cc4e26d7a13. doi:10.1007/s10664-008-9102-8, cited By 2668.

[9] I. Sittón-Candanedo, R. Alonso, S. Rodríguez-González, J. García Coria, F. De La Prieta, Edge computing architectures in industry 4.0: A general survey and comparison, Advances in Intelligent Systems and Computing 950 (2020) 121–131. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85065921255&doi=10.1007%2f978-3-030-20055-8_12&partnerID=40&md5=90b2d6ea943b97751b56a6d752a36cc3. doi:10.1007/978-3-030-20055-812, cited By 43.

[10] I. Sittón-Candanedo, R. Alonso, J. Corchado, S. Rodríguez-González, R. Casado-Vara, A review of edge computing reference architectures and a new global edge proposal, Future Generation Computer Systems 99 (2019) 278–294. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85065095398&doi=10.1016%2fj.future.2019.04.016&partnerID=40&md5=db0d61ba2364be72de352b0a800c03a9. doi:10.1016/j.future.2019.04.016, cited By 168.

[11] V. Prokhorenko, M. Ali Babar, Architectural resilience in cloud, fog and edge systems: A survey, IEEE Access 8 (2020) 28078–28095. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85079607476&doi=10.1109%2fACCESS.2020.2971007&partnerID=40&md5=928c09e47b3eea17729962f8681e0c5d. doi:10.1109/ACCESS.2020.2971007, cited By 43.