1st TAB: Towards Robust Self-Adaptation in Decentralized Large-Scaled Systems of Systems

Daniel Matusek
daniel.matusek@tu-dresden.de
Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany

ABSTRACT

Today's distributed applications require steady maintenance. To tackle this problem, so-called self-adaptive systems (SAS) can be used to change the behaviour automatically to adapt to a changing environment and context. Open challenges remain when those SAS get combined with Systems of Systems (SoS). SoS can get partitioned in multiple sub-parts as a result of errors or connection faults which rises the need for a decentralized self-adaptation approach in SoS. In this doctoral paper, those open challenges are discussed and explained using a scenario of self-driving vehicles. Ideas for solving the problems are presented and the evaluation method of using the Webots simulation environment is explained. Solving the problems of self-adaptive SoS will enable robust adaptations in large-scale systems.

1 MOTIVATION

Today's distributed applications require steady maintenance. To tackle this problem, so-called self-adaptive systems (SAS) could be used to change the behaviour automatically [3]. Solving the problem of adapting systems to changing context and its environment would allow for nearly perpetual running systems [17]. SAS can monitor themselves and their environment and analyse the state to adapt the internal structure and behaviour to the changed context. Many proposed systems use a central instance to control adaptation across multiple devices.

Systems of systems (SoS) are large interconnected collaborative systems [8], which consist of multiple autonomous systems. They work together to achieve a common goal, but the individual subsystems can have their own goals, too. Due to numerous participants from different manufacturers heterogeneity is introduced into those systems. During the lifetime of an SoS, communication errors or internal errors can occur, which impacts the coordination of self-adaptation and disturbs a trouble-free procedure. Due to connection faults, new subsystems can emerge and a system gets partitioned in even more parts. Due to multiple autonomous parts and heterogeneity of the system, a centralized approach for self-adaptation in SoS is not suitable due to possible bottlenecks, communication overhead and a single-point-of-failure.

A decentralized solution for coordinating adaptations increases robustness and allows for improving the scalability of those systems [19]. Recently researchers proposed first approaches for the decentralized coordination of adaptations [16]. By developing a communication protocol to invoke adaptations decentrally, the need for a central instance was superseded. This approach has not been yet adopted for SoS, where the overall system structure is more complex than in a plain distributed system. Current approaches [16] rely on disturbance-free communication and correct working nodes, which might not be suitable in an SoS when many participants are involved. Enabling decentralization for self-adaptation in SoS would help to make those systems more robust and solving the remaining research challenges.

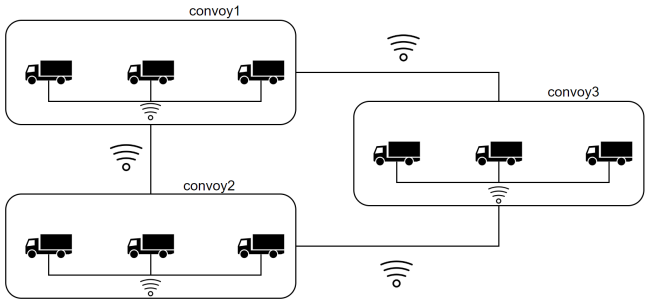

Figure 1: Introduction of the scenario

1.1 Problem scenario

To describe the problems which arise when combining self-adaptation and SoS, a scenario of a self-organising [4] SoS is presented (see Figure 1). The SoS consists of trucks, which span convoys or partitions of convoys on a network of highways. Convoys are organized towards a common goal and work together to optimize their results and fulfilment of the task. Each convoy can communicate with all the other convoys and the trucks can communicate with the other trucks in their convoy. In Figure 1 the communication between trucks and convoys is indicated. Those connections could be disturbed and prevent the message exchange between participants, or even lead to the isolation of single trucks because they cannot connect to other peers. The subsystems get divided into multiple subparts and partitions are emerging. Those subparts should be able to work for their own to keep the whole system running. Split trucks from their convoy should now organize themselves together and continue to realize their system goal and possibly connect to their originating convoy later. Figure 2 shows a connection error for *convoy1*, where the right truck cannot communicate with others at this moment. This results in missing messages about ongoing adaptations by other trucks even in the same convoy. A truck or convoy might not adapt to a new or changed goal.

In the following subscenarios which will be used for the evaluation and for the visualisation of the contribution are presented:

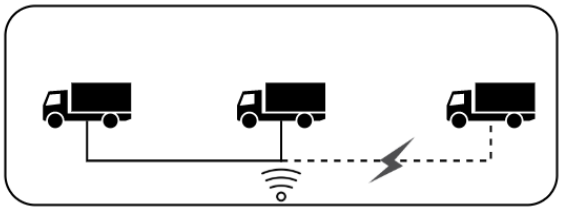

Figure 2: Connection error in a convoy

Overall scenario: Platooning/Convoys. Multiple trucks are driving in a convoy on a highway. Those trucks are creating platoons or convoys to increase their energy efficiency, to increase security and to allow other vehicles to have more space on the roads. Security for all road users rises, since overtaking and driving behaviour is more predictable in an automated setting. This overall scenario enables several subscenarios which can arise.

Subscenario: Construction Site. Construction sites along the highway are an example of collaborating heterogeneous devices or nodes. Construction sites are usually present along highways and so provide a good example. Those construction sites (see Figure 3) can be the destination of platoon members (red truck in the example), where the truck has to leave its convoy to reach the destination since delivery for a construction site could be possible. This truck then joins the construction site and has to collaborate with the construction nodes to fulfil its goal. In this subscenario, nodes have to change their region or subsystem, thus end their collaboration with the old convoy and start interacting with their new subsystem.

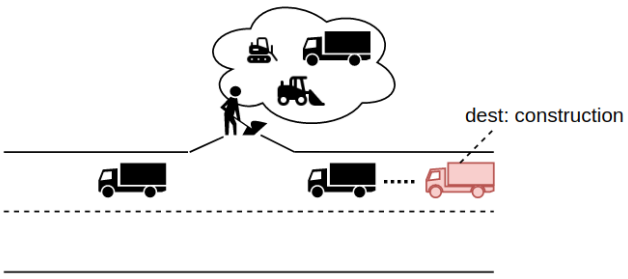

Figure 3: Truck approaching a construction site

Subscenario: Crossing. Next, in a scenario with autonomous driving trucks and cars, the participants will approach crossings, where a safe way to cross the roads must be found. Convoys and cars approaching a crossing must collaborate to guarantee safe crossing of the road. This is a safety-critical scenario since the cars are not allowed to cross the intersection autonomously as they want, but are forced to interact with all other cars which approach the intersection. A common solution must be found and executed. Nodes near the intersection form a temporary subsystem in which collaboration is needed. Guaranteed adaptation in a short time window is needed in such a subscenario.

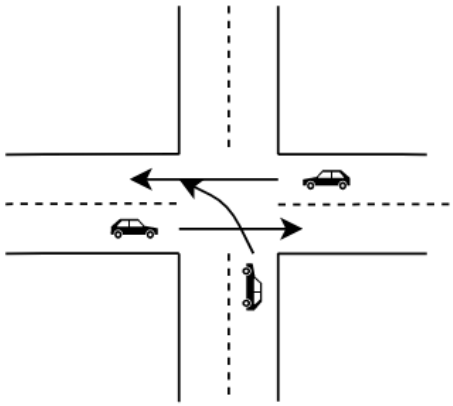

Figure 4: Crossing with collaborating cars

Subscenario: Disconnecting participants. With the following scenario, we want to introduce an error case which can occur randomly in the field. When trucks or cars enter a tunnel, the connection to the peers outside the tunnel can be lost. This prevents the peers outside to communicate with the cars inside the tunnel and vice versa. When adaptation operations are triggered either inside or outside the tunnel with the current protocol from Martin Weißbach [16] would not allow for that adaptation and would abort it. This scenario increases the need for a mechanism to allow for partial adaptation and short-time inconsistency of the overall global state.

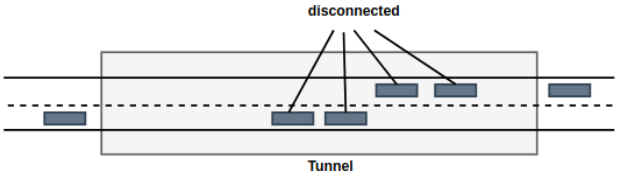

Figure 5: Disconnected Participants in tunnel

For now, following concrete adaptations in addition to the adaptations of Weißbach et al. [16] are intended. In a later stage, they will be mapped onto the adaptation structure from Weißbach et al. [16] and adaptation plans will be created.

`joinConvoy()`: truck joins another convoy
`leaveConvoy()`: truck leaves current convoy
`switchConvoy()`: switch convoy to nearby specified convoy
`mergeConvoys()`: two convoys collaborate
`electLeader()`: leader between cars is chosen

1.2 Scenario system model

For the scenario and throughout the thesis, assumptions must be made about the underlying system model to understand the identified challenges and preliminary research questions. First, a decision about the form of coordination and knowledge sharing, i.e. if every participant must know about each other and if every participant

must learn about all adaptations and changes in the system. Since the member of the SoS can change dynamically, the participants do not necessarily need to know about every single truck in the system.

To perform adaptations, members of the SoS must agree on or decline the adaptation, which leads to a decision whether we need a strong consensus to adopt as one or if subparts are allowed to adapt without the agreement of other subsystems or nodes. Since the scenario is considered a large-scale system, it would be a drawback if all nodes of the SoS must have agreed on an adaptation. Therefore adaptations without the agreement of all other participants are possible.

Next to the aforementioned constraints for the regarded types of SoS, additional properties must be met by the designed scenario. We use the characteristics described by Wätzold et al. [15]. The SoS is open, which allows for new participants during run-time and also allows for leaving members. The system is heterogeneous, which means that different types of members can be part of the system, e.g. different manufacturers. Self-adaptivity is a mandatory property. Next, it is a dynamic system because its internal structure can be varying during runtime. The subsystems in the SoS work collaborative to achieve a common goal but can work independently if they are on their own. Decentralization is a key aspect in this example to ensure higher availability when convoys move independently [2]. A central coordination instance is a single point of failure and if this central instance is not reachable due to shortages or connection issues, adaptations could not be invoked. Besides that, we eliminate a single point of trust and allow multiple instances to make decisions instead of trusting only one central node. Additionally, decentralized control allows for better scalability [19], which is crucial in a scenario of self-driving trucks, where the number of participants is unknown upfront.

2 FOUNDATIONS

During the thesis, the notion of roles will be used to develop a solution for the problem of robust, decentralized SAS. With the help of roles, we can express the context-dependent and collaborative behaviour of objects, which is crucial for self-adaptive SoS. This approach has been proposed by Charles W. Bachmann [1] in 1973. To describe roles, their objects and properties, Steimann introduced 15 features which apply for role-based infrastructure [12]. Kuehn et al. [5] surveyed several approaches for role-based programming languages and defined an extended understanding of roles, in addition to those from Steimann [12]. An object enriched with a role can act according to its played role and extends its original behaviour by the properties of the role. A big advantage of this concept is the ability to dynamically change the role of an object during runtime. This enables run-time adaptation which supersedes the need for long maintenance breaks and downtimes. An object (a so-called natural type), which could be e.g. a real-world entity, can play one or more roles and so change the behaviour, functions and abilities. Functionality could be loaded during run-time and would allow for a nearly perpetual runtime if we took aside arising errors or bugs. Those are some of the properties that can be exploited in the research of self-adaptive software systems. To develop role-based applications, a role-based domain-specific language called SCROLL was developed, which is based on scala and allows for dynamic dispatch [7].

Weißbach et al. [16] introduced an algorithm for coordinated decentralized adaptations. Every node has its adaptation manager (AM) which is responsible for invoking adaptations on the node runtime and is communicating with the AM on other nodes. The structure is indicated in Figure 6. The adaptation coordination is managed by the AM on a higher layer. The adaptation process is performed transactionally and atomically, which means that only if every participant agrees and successfully prepares the adaptation, it will be executed. Otherwise, the changes will be rolled back. Every AM is allowed to invoke an adaptation if necessary. The implementation and concept from Weißbach et al. [16] are using a local role application runtime called LyRT[1] [13, 14] on their nodes, which naturally allows for the run-time adaptations of the objects.

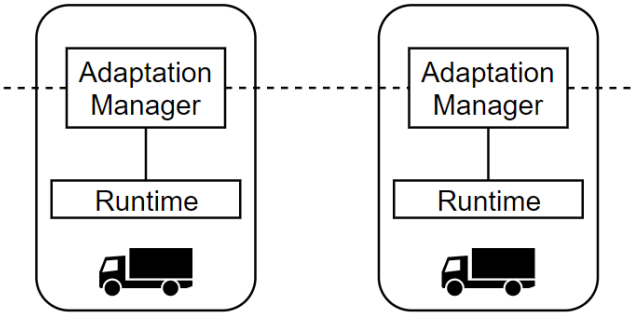

Figure 6: Adaptation Manager in decentralized protocol

The protocol from Weißbach et al. [16] performing adaptations in an atomic and transactional way. This fits the concept of ACID (atomicity, consistency, isolation and durability) properties for database systems. This ensures that an adaptation is only successful if the whole change is applied for every participant. A contrary concept is BASE (Basic Availability, Soft-State, Eventual consistency) database transactions, which ensures that some level of availability is guaranteed and the data might not be the most current. Eventual consistency ensures, that the system will reach some guaranteed state eventually, but not necessarily at every moment.

Conflict-Free Replicated Data Types (CRDTs) is a distributed data structure which allows for the automated synchronisation of distributed replicated data sets. CRDTs can be used offline and are not dependent on their latency. E.g., two parties share a common file and have an offline copy each. They are allowed to work offline on them and to make changes. When they both are online again, the replicas will get synchronised in the background automatically. CRDT enable the concept of eventual consistency of BASE because the replicated datasets are not consistent if the participants are not connected, but synchronise afterwards.

[1] https://github.com/nguonly/lyrt-with-transaction

3 RESEARCH GAP AND RESEARCH QUESTIONS

In the following section, identified problems of the existing work and related research questions will be presented which will be tackled in the next years. Limitations are presented and discussed.

Subsystems in SoS can be the result of errors and connection faults, which partition the system. Those emerged subsystems could now perform adaptations to provide useful services within each partition. Each partition then takes its own independent adaptation decisions and thus, the subparts drift away from a globally consistent configuration. As a consequence, after reintegrating the subparts synchronization might be necessary to unite all system parts back to a single system. This challenge was already identified by Weyns et al. [18]. This results from the independence of SoS because subsystems want to achieve their goals. Projecting this onto the scenario, *Convoy1* and *convoy2* from Figure 1 could both trigger adaptations which affect the whole SoS including *convoy3*. Those changes can be contradictory to each other, which affects the system negatively. Considering the protocol from Weißbach et al. [16], concurrent adaptations could be even invoked by every single truck. This leads to the first research question:

RQ1. How are concurrent adaptations in multiple subsystems handled?

Resulting inconsistency and a possibly false state of the system as a cause of connection errors is another challenge to tackle. Due to missing information because of a connection error, a truck or convoy might have wrong routing information and keep moving towards a wrong goal. Using the Weißbach protocol to perform a decentralized adaptation, the adaptation would be aborted since the disconnected truck could not respond to the request. For our scenario, aborting a change might not be the optimal solution regarding self-optimization. Inconsistency is tolerable to a certain point, but it must be dealt with.

RQ2. How can we recover a consistent state in an SoS after a self-adaptation was performed when a node or connection failed?

Another aspect to investigate is the distribution of leaders in the SoS [6] and the hierarchy of decentralized SoS. In the decentralized coordination protocol from Weißbach et al. [16] each Adaptation Manager can act as a coordinator in the system. The problem is illustrated in Figure 1. With the protocol in its current form, every truck can invoke changes for the whole system regardless of its convoy. It is questionable if this is suitable for SoS, where a system is divided into multiple subsystems because it raises the communication overhead and increases the chance of concurrent adaptations.

The decentralization pattern used in [16] is shown in Figure 10. Those patterns refer to the self-adaptation patterns from Weyns et al. [20]. We can see that every node (in the dashed box) has its MAPE feedback loop implemented and thus monitors its context and internal state, analyses it, plans adaptations and executes them. Thus, we don't have any central instance for either planning, monitoring or analyzing adaptations and the context. This generally allows for good scalability and communication overhead for local adaptations is low [20] and a single-point-of-failure is eliminated. On the other side, this approach has drawbacks for finding a consensus for many distributed nodes and scalability could be compromised in that case [20]. Besides, consistency could be lowered in a fully decentralized approach and sub-optimal adaptations decisions could be made. The problem of maintaining consistent adaptations in an acceptable time for larger system sizes with message loss has also been shown in experiments by [16].

The problem of leader election and the decentralization problem can be transferred to the crossing example in Figure 4. In this case, trucks or cars are approaching an intersection and must collaborate in order to find a safe way to cross it. Structural adaptation is necessary to enable collaborative behaviour and choosing a leader, in this case, would allow one node to take the decision for all members to prevent contradicting adaptations.

The decentralization patterns as shown in Figure 10 and approaches introduce another challenge which is also interesting to tackle. It would be interesting to investigate those patterns for decentralized self-adaptation and enable adaptive coupling where the control patterns can change during runtime. This would dynamically allow systems to mix loosely and tightly coupled parts to fulfil multiple system properties like scalability on demand [20].

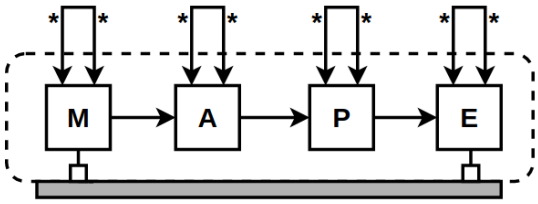

Figure 7: Decentralized coordination of Self-Adaptation

RQ3. Which degree of decentralization is suitable for the adaptation coordination in SoS?

Investigated approaches for decentralized adaptations use the concept of atomic adaptation transactions for performing the distributed adaptations [16]. It is questionable if this is the right procedure if we consider large systems with a high amount of nodes. A truck in *convoy1* could trigger an adaptation to change the behavior of the other trucks in *convoy2* or *convoy3*. An example is route optimization for the other convoys. With atomic adaptation transactions following the ACID principle, if one of the convoys or trucks did not accept the change or revoked it, the whole adaptation will be rolled back. Even if the other convoy would profit from it, the change would not be executed. This could happen in a scenario as shown in Figure 5. Structural changes are invoked by either the trucks inside the tunnel or outside the tunnel and the information is not shared with the participants inside the tunnel. Because either the ones or the other do not reply to the adaptation request, the adaptation is discarded.

RQ4. How do non-atomic adaptation transactions supporting a notion similar to eventual consistency behave in comparison to

atomic adaptation transactions for decentralized self-adaptation?

Approaches for solving and answering the challenges and research questions will be presented in the next part. The possible contributions will be discussed.

4 APPROACH

The presented approach from Weißbach et al [16] will be used as a foundation to develop decentralized self-adaptation in SoS and to introduce more robustness. In the following section goals for the elaborated problems will be presented.

Regarding **RQ1**, the problem of concurrent adaptations must be solved by a synchronization mechanism which compares the originating and resulting states of adapted nodes. The approach for **RQ2** is related to the first, since the result of both problems is missing synchronisation of the knowledge about the global state and an inconsistent or erroneous global state.

A procedure for failed nodes and connections between trucks and convoys must be developed which reestablishes consistency when subparts they get connected again or when concurrent adaptations were performed. Two possibilities can be investigated for this. The first can be used to address both **RQ1** and **RQ2**. In chapter two, CRDTs were introduced, which are mainly used for systems like e.g. Google Documents, which allow for collaborative work and working with offline copies [11]. It would be interesting to apply the concept of CRDTs to self-adaptive SoS. The current state of nodes and application could be treated like a shared dataset, which is replicated on multiple nodes then. In case of a disconnection and possible resulting concurrent adaptations, those changes will be translated into a form similar to changes for CRDTs and after reconnection or finished adaptations, the changes can be synchronized again. Since this is an early stage of research, a deeper investigation on how to use CRDTs for self-adaptive SoS is necessary. In the case of a disconnected node which does not notice an upcoming adaptation (**RQ2**), the protocol from Weißbach et al. [16] could also be extended to replay adaptations when nodes are available again and if the adaptations were not concurrent. If a truck in a convoy failed, the other participants of the convoy are responsible to resend all missed messages. For whole convoys, the nearest convoy should trigger the recovery. This solution approach requires the disconnected subsystem or single node to remain in its original state before it came to an error, to ensure that no adaptations or changes will be overwritten.

Another identified problem regarding decentralized self-adaptation in SoS in the scenario is the leader election. The decentralized protocol from [16] allows every participant of the system to invoke adaptations and all nodes are treated equally. We will investigate if every AM of a subsystem should be able to communicate with others SoS, or if each subsystem has a leader which is responsible for inter-subsystem communication. We will compare the impact of different amount of leaders in an SoS and different hierarchies, or if equal members in an SoS regardless of the different subsystems are the right approach. For a hierarchical approach, a correct leader election mechanism must be chosen. This could be either a random decision or a sophisticated leader election mechanism. An exemplary algorithm for decentralized distributed systems was presented by Mo et al. [10]. The proposed way is stable even on position changes of the devices which is important for the scenario of self-driving trucks or if the topology adapts, which is very important for SoS. Their approach also makes assumptions about the correct moment in time when the leader must be switched. Additionally, we will analyse the needed hierarchical structure for efficient adaptations in SoS. A possible leader distribution and hierarchy is shown in Figure 12. Each convoy has a leader now, which is responsible for the communication with the other convoys. For the internal structure of a convoy, each node is equal, but with this approach, the communication overhead for the whole system is reduced. The leader is indicated in orange. Leader change as proposed in [10] is important, since a leader can get disconnected in a subsystem, which is shown in *convoy1* in Figure 12. Another truck must lead the convoy from now on. In combination with leader election, it is necessary to detect if partitioning occurred and which nodes belong to which partition. The next step would then be a leader election in that partition.

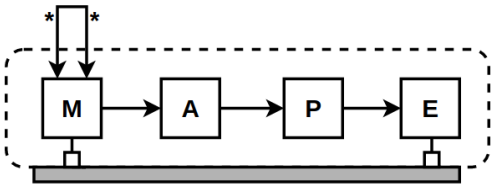

Figure 8: Information Sharing Pattern

A good starting point for investigating further decentralization pattern is the 'information sharing pattern', which is an extension or special case of the decentralized coordination pattern [20]. It provides better scalability than the used pattern by [16] and has less stringent interaction because only the monitoring components are exchanging information. This approach is scaling better wrt. the communication. The A-P-E components don't need any coordination and allow for more timely decisions and execution of adaptations [20]. The drawback of this approach is the worse accomplishment of global goals, but local optimal objectives are more likely reached.

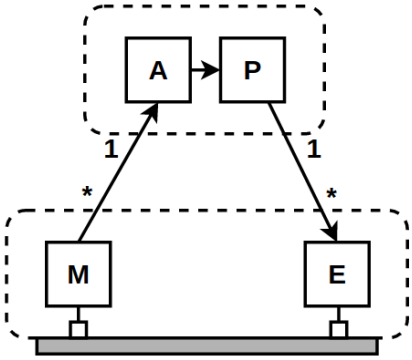

Figure 9: Decentralized coordination of Self-Adaptation

Another approach is the Master/Slave pattern which is presented by [20]. This pattern is suitable for scenarios in which slave components monitor themselves but let master nodes decide about the adaptation and the decision making. The adaptations are executed locally then. This has the advantage of efficient decisions for global objectives and guarantees but generates overhead since data must be collected at the master all adaptation actions must be distributed among all slaves. This could lead to a bottleneck in a large-scale distributed system. Additionally, the problem of a single point of failure exists.

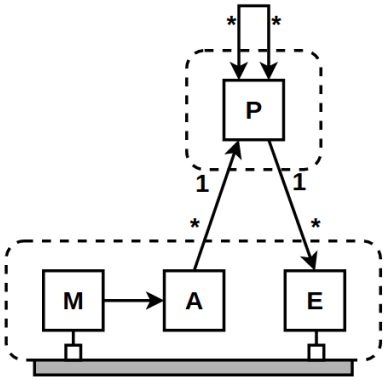

Figure 10: Regional Self-Adaptation Pattern

A related but more scalable approach is the pattern of regional self-adaptation. This pattern introduces a layered separation of concerns which means that several MAPE-loops are delegating their planning to higher-level components and that the managed resources are only performing M-A-E of the MAPE-loop. Each region, which could be a subsystem in our scenario, has its planner which can communicate with each other and thus enables a layer for separation of concerns. This has the advantage that the amount of data and frequency for interaction with the planner is reduced, but runtime coordination of the execute phase among different regions is not possible. Aggregation of local analysis among different planners and their coordination might incur overhead.

A more complex pattern is the pattern for hierarchical self-adaptation. In this approach, the adaptation logic is structured and the complexity of self-adaptation can be managed with that approach. The benefit of this approach is a separation of concerns, where bottom layers focus on concrete adaptation for their own and a higher level of the system have a broader perspective on all devices. That introduces also downsides because it is not easy to separate all concerns, especially when the goals of different subsystems are interfering. Weyns et al. say that there is no guarantee that an overall solution for an adaptation is optimal for all participants [20].

The existing decentralized adaptation protocol [16] uses atomic adaptation transactions for performing adaptations and follows the ACID principle. Since this approach introduces some limitations for self-adaptive SoS as explained in the chapter before, we will investigate the impact of the BASE concept for those transactions,

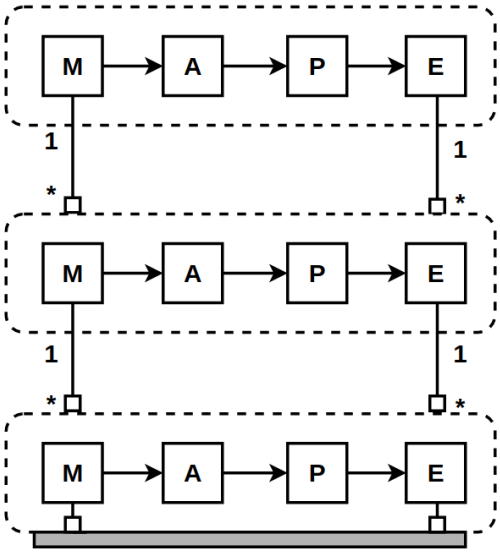

Figure 11: Hierarchical Self-Adaptation Pattern

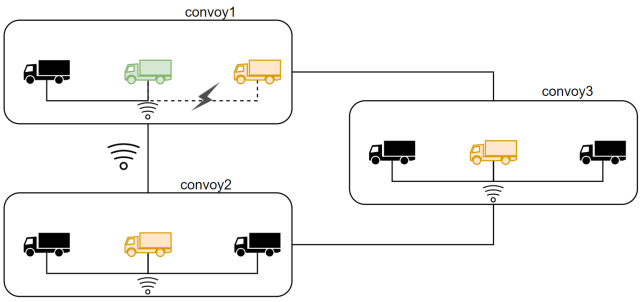

Figure 12: Leader election in SoS

which is motivated by modern database systems[2]. The current adaptation mechanism would allow a certain level of inconsistency and errors of nodes which participate in a transaction. It will be necessary to define a limit of tolerance for the eventual consistency and which states are still valid system states. A strong consensus is not necessary then. This would allow for error tolerance during the adaptation phase since the work focusses on large-scale systems where errors are more likely to happen than in a small system, and only the affected parts must be recovered.

5 EVALUATION

The work will be evaluated using the scenarios described in subsection 1.1. For doing this, the WeBots simulation environment will be used. We will implement and model the scenarios and then apply the role runtime including the developed algorithms. We can then make assumptions about the behaviour of the algorithms in a real-world scenario.

[2]https://neo4j.com/blog/acid-vs-base-consistency-models-explained/

Additionally, we consider testing our self-adaptation techniques in the DEECo^2 environment, which is accepted in the SEAMS community. The DEECo environment introduces temporal disconnections, network delays, vehicle-to-vehicle communication and a simulation framework. This allows us testing our autonomous components and dynamic components and the deployment of different adaptation techniques. Scenarios in the DEECo environment enable mobility of node with limited data availability and makes testing self-adaptive systems with physical distribution possible.

The benchmarks of interest are the adaptation duration, downtime of instances during role adaptation, the message loss rate for the scenarios and the duration of an inconsistent state after BASE adaptation transactions. Artificial errors generated in the evaluation are disconnected convoys, trucks and unavailable leader which must be elected then in a short time frame.

6 FUTURE WORK AND RESEARCH PLAN

A doctoral paper for the ECSA 2020 conference was published in September 2020 [9]. For the rest of this year, a collaborative paper with Ilja Shmelkin and Tim Kluge with the deadline on 31st December about our approaches and a common use case is planned. Besides that, we want to investigate the pattern for the decentralization including the leader election as a next step and publish the results in a conference paper. The next step will be researching ACID vs. BASE adaptation transactions for decentralized adaptations. For the evaluation of the scenarios with its subscenarios, the simulation environment 'WeBots' must be set-up in the near future.

- **Q4/2020**: Finished deeper literature work and final research questions
- **Q3/2021**: Implemented improved algorithms for decentralized adaptations
- **Q4/2021**: Evaluation of the results using Webots
- **Q1/2022**: Begin of write-down
- **Q3/2022**: Handing in the finished thesis

7 CONCLUSION

In this TAB paper, the motivation and challenges in the field of SoS with decentralized self-adaptation were presented. Current problems and challenges were discussed and the state-of-the-art approaches were introduced. It was shown that there are still open fields of interest and issues regarding self-adaptation in partitioned SoS. The approaches for tackling the challenges were outlined and milestones for the PhD project have been introduced.

ACKNOWLEDGMENTS

This work is funded by the German Research Foundation (DFG) within the Research Training Group Role-based Software Infrastructures for continuous-context-sensitive Systems (GRK 1907).

REFERENCES

[1] Charles W. Bachman and Manilal Daya. 1977. The Role Concept in Data Models. In *Proceedings of the Third International Conference on Very Large Data Bases - Volume 3* (Tokyo, Japan) *(VLDB '77)*. VLDB Endowment, 464–476.

[2] Roberto Casadei and Mirko Viroli. 2018. Collective abstractions and platforms for large-scale self-adaptive IoT. In *Proceedings - 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2018*. Institute of Electrical and Electronics Engineers Inc., 106–111. https://doi.org/10.1109/FAS-W.2018.00033

[3] Rogério de Lemos et al. 2013. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science*. Vol. 7475. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35813-5_1

[4] Alois Ferscha. 2015. Collective adaptive systems. In *UbiComp and ISWC 2015 - Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the Proceedings of the 2015 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, Inc, New York, New York, USA, 893–896. https://doi.org/10.1145/2800835.2809508

[5] Thomas Kühn et al. 2014. A Metamodel Family for Role-Based Modeling and Programming Languages. In *Software Language Engineering*, Benoît Combemale, David J. Pearce, Olivier Barais, and Jurgen J. Vinju (Eds.). Springer International Publishing, Cham, 141–160. https://doi.org/10.1007/978-3-319-11245-9_8

[6] Veronika Lesch, Christian Krupitzer, and Sven Tomforde. 2019. Emerging Self-Integration through Coordination of Autonomous Adaptive Systems. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. IEEE, 6–9. https://doi.org/10.1109/FAS-W.2019.00016

[7] Max Leuthäuser. 2015. SCROLL - A Scala-based library for Roles at Runtime. In *Proceedings of the 3rd Workshop on Domain-Specific Language Design and Implementation (DSLDI 2015)*.

[8] Mark W. Maier. 1998. Architecting principles for systems-of-systems. *Systems Engineering* 1, 4 (1998), 267–284. https://doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D

[9] Daniel Matusek. 2020. Decentralized Self-adaptation in Large-Scaled Systems of Systems. In *Software Architecture*, Henry Muccini, Paris Avgeriou, Barbora Buhnova, Javier Camara, Mauro Caporuscio, Mirco Franzago, Anne Koziolek, Patrizia Scandurra, Catia Trubiani, Danny Weyns, and Uwe Zdun (Eds.). Springer International Publishing, Cham, 27–37.

[10] Yuanqiu Mo, Jacob Beal, and Soura Dasgupta. 2018. An Aggregate Computing Approach to Self-Stabilizing Leader Election. *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)* (2018). https://doi.org/10.1109/FAS-W.2018.00034

[11] Nuno Preguiça, Carlos Baquero, and Marc Shapiro. 2018. Conflict-free Replicated Data Types (CRDTs). (may 2018). https://doi.org/10.1007/978-3-319-63962-8_185-1 arXiv:1805.06358

[12] Friedrich Steimann. 2000. On the Representation of Roles in Object-Oriented and Conceptual Modelling. *Data & Knowledge Engineering* 35 (2000). https://doi.org/10.1016/S0169-023X(00)00023-9

[13] Nguonly Taing et al. 2016. Consistent unanticipated adaptation for context-dependent applications. In *Proceedings of the 8th International Workshop on Context-Oriented Programming, COP 2016*. Association for Computing Machinery, Inc, New York, New York, USA, 33–38. https://doi.org/10.1145/2951965.2951966

[14] Nguonly Taing et al. 2016. Run-time variability of role-based software systems. In *MODULARITY Companion 2016 - Companion Proceedings of the 15th International Conference on Modularity*. Association for Computing Machinery, Inc, 137–142. https://doi.org/10.1145/2892664.2892687

[15] Sebastian Wätzoldt and Holger Giese. 2015. Modeling collaborations in adaptive systems of systems. In *ACM International Conference Proceeding Series*, Vol. 07-11-Sept. Association for Computing Machinery. https://doi.org/10.1145/2797433.2797436

[16] Martin Weisbach et al. 2017. Decentralized coordination of dynamic software updates in the Internet of Things. *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016* (2017), 171–176. https://doi.org/10.1109/WF-IoT.2016.7845450

[17] Danny Weyns et al. 2017. Perpetual Assurances for Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems III. Assurances*, Rogério de Lemos, David Garlan, Carlo Ghezzi, and Holger Giese (Eds.), Vol. 9640. Springer International Publishing, Cham, 31–63. https://doi.org/10.1007/978-3-319-74183-3_2

[18] Danny Weyns and Jesper Andersson. 2013. On the challenges of self-Adaptation in systems of systems. In *1st ACM SIGSOFT/SIGPLAN International Workshop on Software Engineering for Systems-of-Systems, SESoS 2013 Proceedings*. ACM Press, New York, New York, USA, 47–51. https://doi.org/10.1145/2489850.2489860

[19] Danny Weyns, Sam Malek, and Jesper Andersson. 2010. On decentralized self-adaptation: Lessons from the trenches and challenges for the future. In *Proceedings - International Conference on Software Engineering*. 84–93. https://doi.org/10.1145/1808984.1808994

[20] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. 2013. On patterns for decentralized control in self-adaptive systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 7475 LNCS. Springer, Berlin, Heidelberg, 76–107. https://doi.org/10.1007/978-3-642-35813-5_4