Patricia Aluko Obe, University of Duisburg-Essen
Jennifer Brings, University of Duisburg-Essen
Marian Daun, University of Duisburg-Essen
Linda Feeken, Offis e.V.
Elham Mirzaei, InSystems Automation GmbH
Martin Neumann, InSystems Automation GmbH
Jochen Nickles, Siemens AG
Simon Rösel, Model Engineering Solutions GmbH
Markus Sauer, Siemens AG
Holger Schlingloff, Fraunhofer FOKUS
Ingo Stierand, Offis e.V.
Jan-Stefan Zernickel, InSystems Automation GmbH

9

Goal-Based Strategy Exploration

When collaborative embedded systems (CESs) connect to form a group, this collaborative system group (CSG) can achieve goals that are beyond the reach of individual systems. The goals such a group can achieve depend on the constituent collaborative embedded systems. Consequently, the ability of a collaborative system group to adapt itself is driven by the capabilities of its collaborative embedded systems. This tight interconnection impedes the manual handling of adaptation strategies. Therefore, this chapter introduces a goal-based approach for strategy exploration that considers the peculiarities of collaborative system groups and collaborative embedded systems. The chapter sets out the model-based approach to adaptive system (group) design, incorporating the goals of collaborative system groups and individual systems, and outlines corresponding automated validation methods. We demonstrate the applicability of our approach for a case example of collaborative transport robots.

W. Böhm et al. (eds.), *Model-Based Engineering of Collaborative Embedded Systems*,
https://doi.org/10.1007/978-3-030-62136-0_9

9.1 Introduction

Challenges

The development of collaborative embedded systems (CESs) faces challenges due to the high degree of complexity that results from the interplay of various CESs within a collaborative system group (CSG). CSGs are formed by CESs to achieve goals that individual CESs cannot achieve on their own. For example, collaborative autonomous transport robots (CESs) can form fleets (CSGs) to optimize the transportation of goods in a factory. In a CSG, it is not only the CESs that have goals; the CSG also has goals which in turn result from the goals of the CESs. However, the different goals may be partially contradictory. For example, an individual robot may be interested in conserving its battery life, while the fleet as a whole is interested in minimizing disruption in production. The decentralized organization and the dynamicity of such CSGs (for example, robots may join or leave the fleet during runtime) makes them highly complex and their development challenging.

Goal-based strategy exploration

Therefore, we propose a goal-based approach for strategy exploration that considers the peculiarities of CSGs and CESs. In detail, we introduce a goal modeling approach tailored to the specification of goals for CESs and CSGs and show how strategies can be developed based on these goals and operationalized. We demonstrate the applicability of our approach for a case example from the industry automation domain. Specifically, we illustrate the impact that different strategies for a fleet of autonomous transport robots have on the fulfilment of goals by the individual robots.

9.2 Goal Modeling for Collaborative System Groups

Goals document objectives and rationales

Goal models are used—often in the early development phases—to document objectives that a system under development should achieve [van Lamsweerde 2001]. Goals typically document the rationales for more concrete and technical requirements, as well as design decisions [van Lamsweerde 2001], [Yu 1997]. For a recent overview of goal modeling, please refer to [Horkoff et al. 2017]. While several goal modeling languages have been proposed, we focus on the use of the Goal-oriented Requirement Language (GRL), which is part of the ITU-standardized User Requirements Notation [International Telecommunication Union 2012] and a good fit for CESs and CSGs (cf. [Daun et al. 2019], [Brings et al. 2020]).

Goal-oriented Requirements Language

GRL is based on the i* framework [Yu 1997] but is less restrictive regarding the use of the notational elements (cf. [Horkoff et al. 2008]). GRL encompasses the use of actors to denote stakeholders that have some goals to be achieved by using a system under development — that is, they may depend on other actors to achieve their goals. Most notably, in agent-oriented software engineering, actors are also used to model technical systems [Bresciani et al. 2004], as is the case here.

Intentional elements

In GRL, the intentions of actors can be further specified by means of several types of intentional elements. Intentional elements are subdivided into (hard) goals, soft goals, tasks, resources, and beliefs, which are related via decompositions and contribution links. In the following, we illustrate how the use of goal modeling can foster the engineering of CSGs.

Local goals and global goals

No reasonable system is supposed to behave absolutely arbitrarily. Each system has goals that it has to achieve. A CSG is formed through the cooperation of the CESs, therefore the goals of the CSG must be considered as early as during the design phase. The following distinction between goals is quite natural: we refer to *local goals* as goals of a CES and *global goals* as goals of a CSG. Global goals are goals that each CES of the CSG aims to achieve, while local goals are goals that represent the interests of individual CESs.

CSG goals vs. CES goals

This allows us to separate goals of the CSG from goals of the CESs but also to denote relationships between both — for example, to identify where the CSG depends on the individual CESs in its goal fulfillment and vice versa. This is important information, as we will see later on that these dependencies drive design decisions and result in the definition of explicit strategies. Figure 9-1 shows an excerpt from the GRL goal model of an individual transport robot. The goal model emphasizes the transportation-related goals of the CESs. The goal model depicts the robots' responsibilities for route calculation, performing transport tasks, and bidding for transport tasks. For each robot, detailed sub-goals are derived that, for example, define how charging is to be handled and how robot breakdowns must be treated.

Hierarchically structuring goals

As we can see, the goal model specifies three important high-level goals (which are defined directly after the root node) regarding the safety of humans, conducting the transport, and the robustness of the robot. These goals are then refined until fine-grained tasks are reached, such as the tasks to determine obstacle positions and to communicate the obstacle's position. These goals identified for the individual robots are closely related to the overall goals of the CSG—the fleet of robots—as each individual robot depends on the fleet of

robots and vice versa. We investigated this issue in [Brings et al. 2019], [Brings et al. 2020] in more detail.

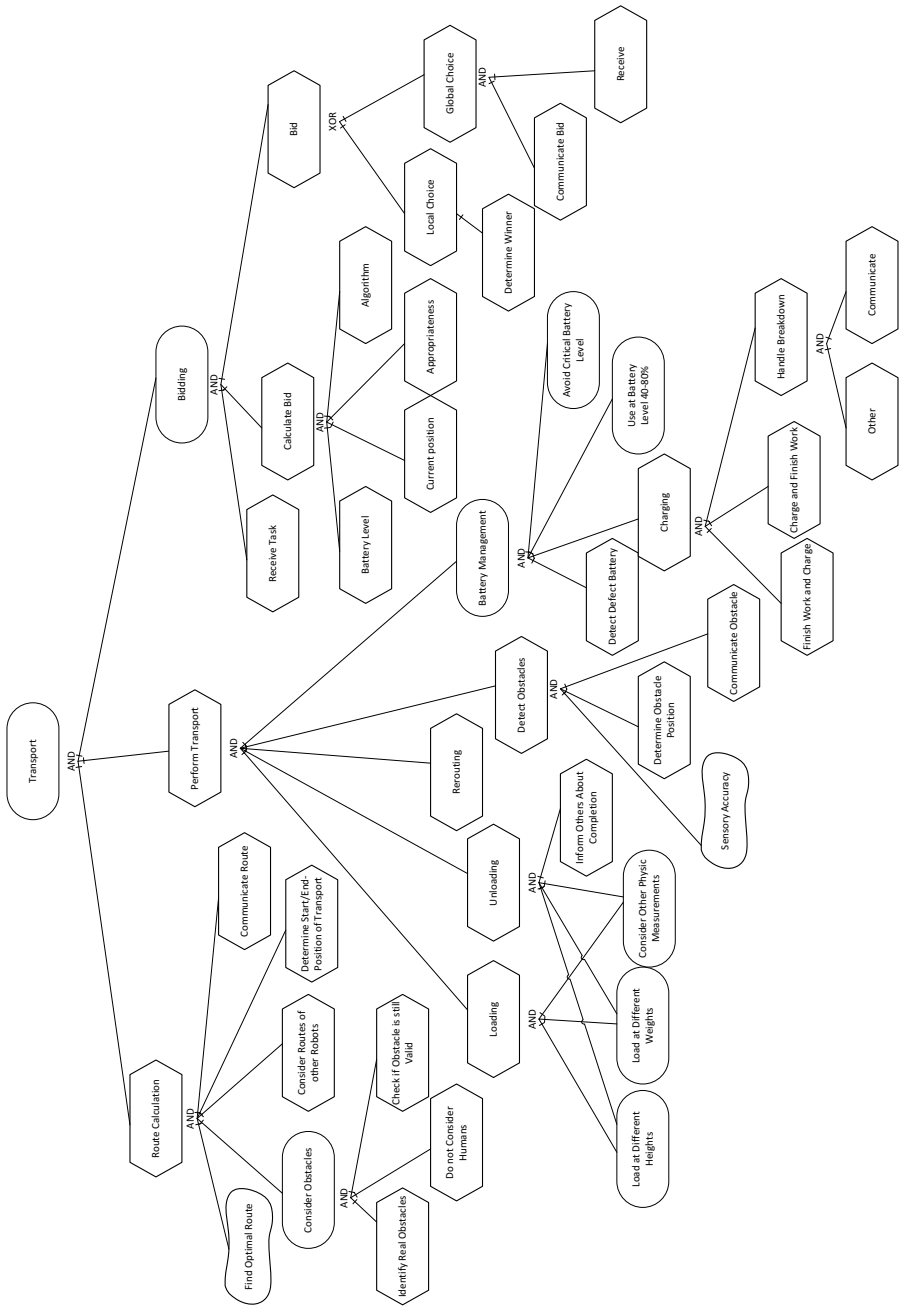

***Fig. 9-1**: GRL goal model for the individual transport robots*

9.3 Goal-Based Strategy Development

Collaboration enables goal achievement

Collaboration can enable embedded systems to achieve goals that cannot be achieved by a single system on its own. Having identified such goals (e.g., by using the approach from Section 9.2), one of the next engineering challenges is to develop behavior (or strategies) for the collaborating systems that will enable the goals to be achieved. In this section, we present concepts for going from identified goals towards behavior and we apply the concepts to an example in the use case of autonomous transport robots.

Starting with goals

The goals of a CSG and of the individual CESs describe what the systems should achieve at runtime. In general, those goals are described in a way that is understandable for humans. There is no specification of the conditions under which the goals are achieved or how to identify goal fulfillment. For example, for a goal such as "the maintenance costs of the system group are minimized," at design time, there is no information about which maintenance costs are minimal but still realistic: maintenance costs of zero are desirable but this cannot be achieved in a dynamic system. Additionally, there is no specified point in time for checking whether this goal is fulfilled and, furthermore, it is not clear what measurements are needed to predict maintenance costs.

Defining key performance indicators

Key performance indicators (KPIs) are used to make goal fulfillment measurable: KPIs relate goals to observable system variables and measure the degree to which goals are fulfilled over time. Each goal is reflected by at least one KPI in order to enable assessment of the system behavior at each point in time with respect to the goals. We look at KPIs in more detail in Section 9.4.

Defining strategies

We describe system behavior in terms of strategies: from a formal perspective, a strategy of a system is a function that maps the history of the system behavior and its context to a (new) valuation of the system variables. In other words, a strategy is an instruction for the system regarding how to behave in any situation it could face. When designing a collaborative embedded system, we aim for strategies that ensure the fulfillment of all goals "as well as possible," using KPIs to measure the degree of goal fulfillment.

Conflicts

Due to potential conflicts between goals and unpredictable context behavior, it is not always possible to find a strategy that fulfills all goals completely. Hence, we have to decide which strategy is the best match for the goals, even if there is no perfect solution. The definition of a quality measure for strategies supports this decision: a quality measure is a partial order relation on the set of all strategies for a

system. In particular, a quality measure is a function that takes two strategies of a system and decides whether one of those strategies is better than the other one. Not all strategies are comparable, hence the quality measure is only a partial order. By deciding whether one strategy is better than another one, the quality measure resolves potential conflicts between goals — for example, by prioritizing the list of existing KPIs. The relationship between goals, KPIs, and quality measures is summarized in Figure 9-2.

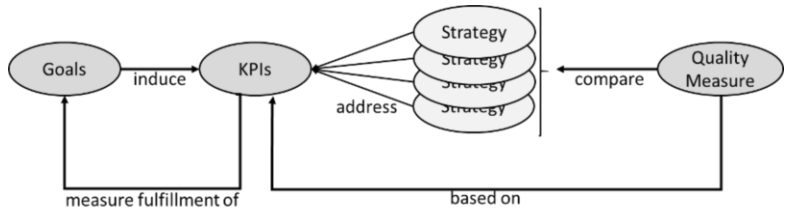

***Fig. 9-2**: Relationship between goals, KPIs, and quality measures of strategies*

In the following, we present a proof of concept example that illustrates the benefit of defining KPIs and quality measures based on goals for finding appropriate strategies.

Application to the autonomous transport robots

In the use case of autonomous transport robots, one of the key objectives is the transformation from a central fleet management to a decentral one. This requires a definition of how the robots distribute transport tasks among the fleet such that not only goals of individual robots, but also goals of the complete fleet, are fulfilled. In this example, we focus on the distribution of transport tasks and the global goal of having equal wear and tear among all robots of the fleet. This goal has industrial relevance, since it supports predictive maintenance and a reduction in maintenance costs for the robots — since all robots can be maintained in a single appointment with a service team instead of needing a service team every time a robot reaches some given threshold for distance driven.

Making goals measurable

To make this goal measurable, we define a KPI for the difference between distances driven per robot. More precisely, we observe the difference between the robot with the least distance driven and the robot with the most distance driven. To keep the example simple, we omit additional KPIs that may also be relevant for the goal. We also focus on the decision on the distribution of transport jobs. Here, therefore, a strategy for a transport robot is determined completely by defining which transport tasks the robot takes over.

Quality measures for strategies

We define the quality measure for strategies as follows:

A strategy s is better than strategy u if the KPI "difference between the robot with the least distance driven and the robot with the most distance driven" after fulfilling a task is smaller (or equal) when applying s than when applying u.

Alternative strategies

We define three alternative strategies for task distribution among robots:

- Strategy 1: The robot with the lowest distance covered so far takes over the job.
- Strategy 2: The robot with the lowest additional distance to cover for the task fulfillment takes over the job.
- Strategy 3: For each robot, calculate the difference in the distance covered if the robot takes over the job. The robot with the smallest calculated difference value takes over the job.

Comparing strategies

Considering the quality measure introduced above, we can use examples to show that the first two strategies are not comparable. Furthermore, we can formally show that the third strategy is better than the first and the second one. This qualitative comparison can be complemented by a quantitative simulation-based comparison: we used a MATLAB model of the fleet of robots and generated 100 random topologies for the factory, each defined by the distances between relevant locations in the factory, such as machines, charging stations, and storage facilities. Each factory generated consists of 5 to 30 relevant locations. The distances between locations were chosen at between 5 and 50 meters. For each of those factory maps, we generated a list of 100 transport tasks between locations of the respective factory. Each of the three strategies for distributing the tasks among the fleet of robots was applied. The number of robots per fleet was chosen randomly as between 2 and 20. As the initial state of the fleet, each robot was randomly set to one of the locations, and an initial value for the distance already covered was chosen randomly as between 0 and 200 meters. After each task distribution, the difference between the minimum and maximum costs of the robots after fulfilling all tasks they took over was calculated according to the quality measure. The simulation showed the following results:

1 In the simulation, we were able to verify that strategy 3 performs better than the two other strategies with regard to the quality measure defined above. Additionally, the

simulation showed that strategy 1 and strategy 2 are not comparable.

2 As a quantitative result of the strategy comparison, we observed that the average differences between the maximum and minimum costs of robots of the same fleet is the smallest if the robots behave according to strategy 3: the mean cost difference between robots with strategy 3 is just 81% of the cost difference between robots with strategy 1, and just 88% of the cost difference between robots with strategy 2.

Figure 9-3 shows the evolution of the average difference between the highest and lowest costs of robots over the number of distributed transport tasks for the three strategies. The figure illustrates that strategy 3 performs the best with regard to minimization of cost differences. Strategy 1 has the least convincing performance.

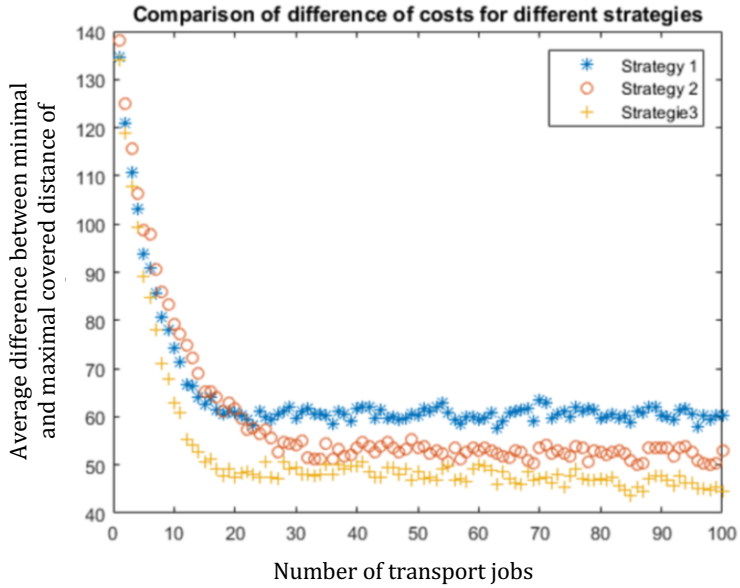

***Fig. 9-3**: Simulative comparison of cost differences for different strategies*

Choosing strategies impacts fleet performance

Our simulation shows that choosing the right strategy has a measurable impact on the performance of the fleet of robots. An explicit definition of a quality measure for strategies in the early design phase allows us to identify good strategies that had not been thought of before (here, Strategy 3, the best of the strategies considered, was introduced as a new strategy after the definition of the quality measure). Hence, considering strategies and quality

measures in the development of autonomous transport robots is a method that helps to improve the performance of the fleet significantly. In our evaluation, the fulfillment of the optimization goal "equal cost distribution among the fleet" has been improved by nearly 20%.

9.4 Goal Operationalization (KPI Development)

Developing a fleet of robots capable of performing transport jobs without central management necessitates good tracking of fleet/robot performance in fulfilling a set of goals.

Goals in the fleet of robots

The fleet of robots must fulfill multi-level goals. Some of the goals must be fulfilled absolutely, while others can be fulfilled in relation to other goals. This defines some kind of trade-off between these goals that have no conflicts with each other. In other words, the level of fulfillment of these goals must be measured and analyzed at runtime to evaluate the strategy performance that defines how and with which priority these goals must be accomplished.

Determining goal fulfillment using KPIs

In order to measure the fulfillment of the goals as well as determine how well the fleet/automated guided vehicle (AGV) is performing, we have to define KPIs. These KPIs serve as feedback data to the strategy components, which can lead to strategy adjustments to achieve better goal accomplishment if the present accomplishment is not good enough. In a multi-level goal system, it is helpful to categorize the KPIs. These categorizations are specific to a use case and would make it easier to define the trade-off between goals. In other words, this would give indications of the prioritization of such a goal, as well as of the definition of the interconnections between the goals and also the impact of not achieving them among each other. An example of KPI categorization for autonomous transport robots is as follows:

- ❑ Local/global KPI
- ❑ Historical/real-time KPI

Definition 9-4*: Local KPIs*

The term local KPI refers to the indicator that reveals how much a local goal is fulfilled during the performance of the robot.

Definition 9-5*: Global KPI*

The term global KPI refers to the indicator that reveals how much a global goal is fulfilled during the performance of the fleet of robots.

Definition 9-6*: Historical KPI*

These KPIs reveal the quantity of goals fulfilled during a certain period of the fleet performance — for example, one week. These KPIs can be used for long-term analysis and adjustments of the strategy catalog at design time.

Definition 9-7*: Real-time KPI*

These KPIs reveal the quantity of goals fulfilled at runtime. These KPIs can be used in analysis to adjust a strategy at runtime.

Example 9-8: Decentralized fleet of robots

As an example, the following table shows a list of goals and their relevant KPIs.

Table		KPIs	
Goals	Battery health and safety	Minimum/maximum cell voltage [V]	Real-time/local
	Equal wear and tear	Distance covered by each robot in a given time frame compared to the distances covered by other robots	Real-time/local
	As soon as possible job fulfillment	The difference between the earliest pick-up time and real pick-up time [s]	Historical/global
	As soon as necessary fulfillment	The time frame between real delivery time and latest due date per transport job [s]	Historical/global

A set of goals for the autonomous transport robot use case is as follows:

- **Battery health and safety**: Transport robots must not let their battery deplete or overcharge.
- **Equal wear and tear**: The transport robots must operate in such a way that all transport robots of the same age show approximately the same usage.

- **As soon as possible fulfillment**: The fleet of robots must fulfill all incoming jobs as soon as they are requested.
- **As soon as necessary fulfillment**: The fleet of robots must fulfill all incoming jobs exactly at the time they are expected to.

By formalizing the KPIs identified and implementing them in a monitoring tool, we can keep track of how well a strategy is fulfilling the desired set of goals.

9.5 Modeling Methodology for Adaptive Systems with MATLAB/Simulink

Adaptive systems face a plethora of complex scenarios to be accounted for

The development of CESs/CSGs requires a well-founded approach for dealing with a number of difficulties that result from the high complexity of the scenarios involved and that have to be incorporated. For instance, an autonomous fleet of robots must react to dynamic changes in the policy of the manufacturing execution system (MES), or the number and nature of its members, in such a way that the overall functionality and efficiency of the CSG is safeguarded. To give an example, the virtual exploration of strategies to address different goals is essential to improve a system's efficiency, cf. Section 9.2. In this context, the consistent application of a model-based development process for CESs offers a variety of benefits, such as early and systematic validation of functional requirements that describe the CSG/CES behavior. Different engineering solutions can be based on suitable system model variants that are validated and compared in a fully or partly virtual context. Moreover, Simulink models can interface with typical robot middleware or communication frameworks, such as the Robot Operating System (ROS). For instance, Simulink models may define ROS nodes or generate standalone ROS nodes based on C++ for use in an ROS network.

Need for tailored tool chains

The model-based approach greatly benefits from tailored tool chains, which automate a large number of development activities, including requirements management, modeling and simulation, as well as integrated quality assurance. For instance, in the case of the fleet of robots, the monitoring of the distribution process for incoming tasks can be automatically included in the Simulink model. The virtual representation provides a sound foundation for developing, maintaining, and extending the actual system and its hardware/software/mechanical components efficiently.

Using Simulink

With regard to the model notation, the domain-independent language Simulink is suitable for describing the functional behavior of

the CSG and the CESs as well as their context. In the case of a fleet of robots, the manufacturing execution system broadcasts different global goals dynamically to the fleet of robots. Typically, the global goals define a trade-off between the following competing objectives:

1. Economy: Minimize the total distance driven by all CESs — i.e., the transport robots.
2. Robustness: Keep the job queue lengths of each robot as short as possible.
3. Performance: Maximize the number of jobs executed per time unit.
4. Maintenance: Distribute the tasks such that all robots drive a similar distance.

As mentioned in the preceding paragraphs, KPIs are used to represent the goals in a measurable way. A suitable collaboration strategy for the collaborative robot fleet members must be designed corresponding to the given goals, cf. Sections 9.2 and 9.3. Therefore, the fundamental part of the modeling is dedicated to the distribution of the incoming transport jobs depending on the dynamically changing objectives. The collaborative fleet of robots consists of a finite number of robots that redundantly control and maintain the required data structures, such as job queues, distances driven, and their batteries' states of charge. Based on this data, a bidding process determines the collaborative robot fleet member with the lowest job execution cost. The global goals are encoded using a suitable bidding parameter vector. The context model, which represents the highest level in the hierarchy of system models, describes the interaction of the transport robot with its environment — for example, the manufacturing execution system. Furthermore, a suitable transport robot architecture that is capable of addressing adaptivity can be introduced based on a hierarchical decomposition. This approach yields a decomposition-type model that defines each transport robot's components and interfaces. Most notably, the collaborative AGV controller (CAC) hosts the logic for calculating the bidding values based on the current system state and goals. Correspondingly, each CAC model consists of the following:

- A reconfiguration unit, which is triggered whenever a new transport job is published or the collaborative robot fleet constituents are altered

- A processing unit for the transport robot goals — that is, bidding values for the autonomous task distribution are computed from the CAC data, as well as from the bidding parameters associated with the currently active transport robot goal and the member-specific local goals (e.g., maintaining a minimum battery level)
- A bidding unit that determines which robot receives the published task
- A unit that holds and updates the CAC data (battery level, path lengths, etc.)
- Units that manage the interface with ROS to determine path lengths and battery states

Figure 9-9 shows the resulting components in the system decomposition model. The system behavior is fully composed from the behavior models of each component. These component-related behavioral models represent the third level in the hierarchy of system models.

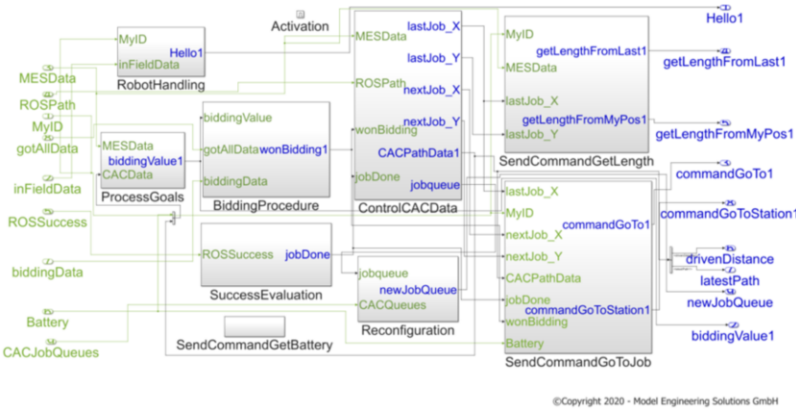

***Fig. 9-9**: System decomposition model in Simulink*

Capturing MES policies in the requirements

The expected adaptive system response, which is subject to dynamically varying manufacturing execution system policies, must be fully captured in the requirements of the fleet of robots. Compared to natural language-based approaches, which are still widely used in practice, formalized requirement formats give rise to unambiguous representations of requirements of the fleet of robots. Moreover, with the model-based approach, formalized requirement formats can be fully integrated in the sense that state-based or event-based triggers and the required signal response can be fully defined using references to model entities, such as signal specifications or design parameters.

In conjunction with the efficient definition of appropriate test cases, virtual validation of adaptive CSG behavior can be automated based on automatic test execution and assessment. The assessment relies on the comparison of the logged output signals of the executable Simulink CSG model with the expected output signals as defined in the formalized requirement.

9.6 Collaboration Framework for Goal-Based Strategies

9.6.1 Fleet Management in Collaborative Resource Networks

Fleet management systems coordinate resource usage

A fleet management system of the transport robots coordinates and monitors the use and status of a CSG, including the offered functionalities emerging from the available resources. For example, a group of transport robots offering the operational resource of transporting items. In a collaborative, goal-based approach, these functionalities should be realized in a decentralized fashion and distributed to the transport robots so that they can be executed collectively in a fleet of robots. As mentioned before, this requires the ability of each CES to achieve its individual goals and to contribute in an optimal way to the goals of the fleet/CSG.

The collaboration framework provides generic functionalities

A collaboration framework provides the generic collaboration functionalities needed during development and operation of the CESs and the CSG. These functionalities support the CESs in making informed decisions. Each CES, thereby, decides independently and takes appropriate actions. This allows for self-governing and self-organizing functionalities to have secure and trusted interactions between the CESs in the CSG. Most importantly, the framework must provide the capability to set up and execute interactions and communication between the CESs in the CSG.

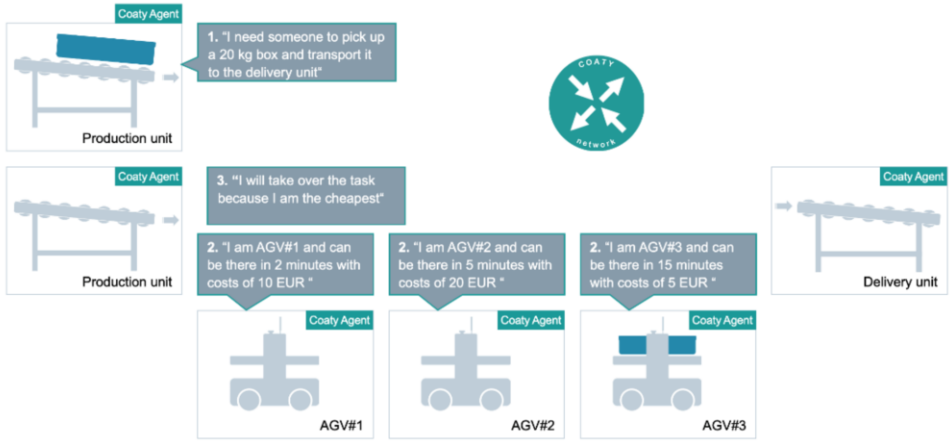

***Fig. 9-10**: Example AGV scenario for goal-based, collaborative fleet management*

Figure 9-10 shows an exemplary scenario for a collaborative, goal-based fleet management. The exemplary scenario represents a factory floor as the scope of a fleet of robots with two production units as transport robots, three AGVs as CESs, and one delivery unit as a CES. Products output by the production units must be transported to the delivery unit by autonomous transport robots. For this exemplary scenario and considering only the transport resource allocation, a collaboration framework must enable:

- ❑ The production units to announce new transport requests
- ❑ The robots to receive transport request announcements
- ❑ The robots and the productions units to coordinate the assignment and fulfillment of tasks
- ❑ All system components to monitor relevant information and behavior in order to elaborate on the level of achievement of individual and fleet goals

The following section explains how such a collaboration framework can be designed based on a set of communication patterns and a typed object model for a decentrally organized CSG. The communication patterns provide models of how the CESs can communicate and hereby interact with each other. The typed object model ensures that the basic communication data model between the CESs matches and is extensible.

9.6.2 Collaboration Framework

A collaboration framework, like the one provided by Coaty (https://coaty.io), is designed to enable autonomous Internet of Things (IoT) devices, as well as people and services, to interact in changing scenarios. Here, we apply it to a self-organizing fleet management, whereby the following properties must be fulfilled:

- Loose coupling: CESs must be able to interact independently of each other. Thus, a tight coupling with other CESs or system components would hinder the collaboration approach. A preferable approach would not apply device-centric communication concepts but instead, a decentralized and data-centric concept based on an event architecture, as typically applied in publish-subscribe communication principles. This also allows CESs to participate in or leave CSGs on demand.
- Any-to-any communication: CESs must be able to interact in one-to-one, one-to-many, many-to-one, or many-to-many CES-to-CES communication scenarios. In addition, all communication scenarios should be available as one-way communication for publishing and subscribing information topics and two-way communication for request-response communication.
- Interoperability: Besides having a standard set of communication patterns for modeling the interaction between CESs, for interoperability, an extensible data model for CES interaction must be established.
- Collaboration functions: CESs must be able to take advantage of generic collaboration functions, such as negotiation or consensus finding. This is especially important to enable implementation of the multi-level goal strategies for CES and CSG.
- Programmability, extensibility, and portability: The collaboration framework must be designed in such a way that it can be easily extended and programmed.

The IoT framework Coaty

The exemplary collaborative IoT framework Coaty fulfills these key properties. It is based on a lightweight and modular architecture that allows extensibility by means of specific connectors, adapters, building blocks, etc.

Event-based communication flows

The framework uses event-based communication flows with one-way/two-way and one-to-many/many-to-many communication patterns to realize decentralized prosumer scenarios for CESs. It thereby combines the characteristics of both classic request-response and publish-subscribe communication, but maintains the data-centric and loose-coupling characteristics. In contrast to classic client-server

systems, all participants in the system are equal in that they can act both as producers/requesters and consumers/responders. These communication patterns (cf. Figure 9-11) allow data to be discovered, queried, shared, and updated on demand in a distributed, decentralized CSG. In addition, the collaborative IoT framework Coaty allows for a distributed implementation of triggering context-specific remote operations and dynamic context-specific information routing between CESs by its IORouting concept. The IORouting concept (https://coatyio.github.io/coaty-js/man/developer-guide/#io-routing) introduces a way to dynamically route information flows between information sources of a CES and information actors that use the information. This information routing takes place based on changes in the observed operation context of the CSG. The challenging issue of reaching programmability in such highly complex, distributed, asynchronous systems of CESs in a CSG is achieved by applying the reactive programming paradigm. The extensible typed object model applied, with a set of basic core object types, can represent domain-specific system characteristics such as tasks, etc. Furthermore, each CES is represented as an object such that interoperability can be maintained without losing extensibility.

Applying this kind of collaboration framework, a set of CESs that form a CSG can collaborate by means of a decentralized interaction and communication network.

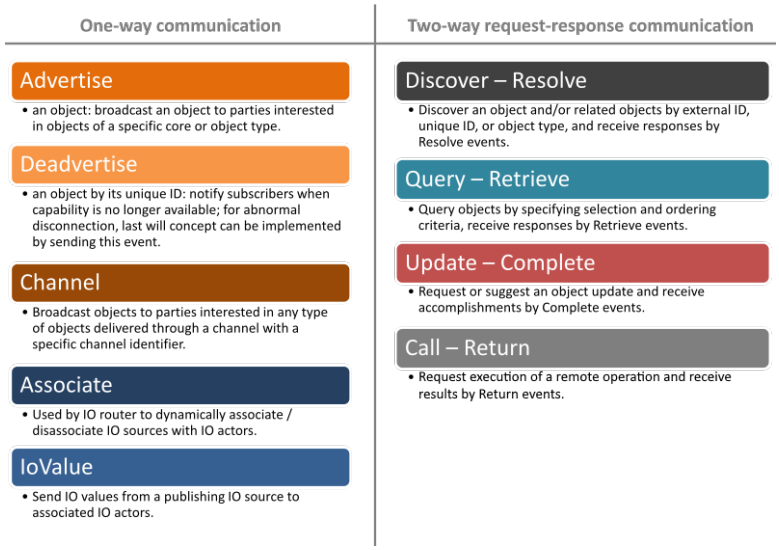

***Fig. 9-11**: Collaborative IoT framework communication pattern as realized in Coaty*

9.6.3 Collaboration Design in Decentralized Fleet Management

The collaboration framework referred to above allows us to perform and model the collaboration design of a decentralized fleet management. The following five different functional areas must be designed for the collaboration:

1. Modeling and announcement of tasks to the fleet of robots with their functional and non-functional requirements to be executed
2. Observation of these tasks by the individual transport robots
3. Monitoring local system's and fleet of robots' states at each individual transport robot
4. Application of the transport robots' goals and the goals of a fleet of robots to calculate an offer for tasks
5. Decentralized coordination of the decision, based on the CES offers, about which CES receives the task

Let us consider the exemplary scenario from Figure 9-10; this could be designed in a simplified way as follows: all transport robots observe transport tasks and other relevant system states. All systems observe transport task bids. The production unit issues a new transport task, with weight, pick-up and drop-off positions, a bidding strategy, and a bidding period. The robots calculate a cost function based on their individual goals and the fleet goals and issue the result as a transport task bid to the CSG. Each robot evaluates the bids received for the defined bidding period and then decides whether it wins the negotiation. If it does, the CES announces the self-assignment of the task and the CSG places the task in its local job queue and executes the task in accordance with its priorities in the job queue.

As mentioned before, this scenario is very simplified and does not include any failure handling etc. It shows that transport robots can interact with each other in a fleet of robots using a collaboration framework in a powerful way, allowing collaborative goal-based strategies to be modeled and implemented in a structured way that can be validated.

9.7 Conclusion

CESs connect to form a group in order to achieve local and global goals by following the best possible strategy. The interconnection between goals, KPIs, monitoring, and strategy shapes the core concept of the

goal-based strategy exploration method. In this chapter, we presented the concepts for moving from identified goals towards strategy development. We then applied the concepts to an example from the use case of collaborative autonomous transport robots. In doing so, we focused on the challenge of how to develop a set of strategies in which multi-level goals must be achieved. Therefore, the goal fulfillment must be measured and qualified for each strategy.

We also introduced the modeling tool and collaboration framework to support the application of this approach to an industrial use case to reveal some of the challenges in forming such a CSG.

9.8 Literature

[Bresciani et al. 2004] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos: Tropos: An Agent-Oriented Software Development Methodology. In: Autonomous Agents and Multi-Agent Systems 8 (3), 2004, pp. 203–236.

[Brings et al. 2019] J. Brings, M. Daun, T. Bandyszak, V. Stricker, T. Weyer, E. Mirzaei, M. Neumann, J. S. Zernickel: Model-Based Documentation of Dynamicity Constraints for Collaborative Cyber-Physical System Architectures: Findings from an Industrial Case Study. In: Journal of Systems Architecture - Embedded Systems Design 97, 2019, pp. 153–167.

[Brings et al. 2020] J. Brings, M. Daun, T. Weyer, K. Pohl: Goal-Based Configuration Analysis for Networks of Collaborative Cyber-Physical Systems. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC '20. Brno, Czech Republic, 2020, pp. 1387–1396.

[Daun et al. 2019] M. Daun, V. Stenkova, L. Krajinski, J. Brings, T. Bandyszak, T. Weyer: Goal Modeling for Collaborative Groups of Cyber-Physical Systems with GRL: Reflections on Applicability and Limitations Based on Two Studies Conducted in Industry. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019.

[Horkoff et al. 2008] J. Horkoff, G. Elahi, S. Abdulhadi, E. Yu: Reflective Analysis of the Syntax and Semantics of the I* Framework. In: Advances in Conceptual Modeling – Challenges and Opportunities. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2008, pp. 249–260.

[Horkoff et al. 2017] J. Horkoff, F. B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos, P. Giorgini: Goal-Oriented Requirements Engineering: An Extended Systematic Mapping Study. In: Requirements Engineering, September 2017, pp. 1–28.

[International Telecommunication Union 2012] International Telecommunication Union: User Requirements Notation (URN), Z 151, 2012.

[van Lamsweerde 2001] A. van Lamsweerde: Goal-Oriented Requirements Engineering: A Guided Tour. In: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, 2001, pp. 249–62.

[Yu 1997] E. S. K. Yu: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997, pp. 226–235.