Constituent Systems Quality Requirements Engineering in Co-opetitive Systems of Systems

Pontus Svenson, Thomas Olsson, and Jakob Axelsson
RISE Research Institutes of Sweden
Stockholm, Sweden
pontus.svenson,thomas.olsson,jakob.axelsson@ri.se

***Abstract*—Systems of systems consist of independently owned, operated, and developed constituent systems that work together for mutual benefit. Co-opetitive systems of systems consist of constituent systems that in addition also compete. In this paper, we focus on quality requirement engineering for a constituent systems developer in such SoS. We discuss the needs and requirements of a structured quality requirements engineering process, with examples taken from the transportation domain, and find that there is a need for mediators and agreements between constituent systems developers to enable quality data exchange.**

Keywords—— Quality requirements engineering, collaborative systems of systems.

I. Introduction

Systems of systems (SoS) are becoming more and more important as digitalization increases. An SoS is defined [1] as a set of independently operated and independently managed *constituent systems* (CS) that collaborate to attain some common goal. Each CS is characterized by its *capabilities*, i.e., what it can do. During its lifetime, the SoS is presented with various tasks that it must address. For these tasks, the SoS puts together *constellations* [2] that consist of a set of CS whose combined capabilities enable it to solve each task. The constellations are thus the working horses of the SoS.

Each CS is assumed to have an owning, an operating, and a developing organization. Because of the managerial and operational independence of CS, there are in general several such organizations represented in the SoS. In general, the developing organizations are independent from each other; this is one component of the managerial independence in SOS. The developing organisations need to prioritize how their engineering resources are used to improve the quality in use of their CS. It is crucial for them to let data drive development. The operation needs to be monitored, since the operating conditions are difficult to fully comprehend at development time, and may change. Hence, to get access to relevant data, there needs to be agreements within the SoS on *which* information to share and *how* it should be shared.

In the work presented in this paper, we are investigating concepts around data sharing in a SoS to support product portfolio management and quality management aspects in SoS where the CS are simultaneously both collaborating and competing. We refer to this as a co-opetitive SoS.

One example of such a co-opetitive SoS is vehicle platooning. Here, different trucks (CS) cooperate by driving closely together, which reduces dynamic drag and leads to lower fuel consumption. A key issue in enabling such platooning is the possibility for trucks produced by different companies and belonging to different haulers to cooperate [3]. CS operators (haulers) and CS developers (Original Equipment Manufacturers, OEM) are simultaneously collaborators and competitors in a platooning SoS:

- The truck manufacturers need to collaborate to ensure that their products can platoon together but are also competing in selling to the haulers.
- The haulers need to collaborate to save on fuel but are of course also competing for the same cargo.

This means that the operating, developing, and owning organizations cannot just share all available data between themselves: they must take account of business needs, and weigh the benefits and drawbacks before sharing info.

Co-opetitive SoS in general is a main focus of our research, and in this paper we focus on the specific issue of product portfolio and quality management for a CS developer and the need to obtain data from other organizations within the SoS. Specifically, we address this issue by adapting the QREME model [4] developed for quality requirements management to the problem of how a CS developer can improve their understanding of the quality needs of their constituent systems when they are part of a co-opetitive SoS

The paper is outlined as follows. We start by characterizing the kinds of SoS we study in Section II, which is followed by an introduction to quality and product portfolio management in Section III. The application of the QREME method to platooning is then discussed in Section IV, followed by a conceptual discussion of quality information handling in Section V, after which the paper is concluded.

II. Co-Opetitive Systems of Systems

In this section we introduce some SoS concepts, apply them to the platooning example, and briefly outline the quality data collection problem.

A. SoS concepts

An SoS consists of independently owned, operated, and developed CS. Figure 1 shows a conceptual overview of the structure of an SoS, including the important roles of CS owner, developer, and operator. (In some SoS, two or even all three of these roles could be fulfilled by the same company. However, the general case, as well as the platooning SoS studied in this paper, may include all three roles.) A CS owner procures a CS from a CS developer and then instructs the CS operator. The SoS includes several CS, which in all but the simplest cases will have different owners, operators, and developers.

In the introduction, we mentioned constellations as the sets of CS that perform actual work solving a specific task that is presented to the SoS. Enabling the constellations to collaboratively solve these *constellation-level tasks* is the raison d'être for building the SoS in the first place. However, the independence of the CS enables them to simultaneously

This research was funded by Vinnova (Sweden's innovation agency), grant no. 2019-05100.

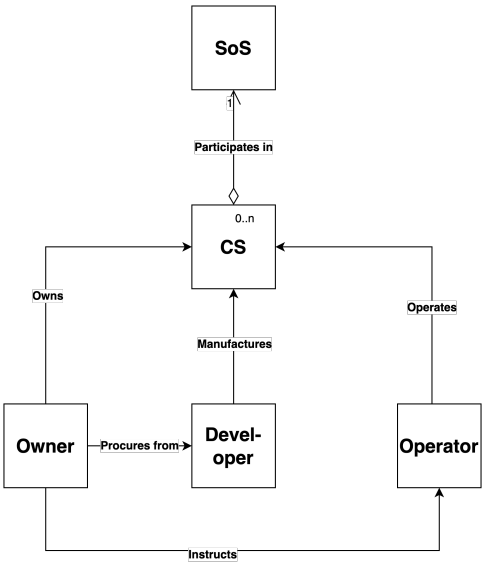

Figure 1. A conceptual view of a SoS, describing the owner, operator, and developer roles for a CS

work on solving *CS-level tasks* that do not require collaboration with others. We show a conceptual model of this in Figure 2.

Our running example of truck platooning illustrates this well: each participating truck delivers its goods (solves its CS-level task) by itself, while simultaneously possibly participating in a platoon (constellation) that solves a constellation-level task – reducing the fuel consumption of the involved trucks. It is this simultaneous work on tasks at different levels that enables the possibility of simultaneous competition and cooperation in the SoS.

Returning to the different CS-related roles, we note that for our purposes it is particularly important to distinguish between the developer and operator of the CS. It is the quality requirements challenges induced by the competition between different CS developers and CS operators that are the focus of this paper:

- Different CS developers compete against each other to sell CS to CS owners – the CS developers can thus be reluctant to share some data with each other, lest business intelligence is given away. However, the CS developers still need to cooperate to ensure that their products can work together within the SoS.

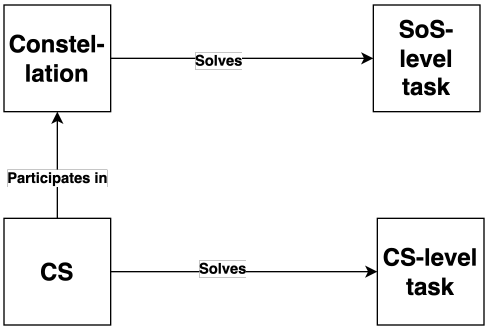

Figure 2. CS participate in constellations that collaborate to solve a constellation-level task, while also solving their individual CS-level tasks

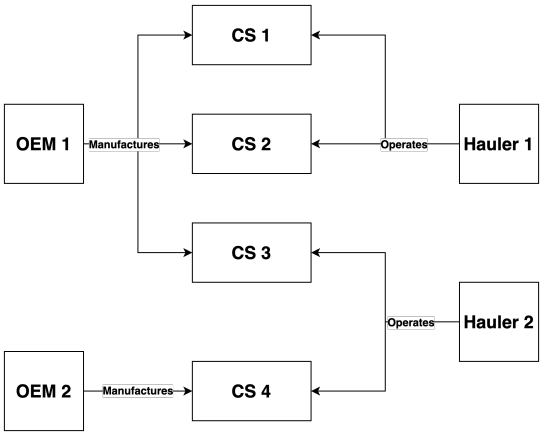

Figure 3. Example composition of a platooning SoS: there are two CS developers (OEM's) and two CP operators (haulers). OEM1 and OEM2 are competitors in selling CS to Hauler1 and Hauler2

- Different CS collaborate to jointly solve a constellation-level task, while each CS also pursues its own CS-level task. The CS compete against each other to get orders from customers – CS level tasks.

A common classification of SoS [1] is in terms of collaborative; acknowledged; directed; virtual. Here we use a characterization of SoS that is not directly related to any of these. From the perspective of this paper, we are interested in co-opetitive SoS where the CS operators and CS developers are both collaborators and competitors. One possible characterization of such SoS is that the CS collaborate because collaboration gives some business advantages but can nevertheless be competitors for the same customers.

B. Platooning SoS

We show an example platooning SoS in Figure 3. A platooning SoS is an example where CS operators and CS developers are both competing and collaborating. Two CS developers OEM1 and OEM2 compete against each to sell CS (trucks). The CS operators (haulers) want their CS to cooperate to form platoons. At the same time, they are also competitors for the same goods transports. Using the terminology introduced in Figure 2, we can say that the CS compete to get as many CS-level tasks as possible for their trucks, while they also collaborate in order to form platoons that solve the constellation-level task of reducing fuel consumption.

Figure 4 shows the same SoS at an instant in time in which CS2, CS3 and CS4 are collaborating in a platoon. While Hauler 1 benefits from collaboration with Hauler2, there is also competition, shown by the fact that CS1 is driving empty. From Hauler1's perspective, it would be better if GoodsB or GoodsC were transported by CS1. However, the benefits of trying to get this business (e.g., by aggressive pricing) need to be compared to the drawbacks if Hauler2 decides to withdraw from the SoS.

We now turn to the issue of data collection for quality purposes.

C. Quality Data Collection

It is useful to collect data about quality aspects to understand the problems and context of the CS [9]. Such data is also needed to understand the formation of the constellations that solve the SoS-level tasks. There is thus a

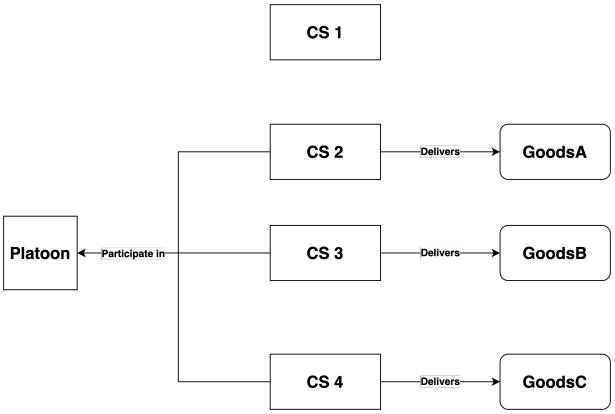

Figure 4. A snapshot of the state of the SoS at an instant in time. CS2, CS3, and CS4 are platooning, which saves fuel and thus increases profit for Hauler1 and Hauler2. This means that OEM1 and OEM2 have a need to cooperate so that the CS can participate in the same SoS. CS1 is currently empty, giving Hauler1 an incentive to compete with Hauler2 for goods.

need for both CS developers and CS operators to share some data – by doing this, both the CS and the SoS will perform better.

However, there are also incentives for the CS developers and CS operators to *not* share data. In platooning and similar SoS, it is the CS-level tasks that bring income. The SoS-level task for platooning consists of driving closely together to reduce fuel – this *saves* money for the hauler companies but does not give them any income. A hauler company whose trucks perform very well in the platooning task but very poorly in the goods transport task will soon go into bankruptcy!

Similarly, while it is beneficial for the truck developers to make their trucks interoperable and able to work well together on the SoS-level task, they must also always try to find better ways of solving the CS-level task, so that the hauler companies buy *their* trucks instead of the competitor's.

III. Product Portfolio Management and Quality

We now describe some quality requirements engineering background and introduce the QREME conceptual model

A. Background

Product portfolio management deals with the long-term development of a set of products. In the platooning case, we are considering the long-term development strategy of the truck manufacturer specifically for quality aspects.

Handling of quality needs is too often an afterthought rather than an informed decision [5]. The quality needs are translated into quality requirements (also known as non-functional requirements) which too often are insufficiently handled. Quality experience depends on the context [5] [6] – both operational environment as well as stakeholders. Stakeholders can be both direct (i.e. users), and indirect (i.a. owners). In a system in general and in a SoS specifically, it is challenging to attain a sufficient understanding of the context at design time [7]. Hence, a systematic quality engineering requires a continuous monitoring and measuring of the operation of all constituent systems in operation.

QREME [4] - Quality Requirements Engineering – is a conceptual model for the engineering activities that an organization must undertake to work on product quality. A

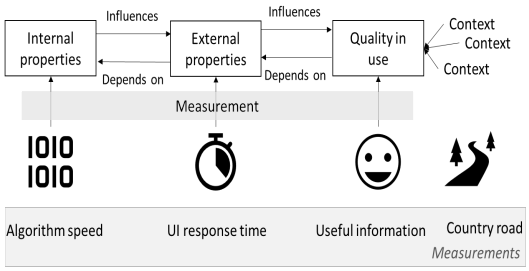

Figure 5. Quality perspectives, adapted from [6].

core concept is combining upfront hypothesis-driven engineering activities with data-driven measurements from operation to understand the quality needs. In a SoS, not only might the developer and the operator of a system be different organizations, but there are many organizations developing constituent systems and there are many organizations operating various constellations of constituent systems. Hence, monitoring of the operation to understand the quality needs requires sharing of data across a consortium of organizations – perhaps even competitors.

B. Quality in Use

The different stakeholders perceive a system as a combination of software, hardware and operating environment. Quality in use refers to how the stakeholders perceive the quality of a system, see Figure 5. Examples from ISO25010 are effectiveness and trust [6].

It is key to be able to connect the quality in use to engineering actions that can be taken for a system developer to understand how to address specific quality in use concerns. The ability to determine the quality in use at design time depends heavily on the operating environment and on the stakeholders. However, modelling the operating environment and stakeholders can be challenging. Relying on design time activities such as focus groups and expert opinions might not necessarily capture the actual relevant quality in use. There is hence a need to find ways to measure product quality in operation and relate those to quality in use – which typically cannot be measured directly on the usage.

In the kinds of SoS under study in this paper, it is a constellation of several CS that jointly address a specific problem instance. Hence, to be able to get relevant measurements from usage requires agreements with many parties on the exchange of usage data.

Quality in use depends not only on the operating environment but also the other CS in the SoS. However, despite some independence, there are agreements among the CS developers and CS operators, making it possible to exchange more information than other systems and stakeholders that are not part of the SoS.

CS developers may not have direct contact with end users [4]. They still need to get quality data from the users to understand the operating environment of their CS and to make as good a product as possible for how to use their engineering resources on their CS.

C. A conceptual view on quality requirements engineering

The QREME conceptual model divides the quality-related artefacts into abstraction levels (strategic and tactical) and context (engineering and operation), as illustrated in Figure 6. Activities, such as creating a quality model or setting up a data

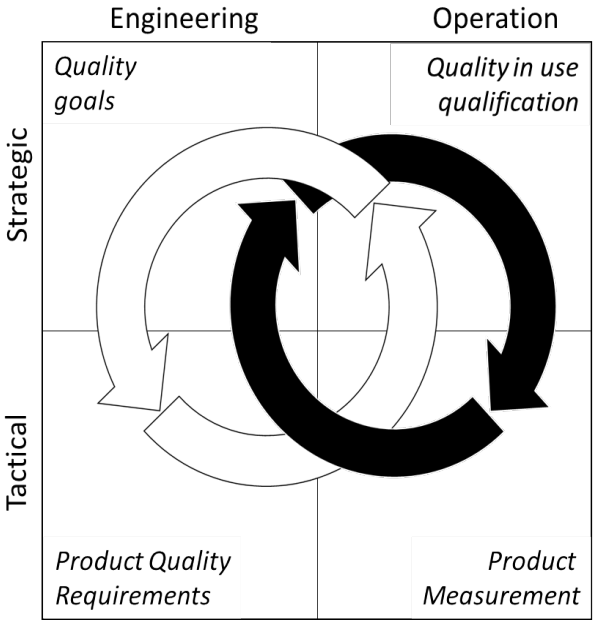

Figure 6. A quality requirements engineering conceptual model.

collection framework, are defined for how to create the artefacts. Lastly, input-output along the forward- and feedback-loop are defined among the artefacts, e.g., product quality requirements on performance and product performance measurements.

The conceptual model should be tailored to the organization and context characteristics. There are both external factors – such as laws and regulations – and internal factors – such as ability to update the software or market characteristics. Data collection is a challenging area when the developing organization is not responsible for the operation of the system. There might be agreements on overall quality requirements on a SoS Engineering level. The SoS quality level in operation should be monitored. If it is not possible to articulate requirements, relevant metrics for SoS quality in use can be agreed upon. Measurements are collected by different CS and needs to be agreed how to share with the relevant CS stakeholders.

IV. Case study: A platooning SoS

A major difference between a SoS and non-SoS contexts is that the different CS developers and operators have direct or indirect relationships in a SoS. Hence, not only do the CS contribute to the SoS but the SoS contributes to the CS as well. To understand the needs, it is vital to analyse the current usage. This has a great potential in a SoS.

A. Reliable estimates for a platooning SoS

A major acceptance factor for a platooning SoS is reliable estimates – *a quality goal*. For example, the ability to estimate the actual fuel savings induced by joining the platoon, or the time needed to catch up (*product measurements*) with a platoon. Fuel savings are related to how closely the vehicles can drive in a platoon (*product quality requirement*). This, in turn, is related to braking ability and not only by a single vehicle but the other vehicles in the platoon as well. The drivers trust in the estimates is an example of *quality in use*.

There are many factors affecting the braking capability such as the brakes, the weather conditions, tire conditions, or the weight of the load. Some of these factors can be tested during development time, but others vary with operating conditions. As safety can not be compromised, if relying purely on development time input, margins will be larger than necessary.

If, however, information is shared in the SoS, estimates can be improved in a way not possible if the CS are not sharing data. By collecting and sharing data on, e.g., road condition and weight, the situation awareness of the different CS can be improved and the estimates for safe distance to the vehicle in-front made more accurate, which in turn contributes to improved estimates of fuel savings.

Hence, if usage data is not shared within the SoS and this data is not used in the product management process to make decisions, the decisions are likely to be less accurate and risk wasting engineering resources on improving aspects not really adding value to the individual CS nor the SoS.

B. Platoon formation

The CS (currently the drivers, but in the future autonomous trucks) must actively decide to join a platoon for platoons to form. Many factors influence this, e.g., incentive models, user experience, traffic conditions, and estimated time loss. A user experience *quality goal* might be to minimize the decisions needed by a driver and try to automatize finding platoons and presenting them in a succinct way (*product quality requirements*).

It is difficult to directly measure quality in use, e.g., driver satisfaction with timing of platoon example presentation. Drivers can be interviewed to rationalize their decisions. Since it is not possible to ask all drivers to rationalize all their decisions, we have to use *product measurements* to understand the behaviour. One example such measurement is the ratio of presented platoon alternatives and actual platoons joined. However, interpreting that metric requires some afterthought as many factors contribute. Another *product measurement* might be to collect data on platoons formed for different geographic areas.

The forming of platoons depends on many factors. Analysing data from just one CS might not lead to actionable insight, but if data from the SoS can be shared, there is a larger likelihood that the data is actionable. With actionable data, the CS developers can experiment and test different ways of automating aspects of finding and presenting platooning information. The different CS developers, therefore, need to share some information even though they are competitors but can still make informed decisions on how to improve their individual CS in competition with others.

V. Quality Data Flows in a SoS

The previous section discussed some of the data which is needed for proper quality requirements management in platooning, i.e., *what* data would be needed. We now turn to the issue of *how* to get that data.

Sometimes there is a need for agreements within the SoS, supported by a *mediator* function. The mediator role is not fully independent of the SoS. They are used to help the CS collaborate. In a platooning SoS, there are needs for several kinds of mediators [3]:

- Platoon forming mediators, that help inform trucks/CS about possible platoons.

- Payment mediators, that help ensure that the benefits of platooning are distributed according to the agreements of the CS operators.

The operating organizations can measure the performance of their CS. The developing organization can sometimes directly get data from the operation, but sometimes need to go through other organizations. Mediators can measure both the performance of their mediating service and have a mediating role in the constellations that collaborate to support the data sharing agreement within the SoS. We note that there is a need to perform measurements on how well the constellations within the SoS perform the SoS-level tasks they are assigned. Are there opportunities for improvement in how the constellations are put together? Or in how they work together to solve the SoS-level task?

Each of these things can be improved in several different ways, hence it is also important to understand who needs to get the information. Is it any of the mediators that need to improve? Is it the CS developer? Or is it the agreements and business models implemented for the SoS as a whole that should be changed?

In Figure 7 we show a conceptual view of the quality data flows in a SoS. Quality data can be collected both on the constellation level and on the CS level. For the constellation level, the quality data is a measure of how well the constellation satisfies the users/beneficiaries of the SoS. For the platooning example, this could be the savings induced by the fuel reduction enables by the platooning. In addition to direct benefits to the users of the SoS, there are also societal benefits (e.g., less pollution and congestion, contributions to CO2 emission reduction). These, too, are collected on the constellation level. It is the responsibility of the SoS to collect this constellation level data. For this, a mediator service could be used. This mediator needs to collate all data and send appropriate summaries of it to CS operators and developers as well as other stakeholder roles. As explained above, there is also a need to share data between different CS operators and developers, both of which can be collecting quality data from the CS. This too can be facilitated by the quality data mediator, as shown in Figure 7.

VI. Conclusions

In this paper, we briefly discussed the quality requirements engineering for constituent systems participating in SoS with some degree of competition between the CS. We described the problem and some background an presented a conceptual model for information sharing. The conceptual model described the need for agreements between CS operators and developers on data exchange, and how a mediator service that can help CS developers receive quality data on the constellation level is needed.

As mentioned above, the long-term goal of this work is to work towards concepts for how SoS actors can form agreement about information and data sharing related to requirements engineering, thus enabling companies to make informed decisions about what and how to share. In order for the companies to find the correct balance between the initial competitive advantage of keeping data proprietary, and the long-term advantages of improved SoS performance, it is vital that they can assess the benefits and drawbacks of different approaches to information sharing.

References

[1] M. W. Maier, “Architecting Principles for Systems-of-Systems,” INCOSE Int. Symp., vol. 6, no. 1, pp. 565–573, Jul. 1996.

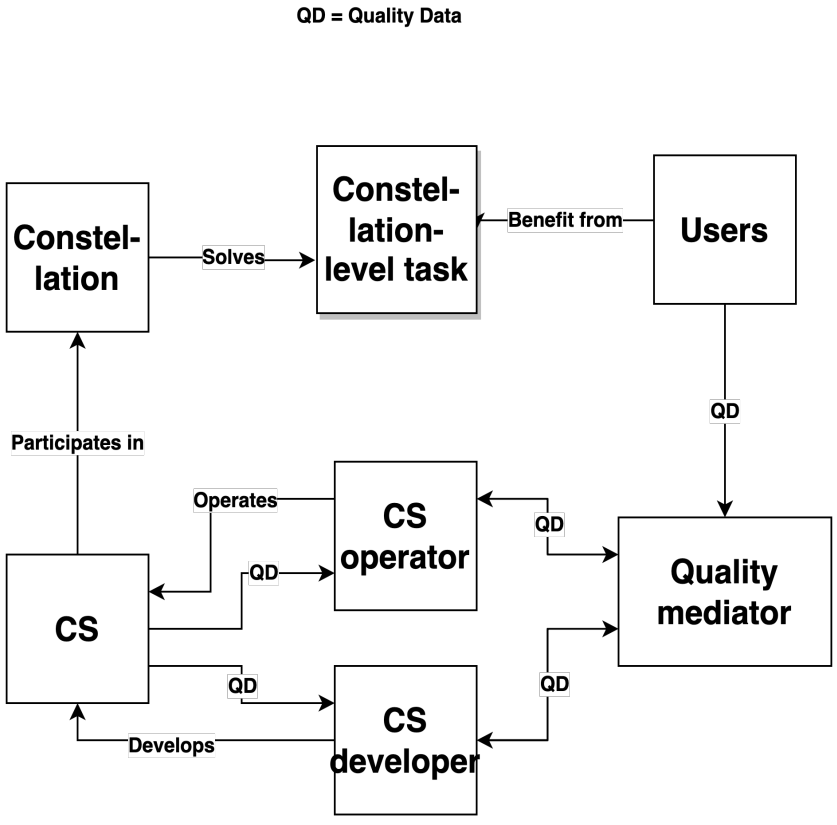

Figure 7. A conceptual diagram of quality requirements and quality data flows in a SoS.

[2] J. Axelsson, “A Refined Terminology on System-of-Systems Substructure and Constituent System States,” in *IEEE Systems of Systems Conference*, pp. 31–36, 2019.

[3] J. Axelsson, “Business Models and Roles for Mediating Services in a Truck Platooning System-of-Systems,” in IEEE Systems of Systems Conference, 2019, pp. 113–118.

[4] T. Olsson and K. Wnuk, "QREME–Quality requirements management model for supporting decision-making." In International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 173-188. Springer, Cham, 2018.

[5] T. Olsson, A. Sentilles, and E. Papatheocharous, “A systematic literature review of empirical research on quality requirements” *Requirements Engineering*, 1–23, 2022. Retrieved from https://doi.org/10.1007/s00766-022-00373-9

[6] ISO/IEC 25010:2011(E): Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SquaRE).

[7] J. Larsson, M. Borg, and T. Olsson. "Testing Quality Requirements of a System-of-Systems in the Public Sector-Challenges and Potential Remedies," *3rd International Workshop on Requirements Engineering and Testing (RET 2016), March 14, 2016, Gothenburg, Sweden*. Vol. 1564. 2016.

[8] D. Firesmith. “Profiling systems using the defining characteristics of systems of systems (SoS*)*”, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2010.

[9] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering", IEEE software 33, no. 1 (2015): 48-54.