Requirements Engineering (2021) 26:325–370
https://doi.org/10.1007/s00766-021-00347-3

ORIGINAL ARTICLE

A GRL-compliant iStar extension for collaborative cyber-physical systems

Marian Daun[1] · Jennifer Brings[1] · Lisa Krajinski[1] · Viktoria Stenkova[1] · Torsten Bandyszak[1]

Received: 1 January 2020 / Accepted: 5 January 2021 / Published online: 4 February 2021

Abstract
Collaborative cyber-physical systems are capable of forming networks at runtime to achieve goals that are unachievable for individual systems. They do so by connecting to each other and exchanging information that helps them coordinate their behaviors to achieve shared goals. Their highly complex dependencies, however, are difficult to document using traditional goal modeling approaches. To help developers of collaborative cyber-physical systems leverage the advantages of goal modeling approaches, we developed a GRL-compliant extension to the popular iStar goal modeling language that takes the particularities of collaborative cyber-physical systems and their developers' needs into account. In particular, our extension provides support for explicitly distinguishing between the goals of the individual collaborative cyber-physical systems and the network and for documenting various dependencies not only among the individual collaborative cyber-physical systems but also between the individual systems and the network. We provide abstract syntax, concrete syntax, and well-formedness rules for the extension. To illustrate the benefits of our extension for goal modeling of collaborative cyber-physical systems, we report on two case studies conducted in different industry domains.

Keywords Goal modeling · Collaborative cyber-physical systems · iStar · GRL

1 Introduction

Goal orientation has proven useful in the development of various kinds of systems [1]. Various goal modeling techniques support developers in eliciting, documenting, and validating stakeholder intentions (e.g., [2–20]). In the development of cyber-physical systems (CPS), it has also proven useful to attribute goals to systems or components rather than stakeholders [21]. This allows for documenting and reasoning about dependencies between the goals of different systems. For example, an automotive cruise control has the goal to maintain a safe distance to vehicles ahead. To achieve this goal, it relies on the electronic stability control to apply the brakes to the vehicle's wheels.

Recently there has been a trend to develop highly connected CPS, often referred to as collaborative CPS that form networks at runtime to achieve goals that cannot be achieved by individual systems [22]. For example, cooperative adaptive cruise control systems allow vehicles to form platoons, where each vehicle maintains the same speed and a safe distance to the vehicle ahead. This allows for reducing the safety distances between the vehicles, which in turn reduces fuel consumption for all following vehicles. The dependencies between goals in such a network are highly complex. Besides each system having its own goals, which can depend on the fulfilment of goals of another system in the network, the network itself has goals that entirely depend on some combination of goals fulfilled by the individual systems. For example, the goal of the platoon to maintain small safety distances depends on each vehicle in the platoon to maintain exactly the preset speed. Moreover, these networks can vary in size and often contain multiple systems of the same kind. Consequently, there is not only one possible network

✉ Marian Daun
marian.daun@paluno.uni-due.de

Jennifer Brings
jennifer.brings@paluno.uni-due.de

Lisa Krajinski
lisa.krajinski@paluno.uni-due.de

Viktoria Stenkova
viktoria.stenkova@paluno.uni-due.de

Torsten Bandyszak
torsten.bandyszak@paluno.uni-due.de

[1] University of Duisburg-Essen, Essen, Germany

configuration but a multitude of configurations that need to be considered. So, a goal might not depend on one goal to be fulfilled by one particular system in the network but rather that one or a certain number of systems fulfill certain goals. For example, collaborative transport robots can form fleets that optimize transportation of goods. They do so, by among others, maintaining a map of their surroundings. To keep this map up to date each transport robot depends on all the other robots to keep their map up to date. Traditional goal modeling techniques are ill-equipped to handle the complex dependencies between systems and between systems and the network [23]. The resulting goal models are difficult to comprehend because of their large sizes and their multitude of dependencies. Hence, there is a need to provide a goal modeling approach that takes the particularities of collaborative CPS and its developers' needs into account.

In previous work we evaluated the use of the goal-oriented requirement language (GRL) for modeling collaborative CPS [23]. While we identified that goal modeling with GRL can considerably contribute to the development of collaborative CPS, we identified several shortcomings of GRL for modeling collaborative CPS. The study was conducted using two industry case examples and involved workshops and discussions with industry partners. Hence, application of GRL for modeling collaborative CPS in industry was the major concern of the investigation.

To this end, this paper contributes a GRL-compliant extension to the well-established iStar[1] language [21, 24] to provide support in the engineering of collaborative CPS. Basing our extension on the iStar 2.0 definition given by Dalpiaz et al. [21] allows principle compatibility with other iStar extensions. In addition, as best practices and guidelines do exist for extending iStar 2.0, this supports the definition of a coherent extension. GRL compliance is desired as we determined a severe need for standardization in industry and the use of GRL[2] was highly appreciated by our industry partners. In this paper, we define requirements based on these shortcomings and provide a GRL-compliant extension of iStar for modeling collaborative CPS.

The major goal of this extension is to provide developers with a goal modeling language that leverages the advantages of goal orientation while reducing the complexity by removing the necessity to explicitly document each individual dependency in all possible network configurations. Thus, in this paper we place emphasis on the graphical modeling, particularly under consideration of reducing the complexity of the resulting models. Our aim is to improve manual analysis, understanding of depicted situations, and communication. At the current point we do not place emphasis on automated evaluation of the goal models. This is particularly for the reason that our industry partners were more interested in gaining an understanding of the system to be developed than a formal goal fulfillment analysis. As industry is typically reluctant to introduce goal modeling approaches in practice [26, 27], we develop the extension based on observed industry needs.

Our extension provides various means to reduce the complexity of documenting goals for collaborative CPS while maintaining precision, comprehensibility, and unambiguousness. In this paper, we provide abstract syntax, concrete syntax and well-formedness rules for our extension. Our extension was evaluated using two case studies: an example from the industry automation domain (a fleet of autonomous transport robots used in a smart factory) and an example from the automotive industry (a modern cooperative adaptive cruise control system). To show that this extension serves observed needs [23], we use the same case examples for investigation. In addition, the same industry partners were involved in workshops and discussions. Both case examples were provided by industry partners in the context of the CrESt-project.[3] Beside reporting on this case study evaluation, we also report findings gained from discussions of the case study with our industry partners.

This paper is structured as follows: Sect. 2 provides background information on goal modeling in general and the iStar modeling language in particular. Furthermore, we detail the specific characteristics of collaborative CPS to illustrate the shortcomings of traditional goal modeling techniques for these kinds of systems and formulate specific requirements to be addressed by our extension. Section 3 discusses related work and evaluates it w.r.t. these requirements in order to highlight the shortcomings of traditional goal modeling techniques. In Sect. 4 we present our extension including its foundations, abstract syntax, concrete syntax and well-formedness rules. The evaluation of the extension is shown in Sect. 5. Section 6 summarizes and discusses the major findings and threats to validity of our case study evaluation, while Sect. 7 concludes the paper.

[1] iStar was originally proposed by Yu et al. [24] and named i*. Later on, Dalpiaz et al. [21] defined a new metamodel for the language taking several extensions into account. This work is typically referred to as iStar 2.0. In the remainder of the paper, we use iStar to refer to approaches dealing with i* or iStar 2.0 as long as the distinction is not relevant for our extension.

[2] The goal-oriented requirement language (GRL) is standardized by Recommendation ITU-T Z.151 [25] which is issued by the International Telecommunication Union. The GRL builds upon iStar so that a common fundament between iStar and GRL is given.

[3] CrESt (Collaborative embedded systems) is a joint research project publicly funded by the German Federal Ministry for Education and Research (BMBF).

2 Background

In this section, we will briefly introduce iStar and goal modeling foundations (Sect. 2.1) and discuss characteristics of collaborative CPS (Sect. 2.2) that result in the need to define an extension to existing goal modeling approaches (Sect. 2.3).

2.1 Goal modeling

Goal modeling is an established requirements engineering technique [28]. Goal modeling helps requirements engineers in focusing on the intentions of stakeholders and documenting these in a structured format which allows for detecting relations between different goals such as dependencies and conflicts [29]. A variety of goal modeling approaches exist. Most of these approaches document goals in a tree- or graph-based fashion, which allows for decomposing goals into smaller sub-goals. Commonly used are the KAOS goal modeling language [30, 31], the iStar goal modeling language [21, 24], and the GRL [25, 32]. For a recent overview regarding the state of the art of goal-oriented requirements engineering, please refer to the systematic review by Horkoff et al. [28]. Our extension targets the popular iStar modeling language which forms the basis for the standardized goal-oriented requirement language (GRL). In Sect. 2.1.1 we provide a brief overview of iStar and in Sect. 2.1.2 we point out differences between iStar and GRL.

2.1.1 iStar

The iStar goal modeling language [21, 24] is graph-based—goal graphs are assigned to different actors (which can be human or other stakeholders, the system under development, other systems in the context, or even components of the system). Between these actors and the goals (i.e. intentional elements as goals are further differentiated) dependencies and contributions can be specified.

Therefore, core concepts underlying iStar include actors, their intentions (e.g., goals they would like to achieve) and dependencies between actors. The iStar modeling language distinguishes two different perspectives. The *Strategic Dependency (SD)* model specifies the actors that have interest in the system (and thus provide rationales for system requirements), and their dependencies. There are several dependency types. An actor may depend on goals or tasks that need to be achieved, or resources provided by some other actor. In contrast, the *Strategic Rationale* (SR) model documents the *internal* intentional elements and their relationships of an actor and thereby provides a detailed view on requirements each actor aims to achieve. iStar distinguishes four different intentional elements: goals, qualities (formerly called “soft goals”), tasks, and resources. It is also common to display both the dependencies among actors, as well as their internal intentional elements in one diagram as a combined or *hybrid* SD/SR model. This way the actor dependencies can be further detailed by, for instance, allowing to express dependencies between a goal and a task of different actors.

Figure 1 shows an exemplary iStar model, which represents an excerpt of a travel booking transaction. It shows the actors *traveler* and *travel agency*. The goal *trip booked* is either fulfilled when the task *book bundle* or the goal *trip parts booked are fulfilled*. The task *book bundle* depends on the *travel agency* regarding the dependum *trip bundle booked*.

2.1.2 Goal-oriented requirement language (GRL)

The goal-oriented requirement language (GRL) is part of the User Requirements Notation (URN) as standardized by the International Telecommunication Union (ITU) in Recommendation Z.151 URN [25]. GRL is based on a subset of iStar [33]. While GRL shares many core concepts with iStar, some differences exist. For example, GRL is less restrictive than iStar, particularly regarding the usage of relationships for linking intentional elements [32], which has also been shown to support the diversity of how goal models are actually created and used [34]. This is favored by our industry partners as it gives them more freedom to express their thoughts and reduces the number of syntactical errors in their goal models. As GRL does not prevent users from adhering to the stricter rules set by iStar, we did not observe any issues arising from the loosening of those restrictions. For a more detailed discussion regarding the differences between iStar and GRL, please refer to the work of Amyot et al. [32]. As the usage of standardized languages is of importance to our industry partner and previous work has shown the suitability of GRL for the development of collaborative CPS [35], we ensured that the proposed extension can be used with GRL as well.

2.1.3 GRL-compliant iStar extension

In our extension, we build upon concepts from both GRL and iStar. This is due to the fact that while being very similar, small differences exist that come with different advantages and disadvantages. Mainly, we target GRL due to its simplicity and its popularity among industry partners. We target iStar because there are established guidelines for extending iStar that can support the development of a high-quality extension. In addition, we reuse useful existing concepts already proposed by other iStar extensions, which helps reduce redundancy and increases acceptance.

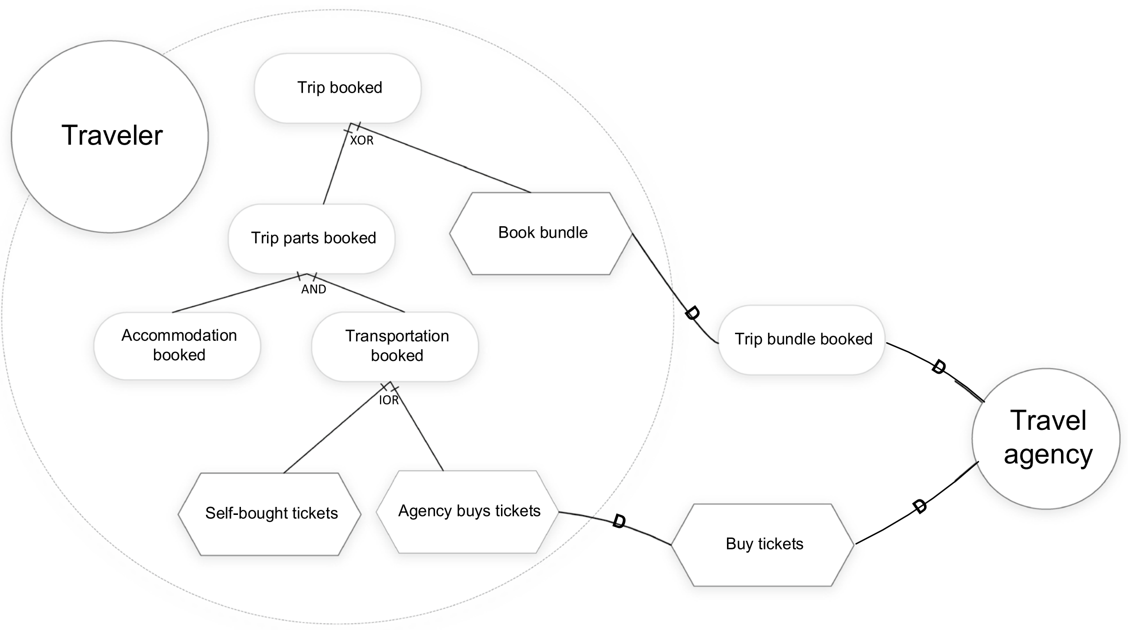
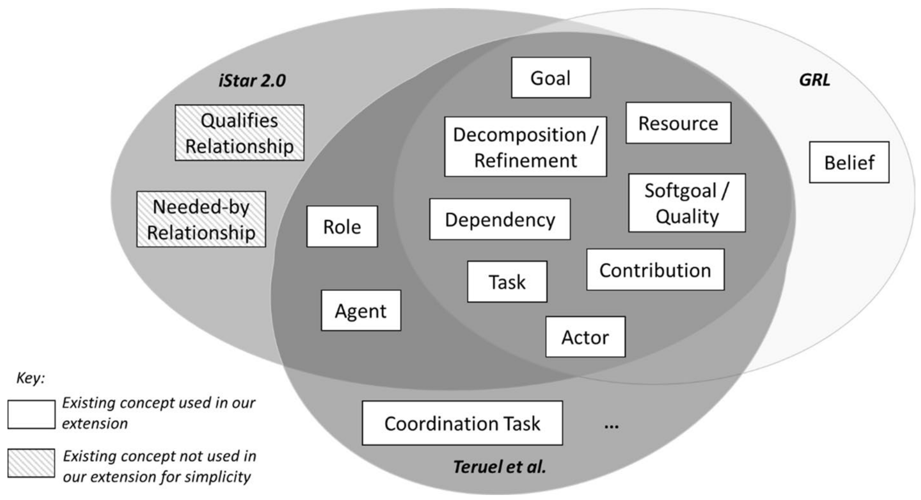


Fig. 1 iStar travel booking example (based on [21])

Fig. 2 Relation between iStar, GRL and our extension



In detail, we use existing concepts as illustrated in Fig. 2. The figure shows the main concepts from GRL, iStar, and an iStar extension from which we borrow a specific concept. As can be seen, iStar 2.0 differs from GRL in that it includes two more specialized relationship types, i.e., qualifies and needed-by relationship. Regarding the relationships between intentional elements, we stick to GRL since it is less complex and less restrictive, which better reflects industry needs as it allows for easier model creation. However, although we do not include these two specific relationship concepts in our extension, as we did not see any need, it is still possible to use them. Furthermore, iStar 2.0 defines roles and agents as specializations of actors, which we take as the basis for defining specific actor types for modeling collaborative CPS and CPS networks. In addition to iStar and GRL concepts, we use the coordination task concept from a related extension proposed by Teruel et al. [36].

2.2 Collaborative cyber-physical systems and their characteristics

CPS are software-intensive systems that closely integrate physical and software parts [22, 37, 38]. In addition, CPS are highly interactive with their environment in sensing and actuating context values and tightly communicating with other CPS [37, 39]. For example, all vehicles in a platoon

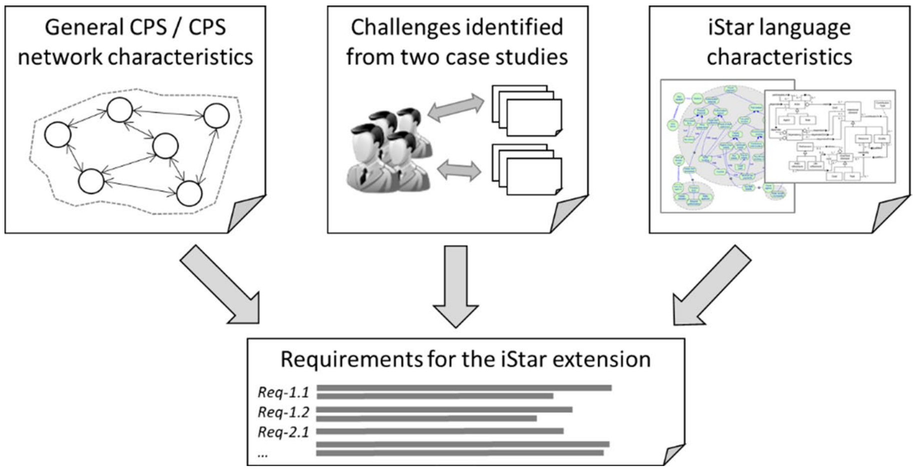

General CPS / CPS network characteristics
Challenges identified from two case studies
iStar language characteristics
Requirements for the iStar extension
Req-1.1
Req-1.2
Req-2.1
...

Fig. 3 Requirements sources for the proposed extension

record their surroundings like other vehicles or road signs with their sensors and communicate with each other via sensor data in order to offer a high level of safety.

Collaborative CPS can form networks in which different constituent systems collaborate and coordinate their activities in order to achieve goals that go beyond the goals an individual system can achieve (cf. [22]). For example, collaborative transport robots can distribute tasks among each other in such a way that all robots remain in motion and there are no overloaded or underloaded robots. This enables them to achieve a higher goal, which means that transport tasks are distributed in a coordinated manner and thus completed faster. These networks are highly dynamic as they reshape at runtime when systems join and/or leave the network. For example, a platoon reshapes as vehicles enter or leave the platoon.

Most CPS must be considered safety–critical, which consequently leads to the need for thorough engineering processes [40]. Vital parts of these engineering processes are early safety analyses. It has been shown that the use of goal models allows for application of safety analyses in very early phases [41] and is therefore considered beneficial. In case of collaborative CPS, safety can be increased through cooperation between individual systems. For example, in the automotive domain, the term "cooperative vehicle safety" is used to denote CPS applications that aim at avoiding hazards and accidents through inter-vehicle collaboration [42, 43]. On the other hand, the safety of collaborating CPS also poses additional challenges, e.g., due to the involvement of several manufacturers and the lack of a central authority governing the development and operation of CPS networks [44, 45]. As will be shown in the remainder of this paper, the use of goal models illustrating the interplay of individual systems and the network can further support increasing the safety of collaborative CPS.

2.3 Requirements for a GRL-compliant iStar extension for collaborative cyber-physical systems

Modeling collaborative CPS with iStar/GRL goal models is challenging as such goal models have the tendency to become large, complex, and thus unsuitable for human engineers and analysts. In our previous work [23] we report empirical results, from which we identified challenges for goal modeling of collaborative CPS. We conducted two case studies with industry partners from different domains. The goal of the two case studies was to systematically identify challenges and limitations of goal modeling with GRL related to the representation of typical collaborative CPS characteristics (see Sect. 2.2). Beside the general observation that goal models of collaborative CPS can easily become large and complex, we identified six major challenges regarding what needs to be represented when modeling goals of collaborative CPS and CPS networks.

We further analyzed and refined these challenges in order to derive specific, detailed requirements for extending iStar so that it allows engineers to specify collaborative CPS in a goal-oriented manner. On the one hand, these requirements are grounded in the characteristics of collaborative CPS and CPS networks. On the other hand, the requirements are also substantiated by empirical evidence from our two case studies reported in our previous work and are thus aligned with the specific needs faced by requirements engineers. Moreover, the requirements are tailored specifically for the iStar goal modeling language that shall be extended. Figure 3 illustrates the three sources that were considered during the requirements definition process.

In the following, we briefly summarize the six major challenges reported in [23] and present the respective iStar extension requirements we derived from these challenges.

2.3.1 Challenge 1: need for distinction between network and systems

Collaborative CPS form networks with other collaborative CPS, which allows them to enhance their functionality and fulfill goals they cannot fulfill on their own. It is important to be able to identify the owner of a goal; i.e. it must be distinguishable whether an individual system strives to fulfill a certain goal or just contributes to an overall goal of the network. In some cases, engineers need to reason about the CPS network's goals independent of the goals of the collaborative CPS, and in some cases, engineers need to be able to reason about the network under consideration of the individual collaborative CPS that are part of the network and their goals. Therefore, we can derive the following specific requirements for an iStar goal modeling extension:

Req-1.1: The iStar extension must allow the distinction between individual CPS and the network of CPS.

Req-1.2: The iStar extension must allow for flexibility regarding the visual representation of the relation of the CPS network and individual CPS. I.e. it must be possible to specify individual CPS as part of the CPS network and also allow for comparing the CPS network and CPS at the same level of abstraction.

2.3.2 Challenge 2: need for mirroring of goals

In many cases goals of the network rely on very similar goals of the individual systems. For example, the vehicles, which are the individual systems, have the goal to reduce their individual driving time, and the platoon, which forms the collaborative CPS network, has the goal to save the overall driving time of all vehicles. Hence, it is often the case that the network and the individual system have very similar goals that mutually depend on each other. Consequently, there is a need to assign goals to individual CPS as well as to the CPS network and to document the relations between those goals, which leads to the following requirements:

Req-2.1: The iStar extension must allow for intentional elements to be attributable to individual CPS.

Req-2.2: The iStar extension must allow for intentional elements to be attributable to CPS networks.

Req-2.3: The iStar extension must allow for documenting of mutual dependencies between intentional elements of the collaborative CPS network and collaborative CPS.

2.3.3 Challenge 3: need for considering multiple identical collaborative CPS

A collaborative CPS network may contain multiple collaborative CPS of the same type, e.g., a platoon consists of several identical vehicles. The explicit specification of each possible network is infeasible as this would require specifying not only a large number of possible network configurations, but also networks of an extremely large size. Consequently, not only is the explicit modeling of the goals for each possible network configuration infeasible, even the explicit modeling of all individual collaborative CPS in large networks is infeasible. Consequently, suitable abstractions are required to enable the modeling of multiple identical CPS whose number can vary. Therefore, we define the following requirements:

Req-3.1: The iStar extension must allow for documenting all networks without the need for modeling each possible network explicitly.

Req-3.2: The iStar extension must allow for documenting identical collaborative CPS in a network without the need for modeling each collaborative CPS individually.

2.3.4 Challenge 4: need for dependencies between systems of the same type

Another common situation that needs to be considered is a collaborative CPS relying on systems of the same type to fulfill the same goal. For example, all following vehicles in a platoon have the goal to avoid collisions, which can partly be fulfilled by regulating their speed based on each other's speeds. As the goal model cannot show each individual collaborative CPS that can be part of such a network, abstraction mechanisms are needed to adequately represent the occurrence of multiple identical systems and the dependencies between them. Particularly, there is a need to consider dependencies, where one system's intentional element relies on an intentional element from other systems of the same type. Therefore, we can derive the following specific requirements for an iStar goal modeling extension:

Req-4.1: The iStar extension must allow for documenting intentional elements of collaborative CPS of the same type.

Req-4.2: The iStar extension must allow for documenting dependencies between an intentional element of a collaborative CPS and the same intentional element of other systems of the same system type.

2.3.5 Challenge 5: need for roles and dynamic role assignments

Collaborative CPS in networks may have different responsibilities. This might even be true for identical collaborative CPS. For example, in a platoon, all collaborative CPS are vehicles, but the foremost vehicle has the role of lead vehicle and thus the responsibility for all vehicles in the platoon. Therefore, there is a need to assign roles to collaborative CPS in a network. As collaborative CPS networks are dynamic, and therefore, reshape at runtime as collaborative CPS join or leave the network, roles must be reassignable at runtime. Therefore, we can derive the following specific requirements for an iStar goal modeling extension:

Req-5.1: The iStar extension must allow for documenting different roles a collaborative CPS can be assigned.

Req-5.2: The iStar extension must allow for documenting mechanisms to reassign roles.

2.3.6 Challenge 6: need for considering conflicts between goals of the individual collaborative CPS and the CPS network

Collaborative CPS join an existing or form a new network to achieve some goals they cannot achieve by themselves. However, participating in a network may be a trade-off that impedes the fulfillment of other goals. Therefore, it is sometimes impossible to assign values to contribution links for intentional elements of the network actor because the value can be different depending on the goals of each collaborative CPS. For example, in a platoon it can happen that vehicles have a common goal they can reach together but differ in the other goals. For example, it can be important for one vehicle to drive in an environmentally friendly manner, while another vehicle in the same platoon may not consider this important. Therefore, we define the following requirement:

Req-6.1: The iStar extension must allow for contributions to be assigned variable values that can change depending on the goals of a collaborative CPS.

2.3.7 Further requirements

As already outlined, the iStar extension shall be GRL-compliant due to the fact that we found GRL well-received by our industry partners in the previous investigation. In addition, the extension shall adhere to established guidelines for iStar extensions [46]. Therefore, Table 1 gives the individual guidelines and briefly explains how they shall be achieved, and which section of this paper elaborates on the respective aspects. Note that some realizations overlap (i.e. the same approach is taken), in these cases we avoid redundancy by simply referring to the aforementioned realization.

3 Related work

For discussing the related work, we focus on three kinds of approaches commonly proposed in the state of the art. Section 3.1 will introduce goal modeling approaches for systems-of-systems, which can be interpreted as a network of collaborative CPS that is designed top to bottom, with exact knowledge about the partaking systems and their compositions. Section 3.2 discusses goal modeling approaches for multi-agent systems, which are in so far related as commonly the case is made that the agents in multi-agent systems collaborate to maximize their goal fulfillment. However, unlike for collaborative CPS, the network itself is typically not given the credit of having its own goals. Lastly, in Sect. 3.3 we review other existing extensions for the iStar goal modeling language, which we partly build upon, as we will show in Sect. 3.4.

3.1 Goal modeling approaches for systems-of-systems

Systems-of-systems (SoS) engineering is a related research area where the consideration of goals is of particular interest. Distinguishing goals of the SoS under consideration from the goals of the individual constituent systems is important in the requirements engineering for SoS [53]. These two levels (which are also sometimes called “macro level” and “micro level” [54]) of goal modeling for SoS allow analyzing collaborations between individual systems by focusing on how their individual goals contribute to SoS-level goals [55]. These contributions are conceptually described by Cavalcante et al. [55], without proposing a specific modeling notation; instead, it is referred to traditional goal modeling syntax elements, such as actors for modeling both SoS and its constituent systems. While decomposition links are mainly used within each goal modeling level, contribution links also occur between goals on different levels [55]. Additionally, Cavalcante et al. propose a new kind of link, *interaction* links, to explicitly account for emergent behavior through goals whose satisfaction results from interactions among individual systems.

In addition to such conceptual approaches, there are also specific guidelines and notations for modeling SoS goals and constituent system goals. Lewis [53] suggests creating separate AND/OR goal trees for the individual systems and the SoS in order to identify common goals in the different individual systems' goal models as well as conflicting goals, both between individual systems and the overall SoS goals. Garro and Tundis use stereotypes to characterize the goals of stakeholders and of complex SoS used to achieve these goals [56]. Additionally, relationships between these goals are modeled in a manner similar to UML use case diagrams.

According to Silva et al., closely connected to SoS goals is the *mission* concept [57]. Goals are associated to the mission of the overall SoS and the mission of the constituent systems. Thereby, the goals related to the mission of an SoS are achieved through collaboration between the individual systems. Hence, Silva et al. [58] propose a mission-centered SoS design process, covering a dedicated mission-level, where missions of individual systems and the SoS are modeled. For modeling missions in an SoS context, they propose the mKAOS approach [58–60] that builds upon the KAOS goal modeling language [31] and includes SoS-relevant extensions. For operationalizing goals, mKAOS includes

Table 1 Realization of the iStar extension guidelines from [45]

Guidelines taken from [45]		Realization
G1	Preserve the language (iStar) original syntax	It is a requirement to propose an extension that makes use of the original syntax and extends this syntax naturally. The extension of the concrete syntax will be shown in Sect. 4.3, the integration of new elements with elements of the original syntax can be seen in Sect. 5.2
G2	Carry out consistent, complete and without-conflicts extensions and follow a process/method to do them	We extend the iStar metamodel systematically to provide a clear definition and also for relating elements of the original iStar notation to the newly proposed elements. The metamodel of the extension can be found in Sect. 4.2
G3	Perform a literature review, include the participation of domain experts and iStar experts and model systems of application area before extending	We conducted a literature review on the topic to find existing iStar extensions that can contribute to the above-mentioned requirements. Section 3 will discuss related works and Sect. 3.4 will explicitly show, how these extensions can contribute to fulfilling the defined requirements. In addition, we conducted a study with domain experts to identify industry needs for an iStar extension for collaborative CPS [23][a]
G4	Describe a clear definition of the extension concepts	see G2
G5	Propose concrete and abstract syntax of the extension	We specify the abstract syntax using a metamodel that extends the iStar 2.0 metamodel. In Sect. 4.1 we introduce a GRL-compliant iStar metamodel extension and extend this in Sect. 4.2 to the specifics of collaborative CPS. We provide a definition for the concrete syntax in Sect. 4.3 and show its application to industrial case examples in Sect. 5.2. This application also allows for verifying consistency between the defined concept and the concrete syntax
G6	Check consistency between abstract and concrete syntaxes	see G5
G7	Relate concepts introduced by the extensions with the iStar concepts	see G2
G8	Define extensions with the smallest possible number of modifications and new representations in order not to complicate the use of the modeling language (iStar)	see G2
G9	Propose careful and simple graphical representations, able to be drawn on paper without a tool	The concrete syntax extensions are designed to seamlessly integrate with the existing iStar syntax. Furthermore, we define the concrete syntax based on guidelines proposed by Moody [47] to achieve a simple and intuitively usable graphical notation

[a]The industry professionals partaking have years of experience in their field and were involved in many substantial projects for their companies, partly taking leading roles. Thus, we consider them domain experts in the domains of automotive, industry automation and robotics. Among the authors of this study are researchers highly experienced with GRL and iStar. They have applied GRL and iStar in various industrial settings, published research on this topic (e.g., [23, 35, 48]), have years of experience in teaching GRL and iStar in university master level requirements engineering courses (cf. [49, 50]) and have defined an industry course teaching GRL to industry professionals [51]. The course is in use at the Schaeffler AG to teach goal modeling for the engineering of automotive CPS [52]. However, please note that the authors are no domain experts and the industry participants no GRL/iStar experts

two kinds of capability models, one of which is concerned with modeling information exchange between individual systems and the resulting capabilities the SoS provides (denoted “communicational capabilities”). Furthermore, mKAOS includes a dedicated emergent behavior model that groups and relates such SoS capabilities to resulting emergent properties/functionalities. Garcés and Nakagawa provide guidelines and recommendations for the creation of mKAOS models [61]. These also include global missions of a SoS on multiple levels of abstraction by goal refinement and abstraction to identify rationales behind a SoS’s missions.

3.2 Goal modeling approaches for multi-agent systems

Another related term is that of multi-agent systems (MAS), which refers to systems composed of several autonomous agents that collaborate in order to autonomously (i.e., without human intervention) accomplish tasks (cf. [62]). According to Wooldridge [63], apart from autonomy, reactiveness, and proactiveness, an agent has essential social abilities allowing the engagement in collaborations and interactions to jointly solve complex problems. In such a collaboration, however, an agent makes rational decisions w.r.t. maximizing its own benefit according to its agent-internal goals and interests (cf., e.g., [63]). This is reflected, for instance, in the established BDI reference model for autonomous agents (cf. [64, 65]), which describes an agent's mental attitudes by information about the current state of its surroundings (Beliefs), the set of tasks it principally aims to achieve (Desires), and the tasks it is actually carrying out (Intentions), all of which determine an agent's behavior. Multiple iStar-based agent-oriented modeling approaches have been proposed [66]. Goal-based MAS approaches typically consider goals as runtime entities that are used during operation of agents to coordinate the interaction within a MAS as well as single agents (cf., e.g., [67]). Thus, goal delegation during operation of a MAS (cf., e.g., [68]) is also an important topic for MAS development. Similarly, the operational semantics of goals as well as their dynamic lifecycle are also considered by some approaches [69].

An important concept considered for the development of MAS is the role concept. The role concept is essential for both describing the static organization and structure of a MAS, as well as for enabling the formation of multi-agent systems (cf. [70, 71]). Roles an agent can take are typically defined by a set of responsibilities and a set of permissions [72]. The current roles of an agent define its functionality and behavior, as well as the possible interactions with other roles that can be taken by other agents (cf. [73, 74]). In particular, the responsibilities can be seen as required functionalities related to a certain role [74]. A role can be responsible for carrying out a task on its own, but also be involved in a collaboration to jointly achieve some task [73]. Such a collaboration is sometimes named an "agent group", i.e. a set of agents that are related via interactions of their roles [75]. There can be relationships between roles, such as compatibility and dependencies [76, 77]. Roles can also determine a hierarchical structure of a MAS [78].

Specific goal modeling approaches for MAS include Tropos [2], where, among others, beliefs are considered as a dedicated modeling concept, in addition to the original iStar goal modeling language it is built upon. The Tropos approach comprises a methodology that covers the early and late requirements phases, where goal models are used, as well as later phases up to the implementation of agent-based software systems. In the late requirements analysis phase, the system under development is introduced as an actor and related to stakeholders using dependency relationships. Goal-based reasoning in the Tropos methodology is described in detail by Giorgini et al. [79].

The goal modeling approach proposed by Zhong and DeLoach [80] explicitly distinguishes goal classes and goal instances. The latter are created and assigned to specific agents at runtime. Furthermore, they introduce relationship types that can materialize between goals in order to specify control flow structures, such as a goal being triggered by another goal, or goal precedence (i.e., a goal requires the execution of some other goals before being allowed to become active). Goal instances are also explicitly considered by Thangarajah et al. [81], where goal models are used to identify interaction between different goals an agent may be able to achieve simultaneously. Cheong and Winikoff use so-called interaction goals, which specify goals of the interaction between different agents, to design multi-agent systems [82–84]. These interaction goals are modeled in a hierarchical goal tree.

3.3 Specific iStar goal modeling extensions

The basic iStar goal modeling language, as described in Sect. 2.1.1, has been extended by researchers in several ways. A recent survey of iStar extensions was provided by Gonçalves et al. [85]. In the following, we will review some of the approaches that are related to our approach.

Teruel et al. proposed an iStar extension for collaborative systems [36, 86, 87]. In this approach, the term "collaborative system", however, is not used to denote the kind of system that is in focus of our work (cf. Sect. 2.2). Instead, it refers to information systems that support the collaboration between humans, e.g., collaborative implementation of code with the help of a version control software like git. The approach of Teruel et al. aims at specifying requirements of such collaborative systems. Hence, the proposed extensions to iStar reflect the collaboration between humans, which results in the definition of additional concepts. Specifically, Teruel et al. propose different task types, i.e., individual tasks of single users as well as collaboration tasks, communication tasks, and coordination tasks. The latter three types of tasks are used to model tasks in which two or more users are involved and are based on the established 3C conceptual model for groupware [88]. Along with these task specializations, participation links are proposed to model which user is involved in which (collaboration, communication, or coordination) task. Cardinality constraints attached to these participation links specify the number of users that can be involved in a task. Furthermore, responsibility links are used to capture goal and task responsibilities of users, which separates responsibility from actually carrying out some collaboration activity. Again, based on the 3C model, Teruel et al. consider a user's awareness of other users' activities in the form of awareness softgoals and awareness resources.

Ali et al. propose a goal modeling approach that enhances Tropos goal models with context information [3]. In this approach, variability that is present in the context of a system under consideration is captured through annotations of goals as well as decomposition, dependency, and contribution links. That way, conditional achievement of goals, depending on relevant context properties, can be modeled. As a result, the overall annotated goal model specifies goal model variants, i.e., different ways goals can be achieved, depending on context information. Ali et al. use the contextual goal modeling approach to support the deployment of variable systems into environments that also contain variable parts [89]. Another approach dealing with variability is presented by Silva et al. [90]. Goal models are used to explicitly document variability of software product lines. Therefore, cardinalities are introduced for different intentional elements as well as for the means-end links connecting variable intentional elements. Borba and Silva, additionally to the cardinality concept, suggest the explicit mapping of feature models and goal models [91].

Another related iStar extension [92] aims at modeling ambient intelligent systems that are deeply embedded in daily human activities and invisible to their users. Such ambient systems, similar to CPS, integrate the physical surroundings and computation, but also human users. Most notably, the approach relates to goal modeling for collaborative CPS in that it utilizes actor decomposition relationships to constituent components of ambient systems. That way, actors being composed of other actors can be modeled. In addition, communication links between actors, including communication between users and technology, as well as between different technological components/subsystems are defined.

Other iStar and GRL extensions often propose the use of stereotypes to document additional information. For instance, Marosin and Ghanavati propose the annotation of vague and informal information in goals, softgoals, and tasks via stereotypes [93]. Gailly et al. propose the documentation of domain knowledge that is annotated using stereotypes and defined using an ontology-based approach [94].

3.4 Requirements evaluation

In summary, there exist a multitude of approaches that can contribute to the individual requirements defined in Sect. 2.3. Table 2 summarizes the state of the art with respect to the requirements. However, existing approaches are typically not capable of fulfilling more than one requirement and not all requirements can be fulfilled. Nevertheless, the integration and harmonizing of existing works can support the definition of a coherent solution concept, as we will show in Sect. 4.

4 GRL-compliant iStar extension for modeling collaborative cyber-physical systems

In Sect. 2, we introduced iStar and GRL as the foundation our extension builds upon in detail and discussed the requirements for the extension. As already some extensions or modifications to the iStar language exist, which at least can be partly used to address some of the challenges of goal modeling for collaborative CPS, we do not rely on the pure version of the iStar language but an adapted one. As outlined above, we had the requirement to develop an extension compliant with the GRL. In addition, we make use of different already existing extensions that provided us with already established modeling concepts. This is outlined in Sect. 4.1. We build our final extension in Sect. 4.2 on this initial metamodel consisting of the combination and integration of proposed concepts from the related work. Based on the metamodel introduced in Sects. 4.2, and 4.3 defines the concrete syntax for the new modeling elements. In Sect. 4.4 well-formedness rules are defined and Sect. 4.5 presents tool support for creating models according to the extension.

In the following subsections, a cooperative adaptive cruise control system (CACC, [95]) is used as a running example to motivate the need of the metamodel extensions and to illustrate the concrete syntax. A CACC is a modern version of a common adaptive cruise control (ACC). An ACC is a cruise control system that, in addition to the cruise control function, also ensures that the distance to the vehicle ahead does not underrun a safe minimum distance. The CACC is a collaborative CPS that also communicates with other CACCs. Thus, they form a platoon (i.e. the CPS network) which allows driving with minimized distances between the partaking vehicles. This reduces fuel consumption, emissions, and increases traffic throughput on motorways [96].

4.1 Foundations for the metamodel of the extension

Figure 4 shows the metamodel for the goal modeling language upon which we build our extension. The goal modeling language can be considered a combination of the iStar language and GRL. The metamodel is similar to the metamodel defined by Dalpiaz et al. [21]. In the following we use UML class diagrams to define the metamodel. This ensures comparability with the definition of iStar 2.0 by Dalpiaz et al. [21], who also used UML class diagrams for metamodel definition. However, some adjustments have been made to maintain compatibility with GRL. For example, we removed various restrictions. An intentional element can contribute to any other kind of intentional element, not just to softgoals,[4] and all intentional elements can be refined not just goals and tasks. Regarding the refinement, the OR-Refinement was further separated into an IOR-Refinement and an XOR-Refinement. A further intentional element, defined by GRL, the belief was added. We do not include the agent concept

[4] In accordance with GRL we use the term softgoal instead of quality.

Table 2 Possible contributions from the state of the art to achieve the requirements

Requirement		Contribution from state of the art
Req-1.1	*The iStar extension must allow the distinction between individual CPS and the network of CPS*	Differentiation of actors, as is commonly suggested to be done by using stereotypes (cf [55, 93, 94]), can contribute to the fulfillment of this requirement
Req-1.2	*The iStar extension must allow for flexibility regarding the visual representation of the relation of the CPS network and individual CPS. I.e. it must be possible to specify individual CPS as part of the CPS network and also allow for comparing the CPS network and CPS on the same level of abstraction*	The composition of actors (cf. [92]) which allows for depicting how a CPS network can be modeled as composed of individual collaborative CPS, can contribute to the fulfillment of this requirement
Req-2.1	*The iStar extension must allow for intentional elements to be attributable to individual CPS*	Specifying separate goal models for the individual systems and the overall system (cf. [53]) can contribute to the fulfillment of this requirement
Req-2.2	*The iStar extension must allow for intentional elements to be attributable to CPS networks*	see Req-2.1
Req-2.3	*The iStar extension must allow for documenting of mutual dependencies between intentional elements of the collaborative CPS network and collaborative CPS*	*Interaction* links, to explicitly account for emergent behavior through goals whose satisfaction results from interactions among individual systems. (cf. [56]) can contribute to the fulfillment of this requirement
		Using UML use case-like diagrams to express the relation between individual goals and network goals (cf. [57]) can contribute to the fulfillment of this requirement
		A *mission* concept to express that goals of individual systems are used to achieve the overall system's goals (cf. [56]) can contribute to the fulfillment of this requirement
Req-3.1	*The iStar extension must allow for documenting all networks without the need for modeling each possible network explicitly*	–
Req-3.2	*The iStar extension must allow for documenting identical collaborative CPS in a network without the need for modeling each collaborative CPS individually*	The concept of multiplicities and cardinalities from variability modeling and software product line engineering can be used to depict actors that can occur multiple times [3, 90, 91], which can contribute to the fulfillment of this requirement
Req-4.1	*The iStar extension must allow for documenting intentional elements of collaborative CPS of the same type*	see Req-3.2
Req-4.2	*The iStar extension must allow for documenting dependencies between an intentional element of a collaborative CPS and the same intentional element of other systems of the same system type*	–
Req-5.1	*The iStar extension must allow for documenting different roles a collaborative CPS can be assigned*	Introducing a role concept (cf. [70, 71]) to differentiate between different roles that a collaborative CPS needs to take within the CPS network can contribute to the fulfillment of this requirement
Req-5.2	*The iStar extension must allow for documenting mechanisms to reassign roles*	Using the concept of coordination tasks (cf. [36, 86, 87]) to express the assignment of a role to a certain actor can contribute to the fulfillment of this requirement
Req-6.1	*The iStar extension must allow for contributions to be assigned variable values that can change depending on the goals of a collaborative CPS*	–

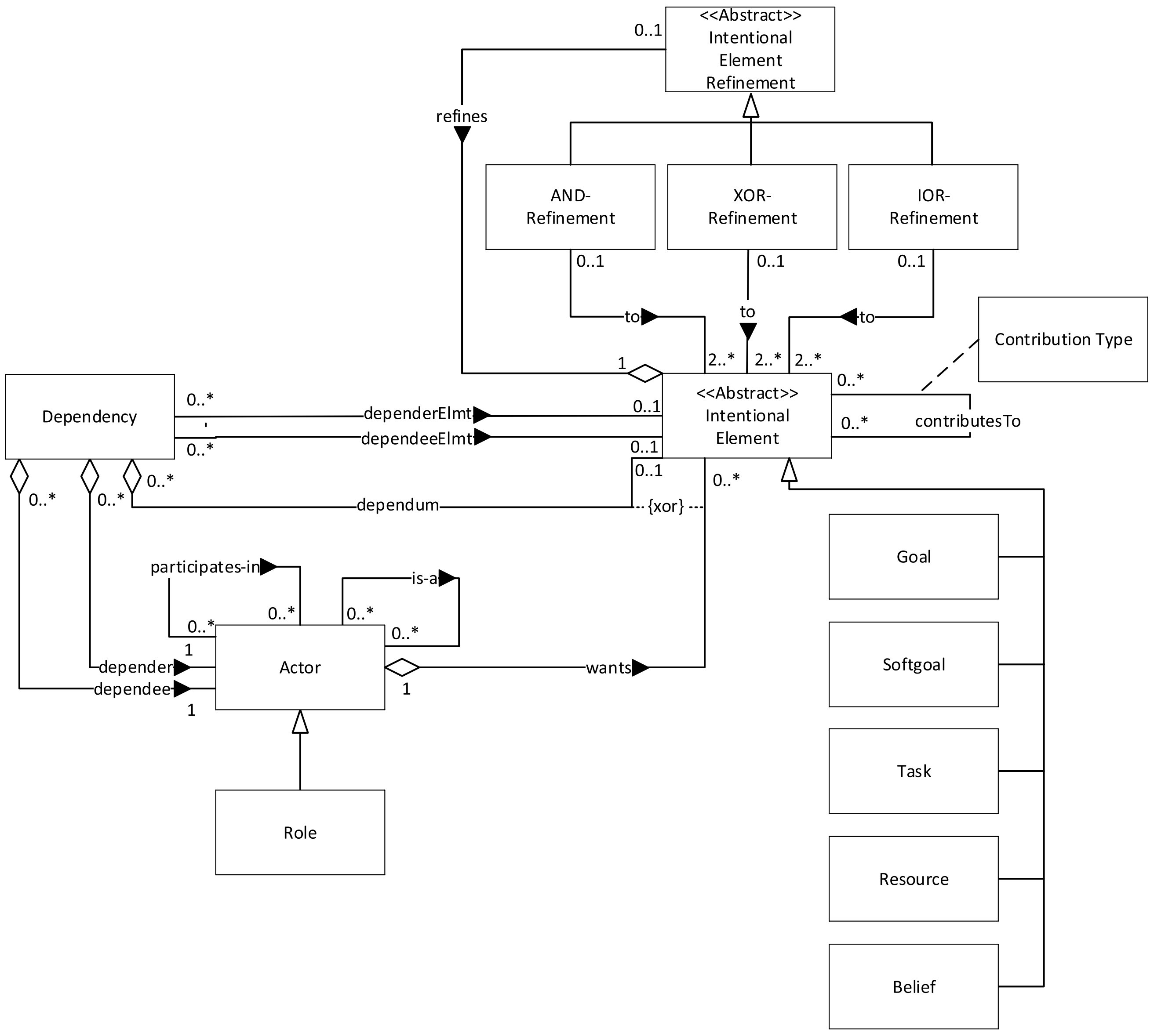


Fig. 4 GRL-compliant iStar metamodel

here as it will be further specialized in our extension for collaborative CPS. All changes made correspond with the GRL metamodel presented in Amyot et al. [32].

In the following, we briefly outline the most important entities and relationships defined by the metamodel. We detail the use of actors, intentional elements, and dependencies especially regarding their use for creating goal models for collaborative CPS networks.

4.1.1 Actors

Actors are commonly used to specify stakeholder intentions or define systems. In our case, we focus on the definition of systems. Actors are assigned intentional elements for which the actor strives to achieve fulfillment. An actor can be linked to another actor through an is-a-relationship or a participates-in-relationship. The is-a-relationship defines that some actor is of a certain type defined by the other actor. For instance, a CACC is also an ACC, thus, sharing parts of its intentional elements with a common ACC. The semantics of the participates-in relationship is defined by iStar 2.0; it depends on the type of actors between which the relationship is modeled. The participates-in relationship resembles a “plays” relationship when modeled between an agent as the source and a role as the target element, and “part-of” when it connects two actors of the same type.

In addition, an actor can be a role (i.e. role as a specialization of actor). This is a bit counterintuitive, as one would typically assume that an actor plays a role (e.g., the role of an intruder vehicle). However, we want to stay consistent to the iStar 2.0 metamodel and thus define role as a specialization of actor. This also has the benefit that for modeling the role the actor symbol can be used. In the CACC example roles a CACC takes in a platoon might be lead or following vehicle.

4.1.2 Intentional elements

An intentional element can be a goal (i.e. “a condition or state of affairs in the world that the stakeholders would like to achieve” [25]), softgoal (i.e. “a condition or state of affairs in the world that the actor would like to achieve, but [...] there are no clear-cut criteria for whether the condition is achieved” [25]), task (i.e. an intentional element that “specifies a particular way of doing something” [25]), resource (i.e. “a physical or informational entity” [25]), or belief (i.e. an intentional element that is “used to represent design rationale” [25]). A goal of a CACC might be to avoid collisions. Intentional elements can contribute to other intentional elements. This means that the fulfillment of one intentional element is supported, satisfied, hindered, or prevented by the fulfillment of another intentional element. For further possible contribution types between intentional elements, please refer to the ITU Recommendation Z.151 [25]. All intentional elements can be refined (often referred to as decomposed) into other intentional elements either through an AND- or an OR-refinement. The AND-refinement connects one intentional element with two or more sub-intentional elements, where the fulfillment of the intentional element depends on the fulfillment of all sub-intentional elements. For OR-refinements not all sub-intentional elements need to be fulfilled to achieve fulfillment of the super-intentional element. OR-refinements can be characterized either as IOR-refinement or as XOR-refinements. The IOR-refinement connects one intentional element with multiple sub-intentional elements, where at least one sub-intentional element needs to be fulfilled to guarantee fulfillment of the super-intentional element. The XOR-refinement connects one intentional element with multiple sub-intentional elements, where the fulfillment of the intentional element can be achieved by only one of the sub-intentional elements.

4.1.3 Dependencies

Dependencies describe the relationship between different actors and between intentional elements of different actors. A dependency defines that one actor is dependent on another actor with respect to fulfilling some of its intentional elements or that the fulfillment of one intentional element of one actor depends on another actor in general or a concrete fulfillment of one of its intentional elements. A dependency can exist between two actors, two intentional elements or combinations thereof. For example, two actors, an actor and a goal, or a goal and a task can be in a dependency relationship. The actor can take the position of a depender, who depends on another actor, for example, to perform a task or achieve a goal. The actor can also be the dependee, who provides the required resource or task execution. An intentional element involved in a dependency can be the depender element, the dependee element or the dependum. The dependum is an intentional element which is the object of the dependency. However, the use of a dependum is not mandatory in GRL (cf. [25]). For instance, when specifying a dependency between two actors, it might simply be unknown to the modeler. Therefore, we altered the multiplicities in so far, as we no longer expect each dependency to explicitly model a dependum which is not required by GRL.

4.2 Metamodel of the extension

To better support goal modeling for networks of collaborative CPS, we developed an iStar extension according to the requirements set out in Sect. 2.3. The metamodel for our extension is shown in Fig. 5. All changes done to the metamodel from Fig. 4 have been highlighted in grey.

We discuss the changes and their rationales again in the categories from Sect. 4.1, i.e. for actors (Sect. 4.2.1), for intentional elements (Sect. 4.2.2), and for dependencies (Sect. 4.2.3).

4.2.1 Actors

Most notable, we differentiate actors into collaborative CPS networks, collaborative CPS, and roles. Thus, we refine the agent concept of iStar 2.0, which covers concrete, tangible actors, into collaborative CPS and collaborative CPS networks. For a collaborative CPS network to be formed, at least two collaborative CPS need to exist and participate in such a network. For instance, in the example of the CACC, the platoon can be considered the collaborative CPS network and the individual CACCs participate in it. At least two vehicles equipped with CACCs are needed to form a platoon.

While we keep—compared to Fig. 4—the is-a relation between actors (although we can now state that the is-a relation is only acceptable between actors of the same kind), we can be now more restrictive regarding the participates-in relationship, because we consider very specific types of “agents”, as mentioned above. We split this dependency into two, more fine-grained relationships: The collaborates-in relationship and the is-assigned relationship. Thus, three kinds of actor relationships can be distinguished:

- Is-a relationship: An actor is of the type of another actor. For instance, a CACC is also an ACC. Note that in iStar and GRL it is prohibited to define that roles are agents

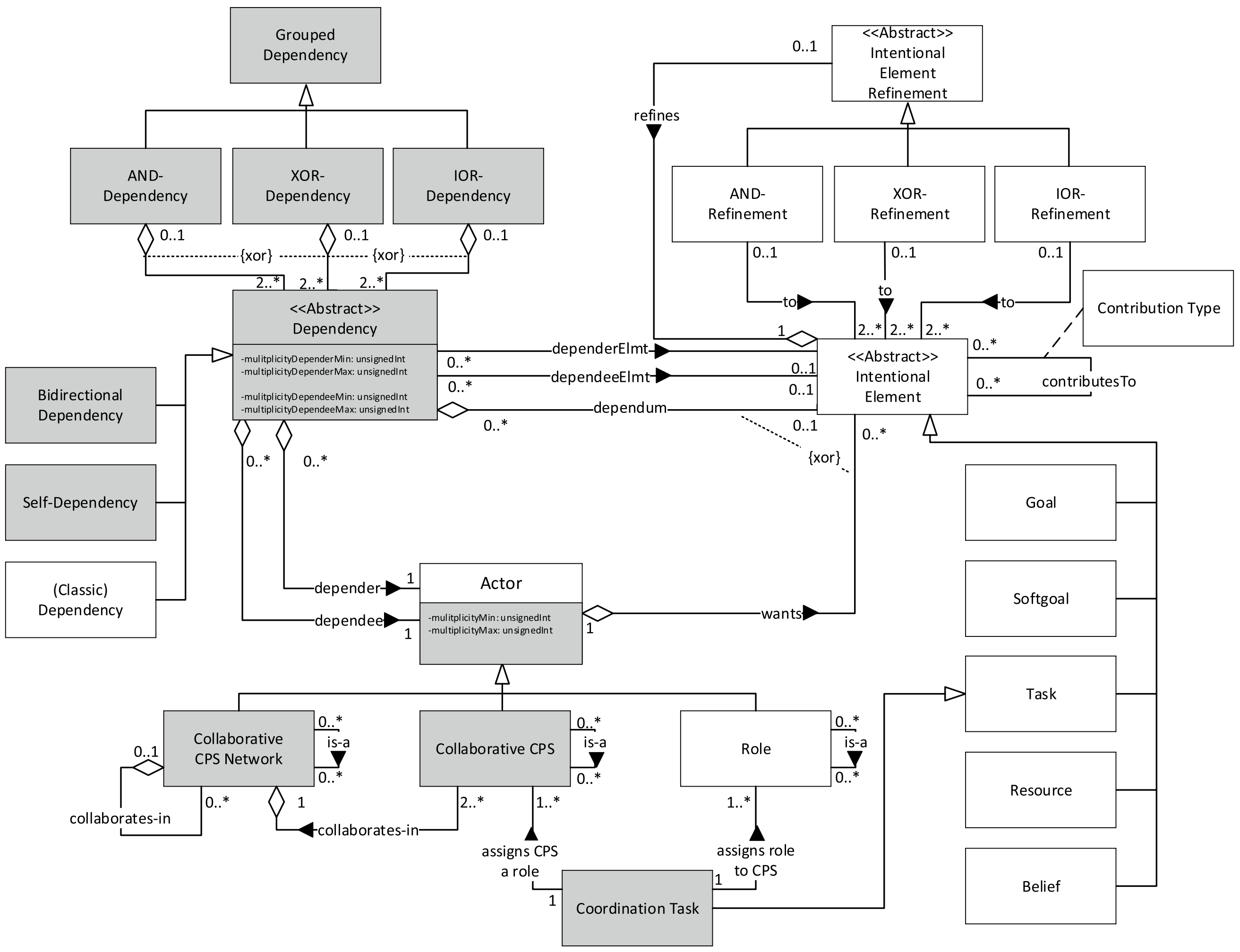


Fig. 5 Metamodel extension

and agents are roles. The same is valid here, is-a relationships only refine actors of the same type (i.e. CPS networks, CPS, and roles). For expressing the belonging of a CPS to a CPS network, the collaborates-in relationship is used, for expressing the assignment of roles to a CPS the is-assigned relationship is used.

- Collaborates-in relationship: A collaborative CPS collaborates with other collaborative CPS by partaking in a collaborative CPS network (which might be part of another greater CPS network). For instance, multiple CACCs cooperate to form a platoon (i.e. each CACC participates in the platoon). Thus, the collaborates-in-relationship denotes the membership of one actor in another actor. In particular, an individual collaborative CPS partakes in a network of collaborative CPS. In case of the CACC example, this means that a CACC participates in a platoon. A collaborative CPS network can also collaborate with other networks in some higher-level collaborative CPS network, such as a smart city. The collaborates-in relationship can be distinguished from the original participates-in relationship of iStar 2.0, because we restrict the use to CPS and CPS networks and exclude its use for role assignments. To assign roles, we define an is-assigned relationship to assign a role to a collaborative CPS. For example, a CACC in a platoon might be assigned the leader role.
- Is-assigned relationship: A collaborative CPS can be assigned a role within a collaborative CPS network. This subsumes two aspects, having a role and taking over a role. First, collaborative CPS can have roles. For instance, a CACC can participate in a platoon either as lead or as following vehicle. Second, roles in CPS networks need to be assigned, i.e. someone has to be responsible for assigning roles to collaborative CPS. This is expressed by a coordination task, which can belong to any actor (i.e. to a CPS network, a collaborative CPS, or a role). For instance, if the lead vehicle exits a platoon, its CACC is responsible for assigning another

CACC the role of the lead vehicle. The coordination task then defines which role is to be assigned to which CPS and, thus, which CPS is assigned which role. Thus, we stress the notion of active role assignment (and, possibly role change) in our extension, which is not specifically emphasized in iStar 2.0, where only "agent plays role" relationships are considered.

The use of multiplicities allows us to specify the goals of multiple configurations within one diagram. Therefore, actors can be assigned multiplicities. This allows us to represent actors of the same type (e.g., several identical CACCs) as one actor. In doing so, we can represent different but similar compositions of a CPS network in one goal model. For instance, we can use a goal model to represent platoons with three, four, five, etc., vehicles. However, it must be stressed that the actors that are subsumed by the use of multiplicities must be of the same type. For instance, an actor for following vehicles will only represent following vehicles that are equipped with a CACC.

This use of actor multiplicities facilitates the specification of CPS networks, as otherwise a multitude of different configurations would have to be specified. For instance, a platoon can consist of two following vehicles, three following vehicles, four following vehicles and so forth. To account for all these configurations, typically all of these must be explicitly specified. Thus, the use of multiplicities for actors is a way to facilitate specification (or considering the number of configurations to be considered) to make goal-based specification of CPS networks feasible. For analysis, however, each of the actors must be considered individually.

4.2.2 Intentional elements

To coordinate role assignment, we define a coordination task to be a specific kind of task that handles role assignment (i.e. allows collaborative systems changing their role or the role of another collaborative CPS). We adopted the idea of a coordination task from Teruel et al. [36]. A coordination task can belong to the collaborative CPS network, where, for instance, a platoon has a coordination task to choose a new leader in case the former leader leaves the platoon. This could be, for example, a voting mechanism where the platoon members collectively define which vehicle becomes the lead and which ones become followers. A collaborative CPS or a role that is assigned to a collaborative CPS can also be responsible for performing a coordination task. For instance, the platoon leader has the coordination task to assign other CACCs the role of a following vehicle when new vehicles join, or to exclude them from the platoon.

Other changes to the intentional elements have not been proposed. The assignment of intentional elements to either a network of collaborative CPS, a collaborative CPS, or a role is already given by the relation between actor and intentional element. Assigning intentional elements to a collaborative CPS does not mean that the collaborative CPS always aims at fulfilling all these intentional elements at the same time. The intentional elements of a collaborative CPS rather indicate which intentional elements can be fulfilled at some point in time. Considering that a collaborative CPS actor can represent multiple identical collaborative CPS, this means that identical but individual CPS can pursue different goals at the same time. For an example consider our CACC with two following vehicles. In addition to platooning relevant goals, each CACC has its own goals that are driver dependent and which might be conflicting. Take for instance, the goal to minimize fuel consumption and the goal to reach the destination as fast as possible. In the platoon the two following vehicles have in principle the same goals but the representation as one actor does not mean that both vehicles try to achieve the same goals as well. For instance, the driver of following vehicle 1 might prefer fast arrival, while the driver of following vehicle 2 aims for minimizing fuel consumption.

Assigning intentional elements to the collaborative CPS network actor means that these intentional elements cannot be assigned to an individual collaborative CPS but belong to the network. As the network consists only of collaborative CPS, it can be argued that each intentional element of a collaborative CPS or a role is also an intentional element of the network. While this is true, assigning intentional elements either to a collaborative CPS/role or the networks allows for distinguishing between those intentional elements that are under the control of the individual collaborative CPS and those that are not.

For Challenge 6, we propose the introduction of a new contribution type: configuration-dependent contribution value. The configuration-dependent configuration value indicates that the value of a contribution depends on specifics emerging from certain configuration aspects. As this concept is somewhat related to the unknown contribution value from GRL and iStar, we introduce a new label that is related to the unknown label. Other means of further defining this particular relationship with potentially changing values turned out to be too complex and unintuitive for it to be of use.

4.2.3 Dependencies

To reduce the size and complexity of the resulting goal model, we introduce further—more complex—dependency types that allow using fewer dependency links. Therefore, we define beside the classic dependency, bidirectional dependencies, self-dependencies, and grouped dependencies. In addition, we define multiplicities for dependencies.

A bidirectional dependency is a dependency, where both actors or their intentional elements depend on each other (e.g., task A from actor A depends on task B from actor B and task B from actor B depends on task A from actor A). This type of bidirectional dependency is quite often needed for collaborative CPS that are part of a CPS network. The CPS network has its own intentional elements. However, as the CPS network is no physical entity, the CPS network depends on the individual CPS in fulfilling these goals. Vice versa, the CPS join the network as this allows fulfilling goals they otherwise could not achieve. For a simple example, consider the goal of the platoon to reduce the driving time to the platoon's destination. To achieve this goal, the platoon depends on the individual CACCs' goals to reduce the driving time to their destination. Vice versa, the CACCs depend on the platoon as the platoon allows for a considerable reduction of driving time.

Our extension also includes a self-dependency. Self-dependencies are used to describe cases where one collaborative CPS relies on collaborative CPS of the same type (which is not the collaborative CPS itself) to fulfill the same goal, execute the same task, etc., for its own goal fulfillment, task execution, etc. For instance, to follow the leader of a platoon each vehicle (i.e. each CACC of each vehicle) in the platoon depends on other following vehicles to fulfill their tasks in following the respective vehicle ahead.

Furthermore, we now allow dependencies to be grouped. A grouped dependency subsumes several other dependencies. This allows building complex dependencies that include relations to multiple actors and/or their intentional elements. For instance, a network of collaborative CPS relies on the fulfillment of one of its tasks on all the participating collaborative CPS in fulfilling their tasks (e.g., to drive with constant speed the platoon relies on the individual CACCs to maintain the individual vehicles' speed).

As we allow multiplicities for actors to simplify the specification of multiple actors of the same type (e.g., multiple CACCs in the role following vehicle), we need to also consider multiplicities for dependencies that stretch between these actors. Thus, we can define that multiple dependers of the same type depend on multiple dependees of the same type. For instance, for coordinating the opening of a gap in a platoon, the CACC of the lead vehicle depends on the existence of at least two following vehicles.

4.3 Concrete syntax

4.3.1 Collaborative CPS

In iStar systems are represented as actors. A collaborative CPS is a system and is therefore modeled as an actor, as shown in Fig. 6. In addition, we use stereotypes to distinguish between the different types of actors, e.g. in Fig. 6 <<CPS>> defines that Actor A is a collaborative CPS and neither a role nor a CPS network. Inspired by Moody's

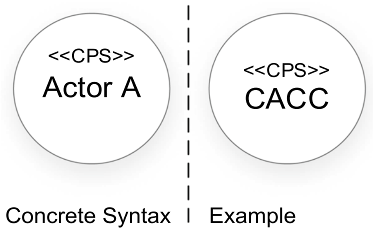


Fig. 6 Collaborative CPS actor

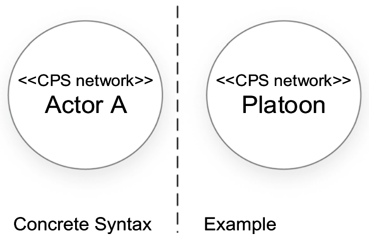


Fig. 7 Network of collaborative CPS actor

principles for constructing notations [47], the following notation follows the principle of semiotic clarity. According to Moody's principles, the same symbols should not be used for different concepts, otherwise a symbol overload may occur. However, we want to reduce the number of symbols and use the circle consistently for all actors and the stereotypes only for specialization. Here we follow the pattern of approaches from the related work that use symbols for denoting the supertype (i.e. the actor) and stereotypes for denoting its specializations (cf. [56, 93, 94]). Thus, since the notation for actor is specified by iStar, it is supplemented by stereotypes which serve to differentiate between network CPS and role.

4.3.2 Network of collaborative CPS

The CPS networks are also modeled as actors as shown in Fig. 7. Similar to the notation of collaborative CPS, the notation of the CPS network uses Moody's principle of semiotic clarity [47] by adding a stereotype referring to the CPS network to the existing notation.

To show that CPS belong to a CPS network, the CPS can be positioned in a CPS network. According to the principle of semantic transparency [47], which recommends the appearance of a notation should suggest its meaning, the CPS that is a part of a CPS network is displayed inside the CPS network actor. The idea of graphically nesting CPS actors inside of CPS network actors is inspired by the work of Guzman et al. [92] (see also Sect. 3.4). An example of our notation for nesting actors and thereby relating CPS to a CPS network is shown in Fig. 8. The CPS Actor B was modeled

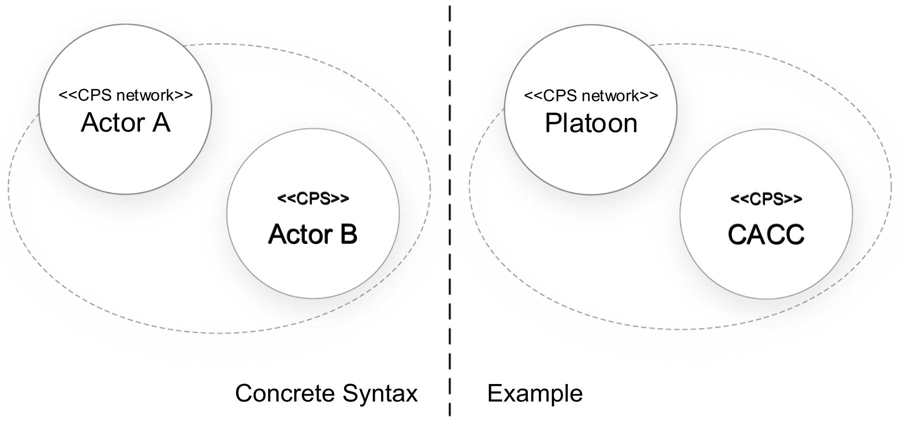


Fig. 8 Nested actors

inside the CPS network Actor A which indicates that Actor B participates in the CPS network Actor A.

The nesting of actors, i.e. placing a collaborative CPS inside a CPS network, has implications for the intentional elements of the actors:

- Intentional elements of the CPS network: For nested actors, each intentional element belongs to the actor it is directly placed in. This also means, that the intentional elements that are not within a CPS boundary but only in the CPS network boundary only belong to the CPS network. Intentional elements of the CPS network cannot be assigned to individual collaborative CPS. However, as the CPS network is no physical entity on its own but only consists of the physical CPS that form the CPS network, all these intentional elements depend on the CPS actors placed within the CPS network. To make this relation obvious we propose the use of dependency links to the respective intentional elements of the CPS they depend on.
- Intentional elements of the CPS: They do belong to the CPS and as the CPS belongs to the CPS network, they obviously are also part of the CPS network. However, they do not necessarily need to address purposes of the CPS network itself. For instance, a CACC has goals it tries to achieve when driving alone. When the CACC joins a platoon, it still has these goals, however, as the participation in a platoon allows fulfilling other CACC goals, the CACC will not try to achieve the original goals when in a platoon. Consequently, the corresponding intentional elements do not contribute to the platoon and are, thus, just part of the CPS but not intentional elements of the platoon.

Consequently, the nested representation of actors allows for illustrating the relations between intentional elements and actors while supporting the important distinction between intentional elements that can be assigned to individual systems and those that cannot.

4.3.3 Roles

iStar 2.0 [21] represents roles as shown in Fig. 9 (a). However, as we define a *participates-in* relation between actors to mean that a collaborative CPS participates in a CPS network, we use a *is assigned* relation to indicate that a collaborative CPS assumes a role (cf. Sect. 4.2.1). This is illustrated in Fig. 9 (b), where the collaborative CPS *Actor B* assumes the *Role C*. We allow for further simplification of the notation to depict the situation that a collaborative CPS assumes a role in one model element as shown in Fig. 9c. This allows reducing the size of models but prevents distinction between the intentional elements of a collaborative CPS and those of its roles. Thus, if the notation of Fig. 9 (c) is used, only the intentional elements belonging to the respective role shall be modeled. If a certain CPS can participate with different roles in the same CPS network, the CPS needs to be modeled multiple times as different actors with different roles. If the actor notation is used without definition of a role, only intentional elements belonging to the actor in any role should be modeled.

4.3.4 Coordination task

In Fig. 10 the concrete syntax for a coordination task is shown. Again, we use the well-known symbol of a task and use stereotyping to denote the difference. This is in accordance with Moody’s principles [47], to allow users easy identification of the overall concept (i.e. task). In addition, an assignment relation shows which actor (i.e. which type of collaborative CPS) is assigned to which role.

4.3.5 Bidirectional dependency

The bidirectional dependency represents a dependency in both directions between two actors or intentional elements. The direction of a regular dependency is represented by the "D". Since we have a dependency in both directions, we use the “D” in both directions as this is intuitive according to

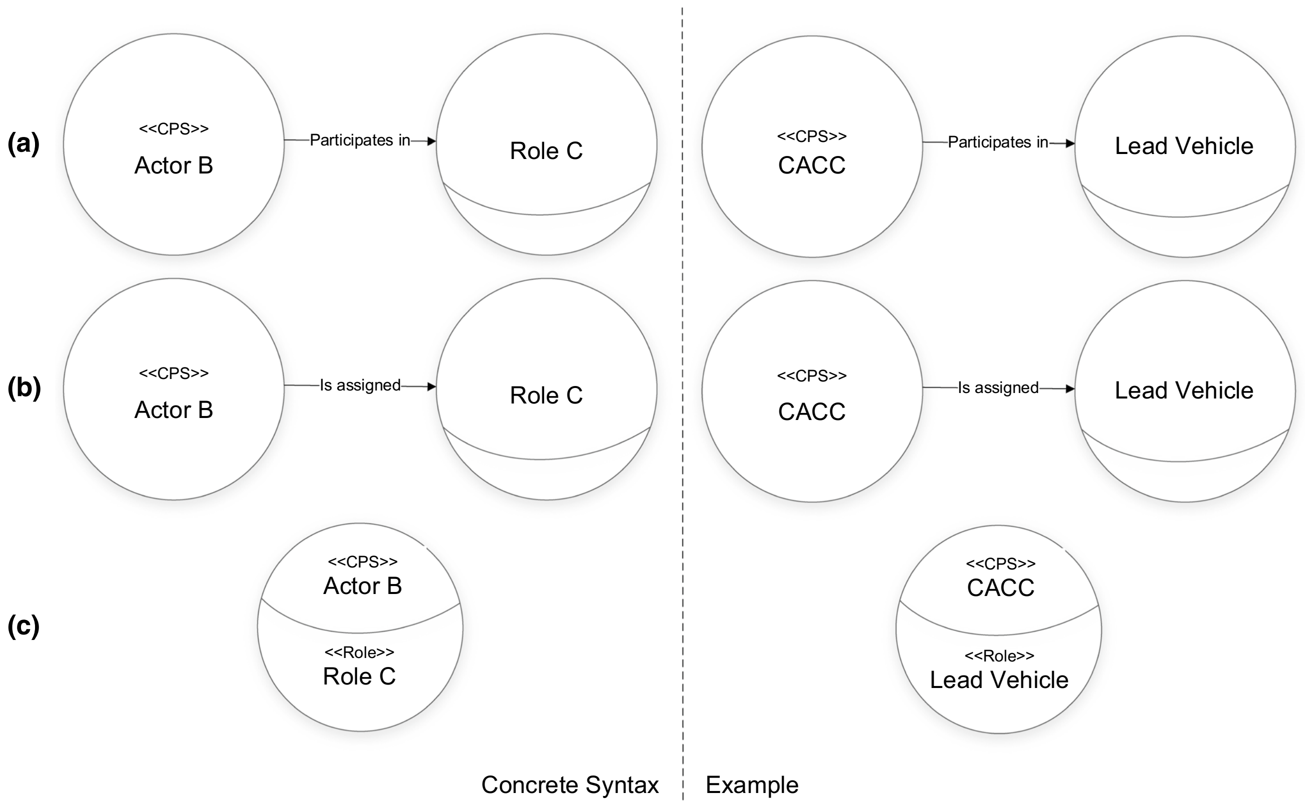


Fig. 9 Roles

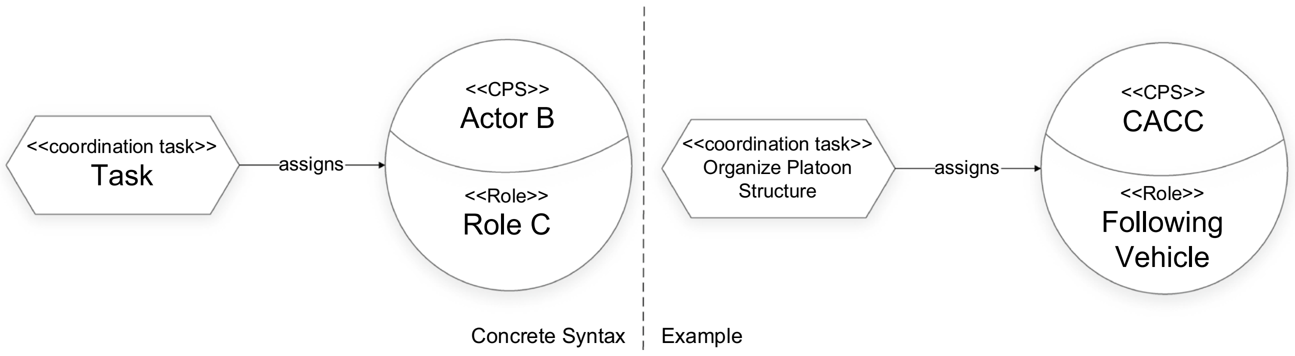


Fig. 10 Coordination task

the principles of Moody [47]. This is illustrated in Fig. 11. Like regular dependencies, bidirectional dependencies can be defined including or excluding a dependum.

4.3.6 Self-dependency

In a network of collaborative CPS there can be systems that have the same role, the same goals, the same tasks, etc. For simplification we represent all these systems using only one actor in the goal model, which avoids redundancy. As we can represent more than one system as one actor, several peculiarities can occur, such as dependencies between the goals of these systems (i.e. a system of a certain type or role depends on another system of the same type or role). As a consequence, we allow for defining dependencies within one actor. To indicate the different nature of such as dependency (i.e. to indicate that the system does not depend on itself but on the systems of the same type or role), we define

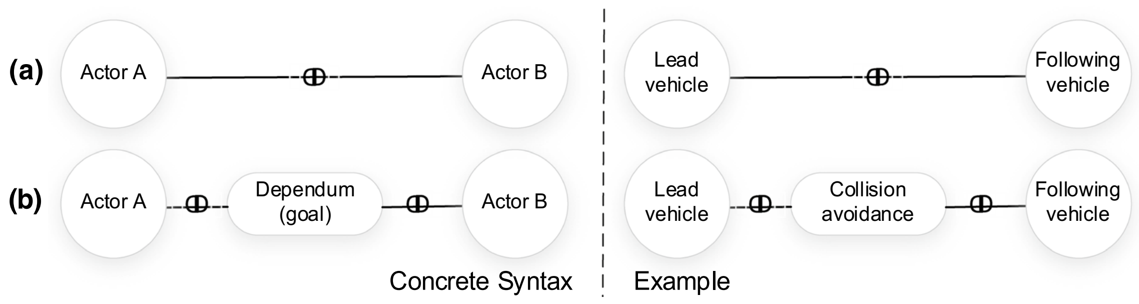


Fig. 11 Bidirectional dependency

Fig. 12 Self-dependency between different tasks

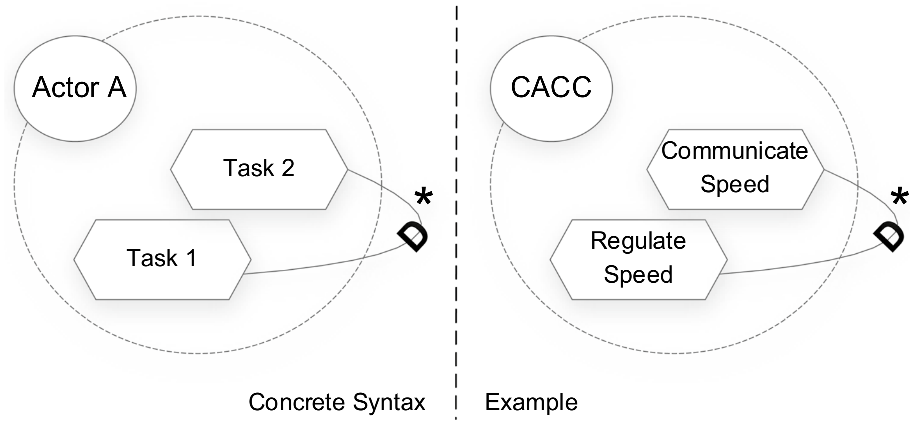
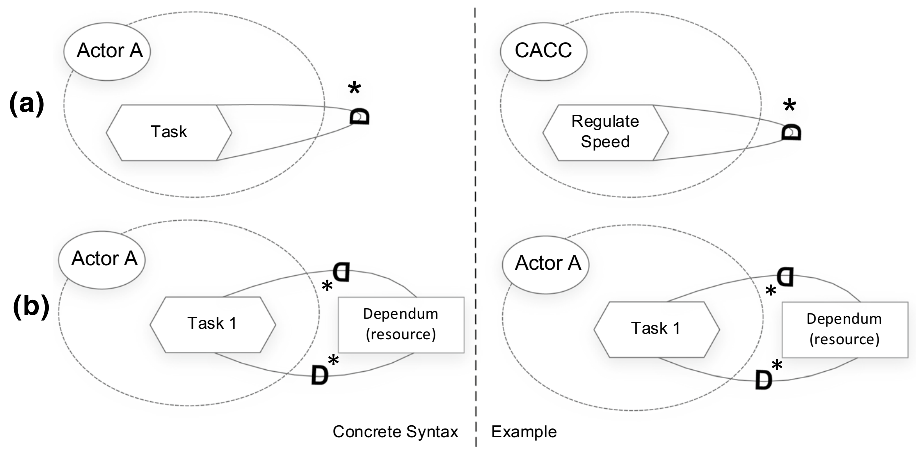


Fig. 13 **a** Self-dependency without a dependum and **b** self-dependency with a dependum



a self-dependency as a new construct. This dependency is represented as shown in Fig. 12 by a D with a * operator as it is a well-known symbol for self-properties [97]. The D* symbol needs to be placed outside the actor boundary so as to avoid misinterpreting this element for a dependency within the same instance of an actor.

A self-dependency can exist between different intentional elements, but it can also exist for a single intentional element and a dependum as is shown in Fig. 13.

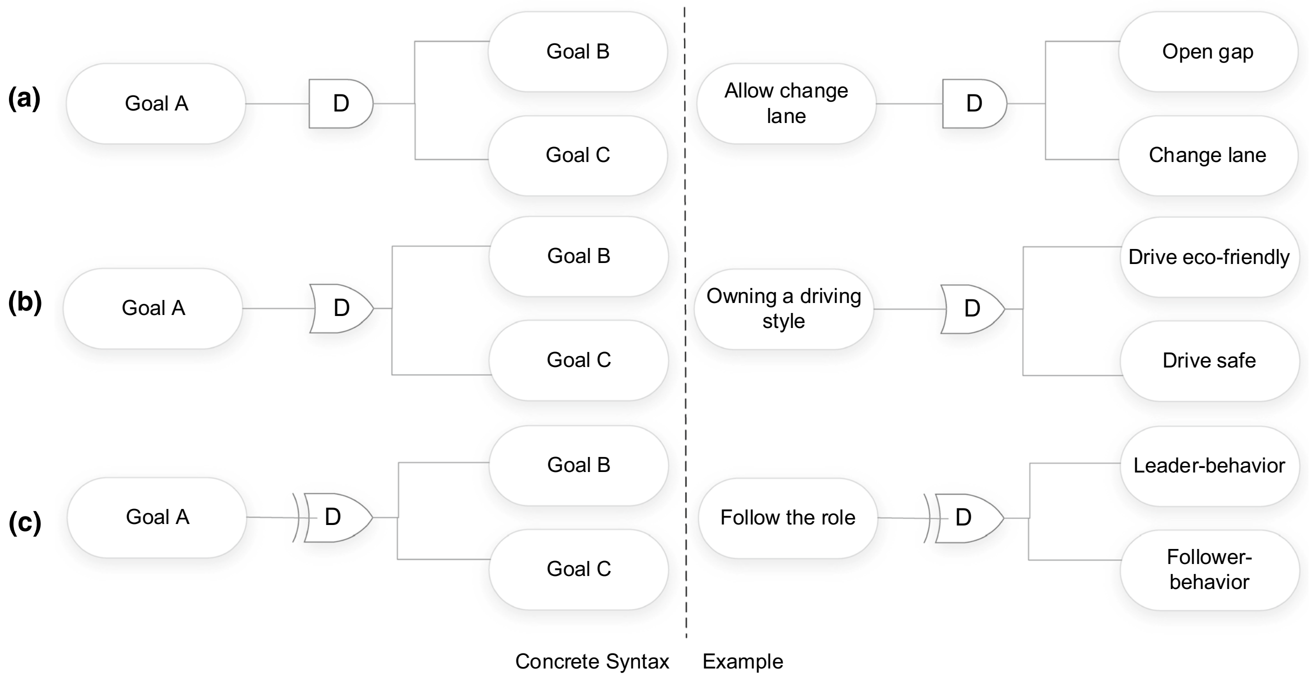


Fig. 14 **a** AND-dependency, **b** IOR-dependency and **c** XOR-dependency

Note that traditionally the use of dependencies between intentional elements of one actor is discouraged.[5] Here, we explicitly define a special kind of dependency to be used between intentional elements that are modeled within the same actor boundary. However, recall that we represent multiple actors by displaying just one actor to facilitate the specification of CPS networks consisting of multiple different but identical actors. Therefore, this self-dependency does not link intentional elements of the same actor but of different actors of the same type and is thus in line with the common usage of dependencies.

4.3.7 Grouped dependency

The concrete syntax for grouped dependencies is similar to logical gates as these provide symbols for AND, IOR, and XOR. The AND-dependency is shown in Fig. 14a. Goal A depends on both, Goal B and Goal C. An IOR-dependency is shown in Fig. 14b Goal A depends on Goal B or Goals C. The XOR-dependency is shown in Fig. 14c. The XOR-dependency shows a dependency, where Goal A can depend on either Goal B or Goal C but cannot depend on both. We use symbols well known from logic gates for conjunctions and disjunctions and combine them with the iStar symbol *D* used for dependencies.

4.3.8 Multiplicities

Multiplicities can be assigned to actors and dependencies. The use of multiplicities for actors is shown in Fig. 15. For multiplicities we use the well-known notation for multiplicities in the UML. Using multiplicities allows to state that a certain type of actor is involved multiple times in the same goal model. For instance, a platoon consists of multiple CACCs. In Fig. 15, this is specified by using a [1 … *n*] multiplicity, showing that at least one CACC must exist to form a platoon and the upper bound is unlimited. Note that there exist different assumptions on the relation between CACCs and platoons, while some developers might

[5] For GRL, recommendation Z.151 [25] defines in its abstract grammar a dependency as specialization of *ElementLink* which links *GRLLinkableElements* (i.e. actors and intentional elements). Each *ElementLink* has a source and a destination. It is not explicitly defined that source and destination cannot be identical. However, the detailed guidelines for the use of dependencies illustrate six common usage scenarios that are explicitly suggested. All of these introduce dependencies between different actors, or intentional elements of different actors, or between a combination thereof. Thus, it can be assumed that the use of dependencies between intentional elements of the same actor is not intended.
For iStar, Dalpiaz et al. define that "the depender and dependee of a dependency should be different actors" [21].

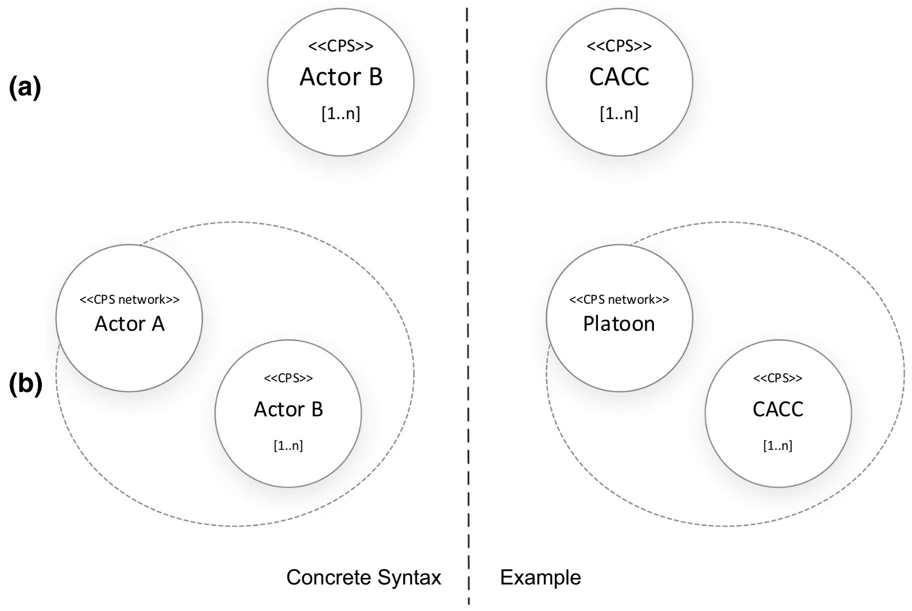


Fig. 15 Multiplicities, in **a** a single actor in **b** a nested actor

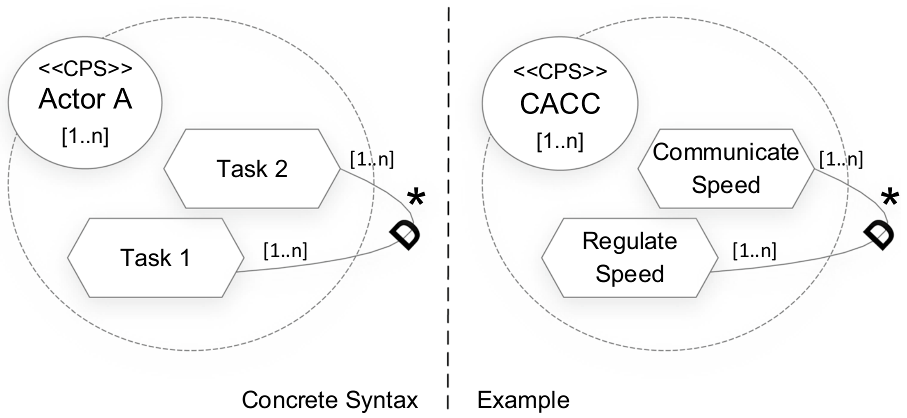


Fig. 16 Self-dependency with multiplicities

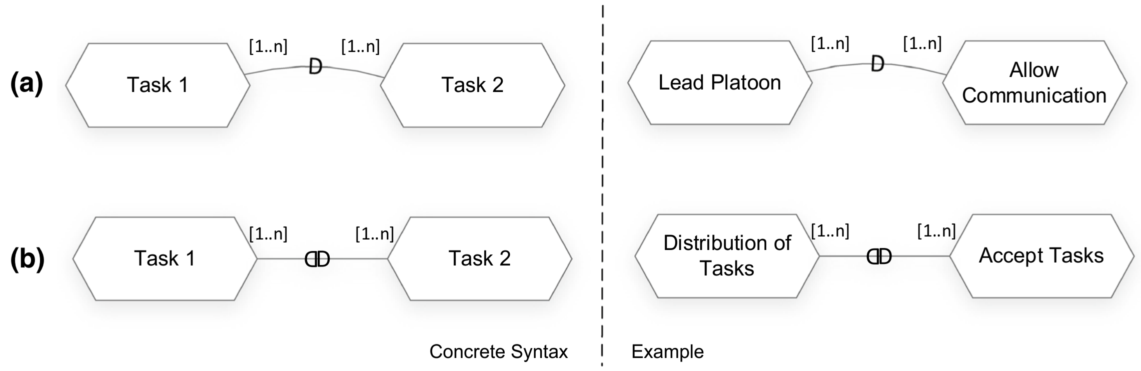


Fig. 17 **a** One-directional dependency and **b** bidirectional dependency with multiplicities

want to consider open ended platoons, others might rather want to work with realistic upper bounds as the platoon is typically limited in its length by regulations. In addition, it is a rather philosophical question, whether a CACC on its own can be a platoon. Thus, also a multiplicity of [2…8] might be a valid assumption, depending on the current development project.

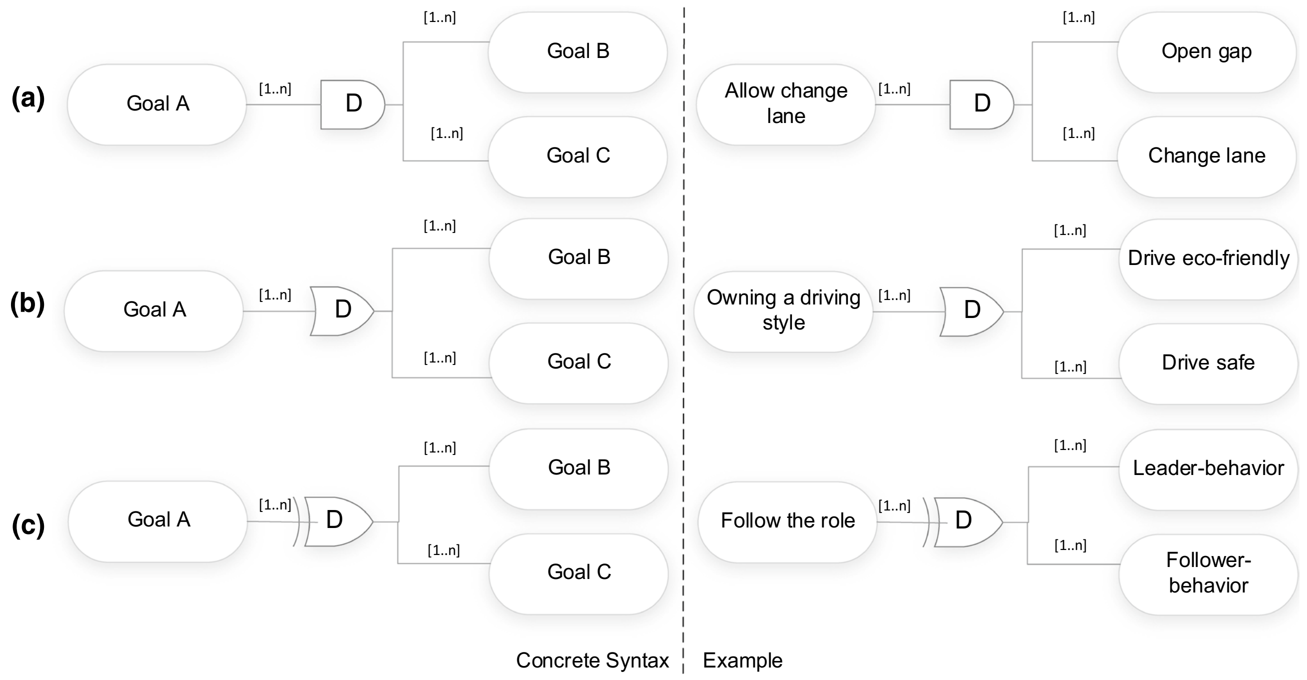


Fig. 18 **a** AND-dependency **b** IOR-dependency and **c** XOR-dependency (**c**) with multiplicities

Multiplicities can also be assigned to the intentional elements involved in a dependency. The depender, dependee, and dependum can be assigned a multiplicity regardless of the kind of dependency (see Figs. 16, 17 and 18). We do not restrict multiplicities, but we never came across a need for a $[0\ldots n]$ dependency, as this would mean that the depender does not necessarily depend on a dependee. Note that in case of self-dependencies, it is necessary that the multiplicities allow for there to be more than one actor, as self-dependencies do not define dependencies within the same actor but between actors of the same type.

Note that in principle iStar and GRL are already equipped with the potential to define dependency decompositions. However, this always requires the existence of a decomposition between the depender and dependee elements. A brief example is given by Fig. 19, which highlights the usefulness of our extension that limits the complexity of the model and allows for a different use of grouped dependencies. In particular, our proposed grouped dependency allows expressing that an intentional element depends on multiple other intentional elements which do not need to be related to each other.

4.3.9 Configuration-dependent contribution value

To express that the value of a contribution depends on a configuration or on certain aspects related to multiple configurations, we define a new label for contributions. This contribution is closely related to the unknown contribution value relation, where it is also not obvious whether the contribution is positive or negative. However, the difference between unknown contributions and configuration-dependent contributions is that for the latter, we can know how the contribution impacts but the contribution values are usually too complex to define all possibilities in graphical model. However, in addition we propose the formal definition of the contribution dependence in a comment field or a separate referenced document. Thus, we provide an index that can be used for reference. The proposed label compared to the

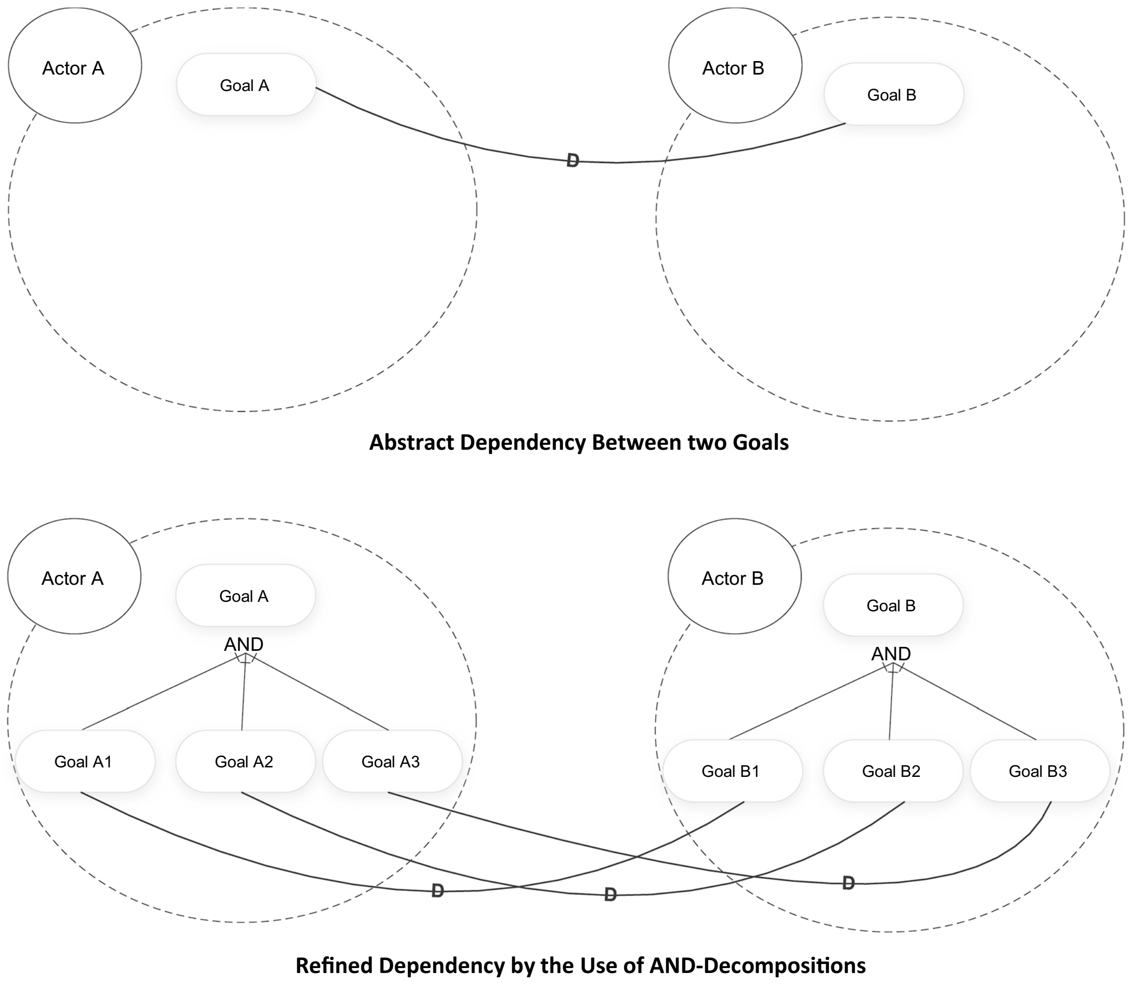
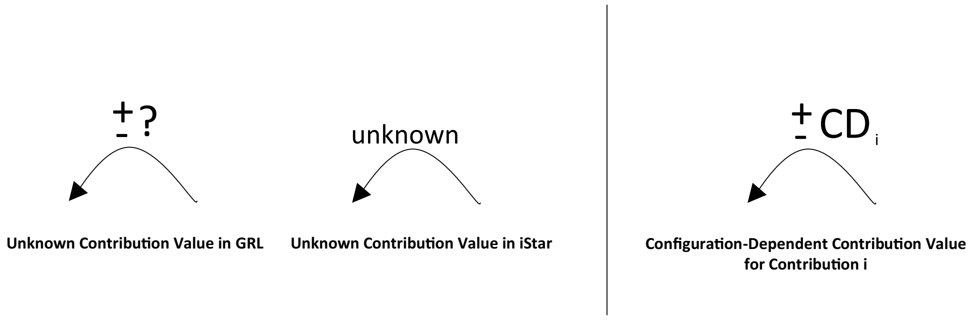


Fig. 19 Traditional dependency refinement

Fig. 20 Configuration-dependent contribution value


±?
unknown
±CD_i
Unknown Contribution Value in GRL
Unknown Contribution Value in iStar
Configuration-Dependent Contribution Value for Contribution i

labels for unknown contribution values from GRL and iStar can be found in Fig. 20.

4.4 Well-formedness rules

We define well-formedness rules using OCL [98] for goal models according to the proposed extension to support the creation of correct goal models for collaborative CPS as is recommended for defining modeling languages or extensions to modeling languages [99].

A self-dependency is defined as a dependency between actors of the same type, which are represented in a goal model by one actor. Therefore, the depender and the dependee must be the same actor (Well-formedness rule 1) and this actor must have a maximum multiplicity of more than one (Well-formedness rule 2), as otherwise there could not be more than one actor of this type in a CPS network.

Well-formedness rule 1 Depender and dependee of a self-dependency must be the same actor or intentional elements that belong to the same actor.

```
context Self-Dependency
inv: self.depender = self.dependee
```

Well-formedness rule 2 The actor a self-dependency belongs to (i.e. the depender and dependee or the actor the depender and dependee elements belong to) must have a maximum multiplicity larger than one. This is important as the self-dependency is not a dependency between elements of the same actor but between elements of actors of the same type that are just represented by one single actor element.

```
context Self-Dependency
inv: self.depender.multiplicityMax > 1
```

To prevent inconsistencies regarding multiplicities, we stipulate that the following rules must be adhered to. The minimum multiplicity of an actor may not be larger than the maximum multiplicity of the same actor. The same holds for multiplicities related to dependencies (i.e. the multiplicities of the depender, the dependee, and the dependum).

Well-formedness rule 3: A maximum multiplicity may not be smaller than the corresponding minimum multiplicity.

```
context Actor inv: self.multiplicityMin <=
self.multiplicityMax

context Dependency
inv: self.multiplicityDepdenderMin <=
self.multiplicityDependerMax

 context Dependency
inv: self.multiplicityDepdendeeMin <=
self.multiplicityDependeeMax
```

The minimum multiplicity of a dependency's dependee may not be smaller than the minimum multiplicity of the dependee actor. This prevents cases where the dependency would allow for requiring a smaller number of collaborative CPS than are actually permissible according to the actor's multiplicity. The same holds for the depender.

Well-formedness rule 4: The minimum multiplicity of a dependee/depender may not be smaller than the minimum multiplicity of the dependee/depender actor.

```
context Dependency
inv: self.multiplicityDependeeMin >=
self.Dependee.multiplicityMin

context Dependency
inv: self.multiplicityDependerMin >=
self.Depender.multiplicityMin
```

The maximum multiplicity of a dependency's dependee may not be larger than the maximum multiplicity of the dependee actor. This prevents cases where the dependency would allow for requiring a higher number of collaborative CPS than are actually permissible according to the actor's multiplicity. The same holds for the depender.

Well-formedness rule 5 The maximum multiplicity of a dependee/depender may not be larger than the maximum multiplicity of the dependee/depender actor.

```
context Dependency
inv: self.multiplicityDependeeMax <=
self.Dependee.multiplicityMax

context Dependency
inv: self.multiplicityDependerMax <=
self.Depender.multiplicityMax
```

Well-formedness rule 6 Grouped dependencies must have either the same depender or the same dependee. This rule prevents the definition of complex dependency relationships that are difficult to comprehend which carry a high risk of misinterpretation.

```
context Dependency
inv: self.GroupedDependency.Dependency->
((forAll(d|d.dependerElmt=self.
dependerElmt) or (forAll(d|d.dependeeElmt=
self. dependeeElmt)))
```

4.5 Tool support

We provide tool support for the extension as a Visio stencil. Microsoft Visio is a commonly used modeling tool that provides mechanisms for the definition of modeling languages. In our case, the decision to use Microsoft Visio was made due to its availability for industry partners from different domains. In addition, particularly in the industry automation domain it is very heavily used for the design of production systems. Figure 21 shows the stencil and the shapes it defines. We provide shapes for the newly defined constructs as well as for the existing constructs. The shapes can be drag-and-dropped to the drawing to create a goal model for collaborative CPS according to the proposed extension. The stencils are available for download at https://doi.org/10.6084/m9.figshare.13313093. While Visio primarily focuses on providing support for modeling, add-ins can be created to support model validation, such as checks for violation of syntactic or well-formedness rules. Implementing these checks as well as support for goal fulfillment analysis is part of future work.

5 Evaluation

We evaluated the proposed extension by conducting two case studies in different industry domains. Section 5.1 elaborates on the case study research design chosen (cf. [100]). Sections 5.2 to 5.4 present the results. Subsequently Sect. 6 will discuss the findings and limitations of the evaluation.

5.1 Study design

5.1.1 Goals

The study aims at evaluating the proposed extension for goal modeling of collaborative CPS. Therefore, we applied the iStar goal modeling extension to two industrial case studies (i.e. a cooperative adaptive cruise control and a fleet of collaborative transport robots). Thereby, we evaluate the applicability of the approach as well as the benefits of each introduced modeling element.

5.1.2 Research questions

To achieve the overall goal of the study, i.e. does the proposed extension aid in goal modeling for collaborative CPS, we defined several research questions to be answered in the study:

- *RQ1:* Is the proposed iStar extension applicable to industrial case examples of collaborative CPS?
- *RQ2:* Does the use of the proposed iStar extension lead to more concise models?
- *RQ3:* Are the proposed modeling elements useful in the context of modeling collaborative CPS?
- *RQ4:* What challenges remain?

We further refine RQ1 and RQ2 with regard to the two industrial case examples:

- *RQ1.1:* Is the proposed iStar extension applicable to model a cooperative adaptive cruise control?
- *RQ1.2:* Is the proposed iStar extension applicable to model collaborative transport robots?

For RQ 2, we need to define the meaning of concise. Concise means "marked by brevity of expression or statement: free from all elaboration and superfluous detail."[6] With regard to goal models we refer to a goal model as more concise if it has fewer elements than another goal model that expresses the same content.

- *RQ2.1:* Does the use of the proposed iStar extension lead to a more concise yet still comprehensible model of the cooperative adaptive cruise control?
- *RQ2.2:* Does the use of the proposed iStar extension lead to a more concise yet still comprehensible model of the collaborative transport robots?

For RQ3, we need to define the metrics for usefulness. Usefulness can be defined as "the quality of having utility and especially practical worth or applicability."[7] Thus, additionally to the investigation of the general applicability of the iStar extension (see RQ1), we investigate the applicability of each modeling element. Furthermore, it is investigated whether industry partners deem the modeling element useful (i.e. is it worth to have the modeling element as part of the iStar extension).

[6] cf. https://www.merriam-webster.com/dictionary/concise.

[7] cf. https://www.merriam-webster.com/dictionary/usefulness.

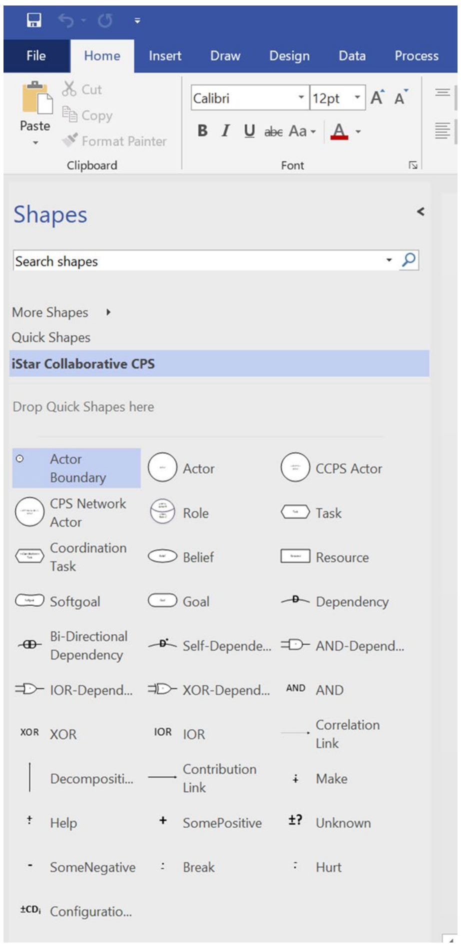


Fig. 21 Visio stencil for the extension

In addition, we need to further refine this research question for all proposed modeling elements, i.e.:

- *RQ3.1:* Is the use of collaborative CPS and the network of collaborative CPS as actors useful?
- *RQ3.2:* Is the use of the coordination task useful?
- *RQ3.3:* Is the use of bidirectional dependencies useful?
- *RQ3.4:* Is the use of self-dependencies useful?
- *RQ3.5:* Is the use of grouped dependencies useful?
- *RQ3.6:* Is the use of multiplicities for dependencies useful?

For RQ4, we separate between limitations of the proposed iStar extension and the resulting needs to be coped with in future work, i.e.:

- *RQ4.1:* What are limitations of the proposed iStar extension?
- *RQ4.2:* What are industry's needs for future work?

5.1.3 Subject selection

Industry partners and case examples were recruited within the CrESt-project, a joint research project, publicly funded by the German Federal Ministry of Education. The project, aiming at developing engineering methods for model-based software engineering of collaborative CPS, started in February 2017 and concluded in April 2020. Industry partners contributed four case example specifications. For the application of our extension we chose two case examples. The decision was made based upon interest of involved industry partners (i.e. the involved partners were highly interested in applying goal modeling techniques to investigate their case). While each case example was mainly driven by one responsible industry partner, other partners from the respective domain were also involved and contributed to the case study. Industry partners thus participated and contributed due to their commitment to the project but also due to their interest in the case and the definition and evaluation of solutions that foster the model-based engineering of collaborative CPS. While research in the project was partly conducted in close collaboration and resulted in co-authored publications (e.g., [35, 101, 102]), no further interdependence of interests exists between the authors of this paper and the involved industry partners.

The automotive case example of cooperative adaptive cruise control systems was provided by a large automotive supplier located in Germany. In addition, other suppliers—including one of the world largest automotive suppliers—and original equipment manufacturers (all based in Germany) were involved in the case example. The transport robot case example was provided by a medium-sized Germany-based internationally operating company specialized in the production of autonomous transport robots. In addition, a very large international company with headquarters in Germany and multiple interests as well as a broad portfolio of products and domains that has a major interest in the domain of industry automation was involved.

5.1.4 Procedure

During the case study the approach under investigation (i.e. the iStar extension for collaborative CPS) was applied to two case examples provided by industry partners. Therefore, the following procedure was adhered to, to allow answering Research Questions 1–4.

Over the course of 3 years, we conducted a total of twelve workshops, one workshop every 3 months. Each workshop lasted about 2 days. The workshops were closely integrated in the working structure of the surrounding CrESt project. Therefore, they were not exclusively used for discussing the iStar extension but also for other research related to collaborative CPS. We did not use fixed time slots so that it was always possible to have as much discussions as needed. The workshops were attended by about a dozen people, among them employees from various companies and research institutions. First, we were provided with a brief specification and description of the case examples. In workshops the details of the case example were discussed and answers regarding specific aspects that remained unclear from the description were given by industry partners. Subsequently, initial sketches for the goal models were made. At first this was done without using the extension to get an understanding of the shortcomings of iStar/GRL with respect to goal modeling for collaborative CPS, for a report on the findings of this phase, please refer to our previous work [23]. Sketches were handed in for critique and revised based on the feedback. After additional workshops, the goal models without the extension were finalized and a number of shortcomings and potential solution concepts were discussed with the industry partners.

Next, we created goal models using the proposed extension for goal modeling of collaborative CPS. To allow for comparability, we started with the agreed upon goal models without the extension and made changes according to the extension. After another round of critique and a final workshop the final versions were created. In addition to the workshops, regular biweekly web conferences presented the opportunity to discuss upcoming questions in a timely manner. Furthermore, industry partners provided in-depth feedback on the extensions and derived models via mail.

Thus, information was collected during workshops and web conferences, as well as from documents. These documents included case descriptions of the case studies, requirements for modeling approaches for networks of collaborative CPS, and goal models in various stages of completion. These documents were created in close collaboration between domain experts and goal modeling experts under the auspice of the respective experts. Thus, we did not use specific questionnaires to answer the research questions but used an open and exploratory approach. We took notes on the meetings and the documents and models created over the course of the project were iterated regularly. In addition, written feedback was also received from industry partners. The workshops were used to discuss the case examples, the requirements, and the goal models. These discussions involved clarification of misunderstandings, detailed discussions of interesting aspects, discussions of created goal models and the proposed extension. These discussions were documented during the workshop. The notes taken during the workshops served as input for the proposed extension as well as the evaluation results. This allowed everyone involved to provide input according to their opinions. However, participants were also free to keep opinions to themselves.

Results from the application (RQ1) and the impact on the resulting models (RQ2) can be found in Sect. 5.2. To answer RQ3 (see Sect. 5.3) we discussed the proposed modeling elements of the approach with our industry partners to ensure that these are adequately reflecting the respective complex situations, are not misunderstood, and are deemed supportive. For RQ4, we discussed remaining challenges with our industry partners, particularly with respect to the severity of the various needs.

5.1.5 Case examples

To show the benefits of the proposed extension, we conducted two case studies, one in the automotive and one in the industry automation domain. Section 5.1.5.1 introduces the cooperative adaptive cruise control case example and Sect. 5.1.5.2 the collaborative transport robots.

5.1.5.1 Cooperative adaptive cruise control Cooperative adaptive cruise control (CACC) systems allow vehicles to form a platoon [95]. A platoon is a network of vehicles driving behind one another with small distances between them. A platoon consists of a lead vehicle and at least one following vehicle. The lead vehicle is the first vehicle of the platoon and thus bears the responsibility for the platoon, since it has to decide, for example, which maneuvers to execute. All other platoon vehicles are following vehicles, as they usually adopt the driving style of the preceding vehicle and reproduce it. Platooning offers many advantages, as the reduced distance between the vehicles allows driving in the slipstream of the previous vehicle. As a result, the following vehicles consume less fuel. Furthermore, platooning can reduce congestion on streets, is safer, and provides more comfort to drivers [96]. Having a CACC allows vehicles to participate in a platoon, as it enables the vehicles to communicate with each other within a platoon. With this communication, vehicles can agree on a common speed, a common destination, or a common driving style.

5.1.5.2 Collaborative transport robots Collaborative transport robots are tasked with transporting materials and products between machines and conveyor belts and with disposing of material that is no longer used. They can do so without getting in each other's way and more efficiently by forming a fleet [103]. In order to form a fleet, the robots need to communicate with each other about their current positions, tasks, battery statuses, etc. Forming a fleet allows the individual robots to be better utilized, as the individual tasks can be divided evenly between the robots. There are further advantages to having transport robots collaborate as a fleet rather than individually. For example, if there is an obstacle in a route, it is included in the map so that all robots know that the route cannot be taken and that they have to find an alternative route [104]. In addition, the robots can automatically visit a charging station if the remaining bat-

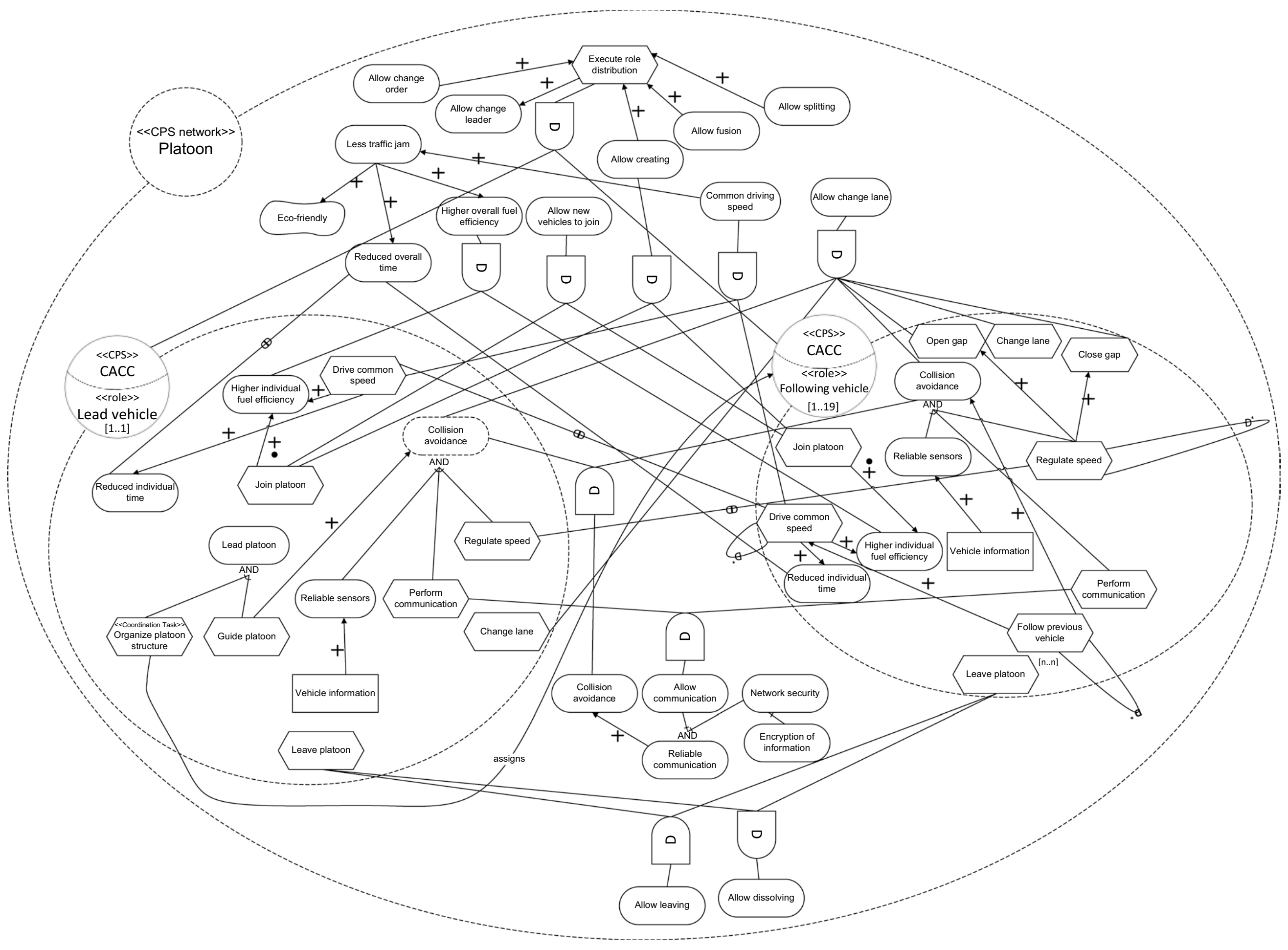


Fig. 22 Goal model cooperative adaptive cruise control platoon with extension

tery is at a previously set remaining level. If the workload is high and a large number of transport tasks have to be executed, the value of the battery life, at which the robot is to visit a station, can be set to a lower value so that it can still complete as many transport tasks as possible. Furthermore, since there are different types of transport robots that are used for different products and materials, for example because they differ in their load capacity, the fleet can take this into account when distributing transportation orders.

5.2 Application results

This section introduces the application of the approach in the context of the case studies. Therefore, Sect. 5.2.1 shows the goal model of the cooperative adaptive cruise control case example, Sect. 5.2.2 shows the goal model of the collaborative transport case example. For comparison in both sections goal models for case examples are shown with the proposed extension and without the proposed extension.

5.2.1 Application to the cooperative adaptive cruise control case example (RQ1.1)

We applied the iStar extension to the case example of a cooperative adaptive cruise control from the automotive domain. Figure 22 shows the resulting goal model. As can be seen, the CACC itself is not directly represented by an actor. A platoon, a lead vehicle, and a following vehicle are depicted as actors. The platoon represents the network which is formed by the collaboration of multiple vehicles equipped with a CACC. A CACC takes part in only one of two possible roles in a platoon, one CACC is the lead vehicle, the other CACCs have the role following vehicle. Hence, the three actors shown represent the roles a CACC can take and the collaborative network a CACC takes part in.

Each actor has its own intentional elements. However, as can be seen, the actors, and thus their intentional elements, heavily rely on each other. Particularly, the platoon's intentional elements depend on goals and tasks of the vehicles.

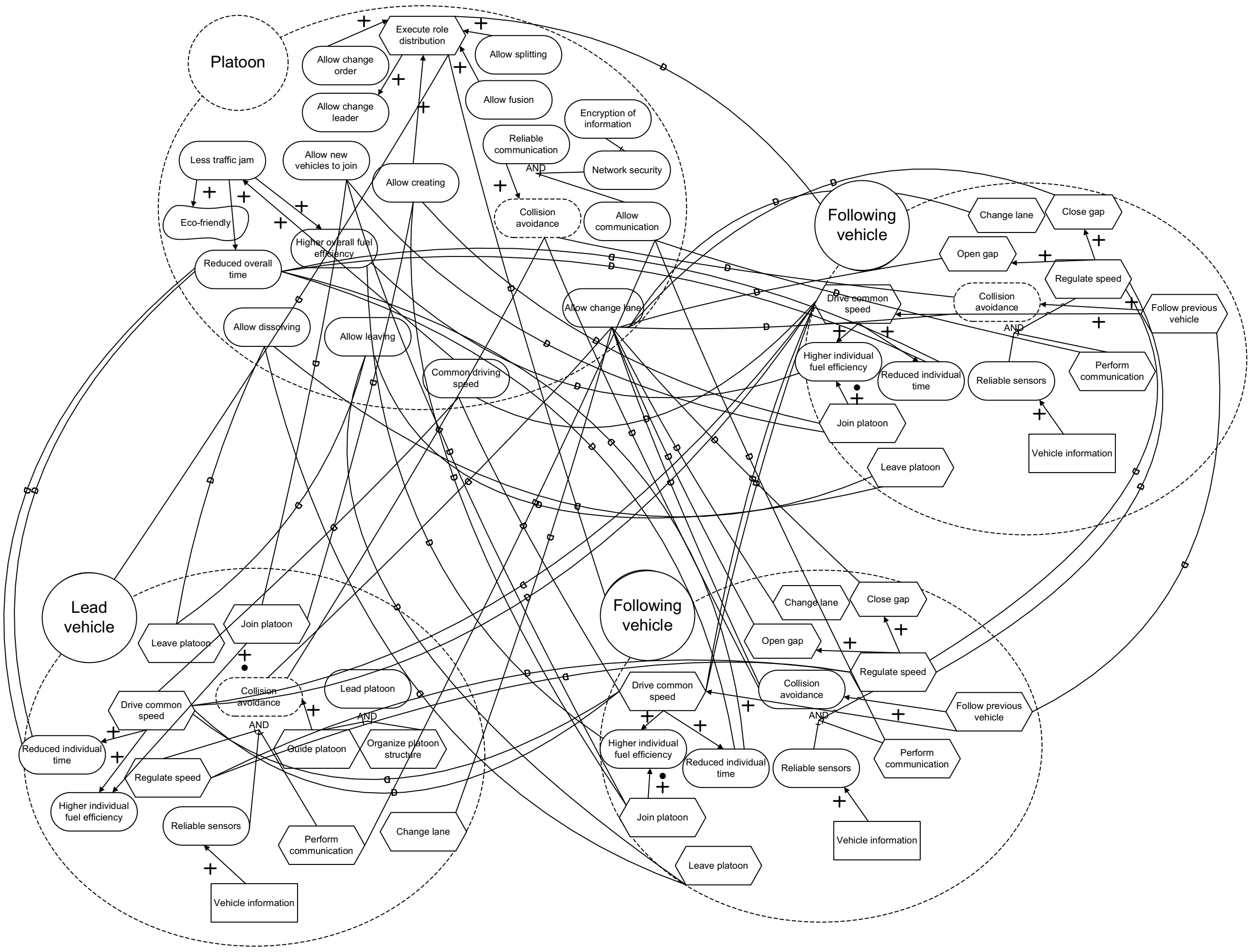


Fig. 23 Cooperative adaptive cruise control platoon without extension

This is not surprising in so far as the platoon does only exist in the interplay of its physically partaking CACCs. Therefore, each functionality the platoon shall exhibit must originate from at least one CACC. For example, the platoon shall be able to allow new vehicles to enter the platoon (i.e. it has the goal *allow new vehicles to join*). To fulfill this goal, the platoon depends on the vehicles in the platoon that need to open a gap so that a new vehicle can join the platoon by entering this gap. Hence, the goal *allow new vehicles to join* of the platoon depends on the CACC tasks *open gap*.

There are also mutual dependencies between the platoon and the vehicles, for example, the goal *reduced individual (driving) time* of the CACC depends on the goal *reduced overall (driving) time* of the platoon and vice versa. This is shown by the bidirectional dependency between those goals.

Furthermore, dependencies between the different roles exist and are modeled. For example, a following vehicle must be able to execute the task *follow previous vehicle.* Therefore, it depends on other CACCs (either in the role following vehicle or lead vehicle). In this way it is specified that each following vehicle needs to follow another vehicle. In cases where the vehicle ahead is also a following vehicle, a self-dependency is used to show that a following vehicle depends on another following vehicle regarding the fulfillment of the task *follow previous vehicle*. This self-dependency possesses a condition whereby the dependency only exists if there are more than one following vehicle in the platoon, because otherwise a following vehicle could not follow another following vehicle.

With respect to RQ1.1 we can state that the iStar extension is applicable to model the case example of a CACC. Important aspects of the case example can be specified and the existing mutual dependencies between the different roles of the collaborative CPS and the collaborative network can be defined accordingly.

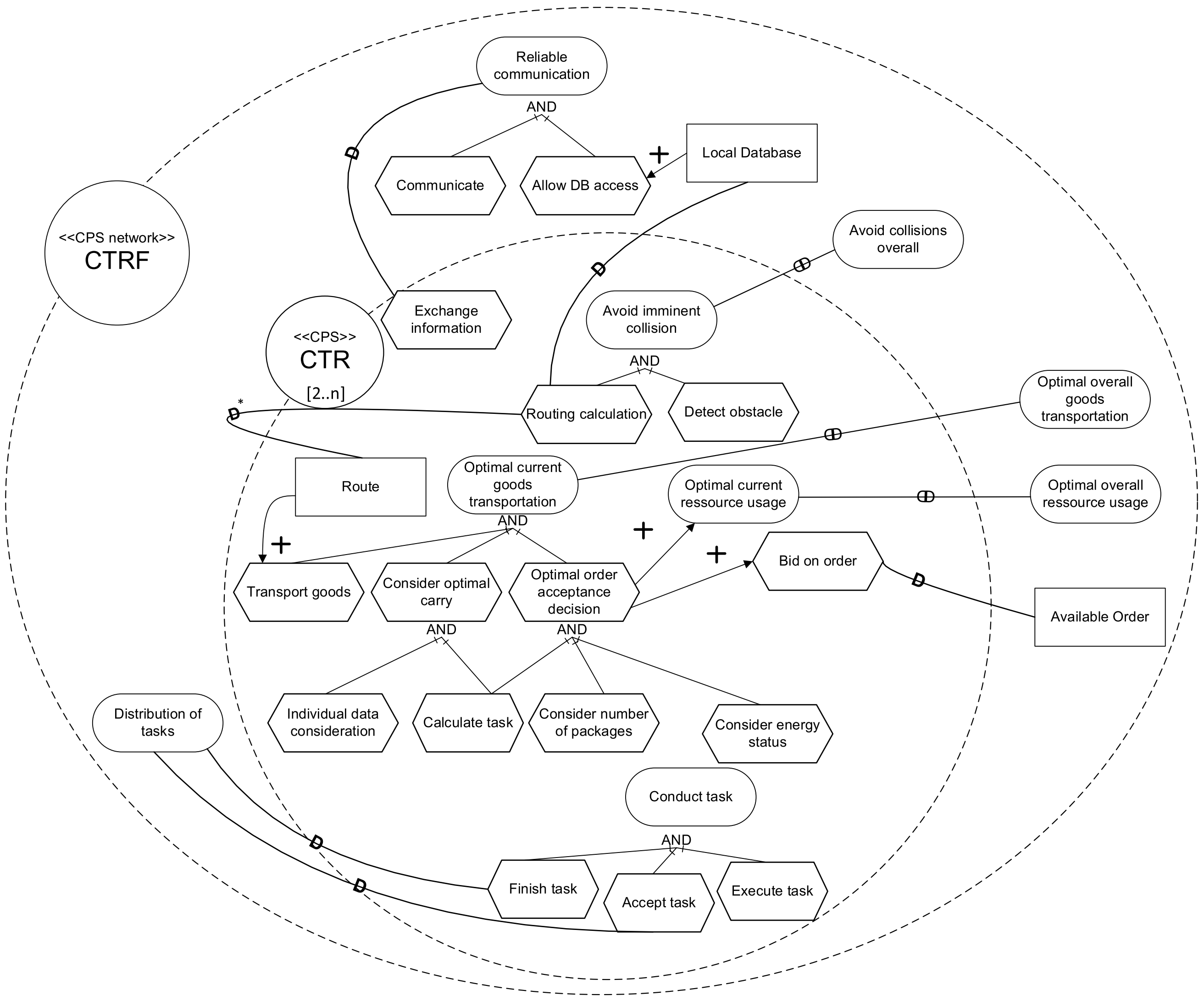


Fig. 24 Goal model collaborative transport robot fleet with extension

5.2.2 Comparison with original iStar notation for the cooperative adaptive cruise control case example (RQ1.2)

To investigate RQ1.2 Fig. 23 shows a goal model for the CACC that has been created without the proposed extension. As can be seen, this model is considerably more complex and contains more connections. For instance, another actor is needed representing another following vehicle. As it would otherwise not be possible to describe that the task *follow previous vehicle* depends on other following vehicles to also follow the previous vehicle. Among others, the bidirectional dependency and the grouped dependency reduce the number of lines which improves the readability of the model. In summary, we can state that the goal model from Fig. 22 which was created using the iStar extension is more concise than the goal model from Fig. 23.

5.2.3 Application to the collaborative transport robots case example (RQ2.1)

Figure 24 shows the resulting goal model for the case of the collaborative transport robots. As can be seen, the collaborative transport robot (CTR) is directly represented by an individual actor as no roles need to be distinguished. However, as a CTR partakes in a collaborative network, i.e. in a collaborative transport robot fleet (CTRF), another actor is used to represent this network. Like for the CACC case example, the nested representation for network and CPS is used. As only one type of CTR does exist and no roles need to be distinguished, the network does not depend in its goal fulfillment on multiple CPS of different types or roles. Consequently, unlike for the CACC case, no grouped dependencies have been used. However, self-dependencies do exist, which describe dependencies between identical CTRs. For

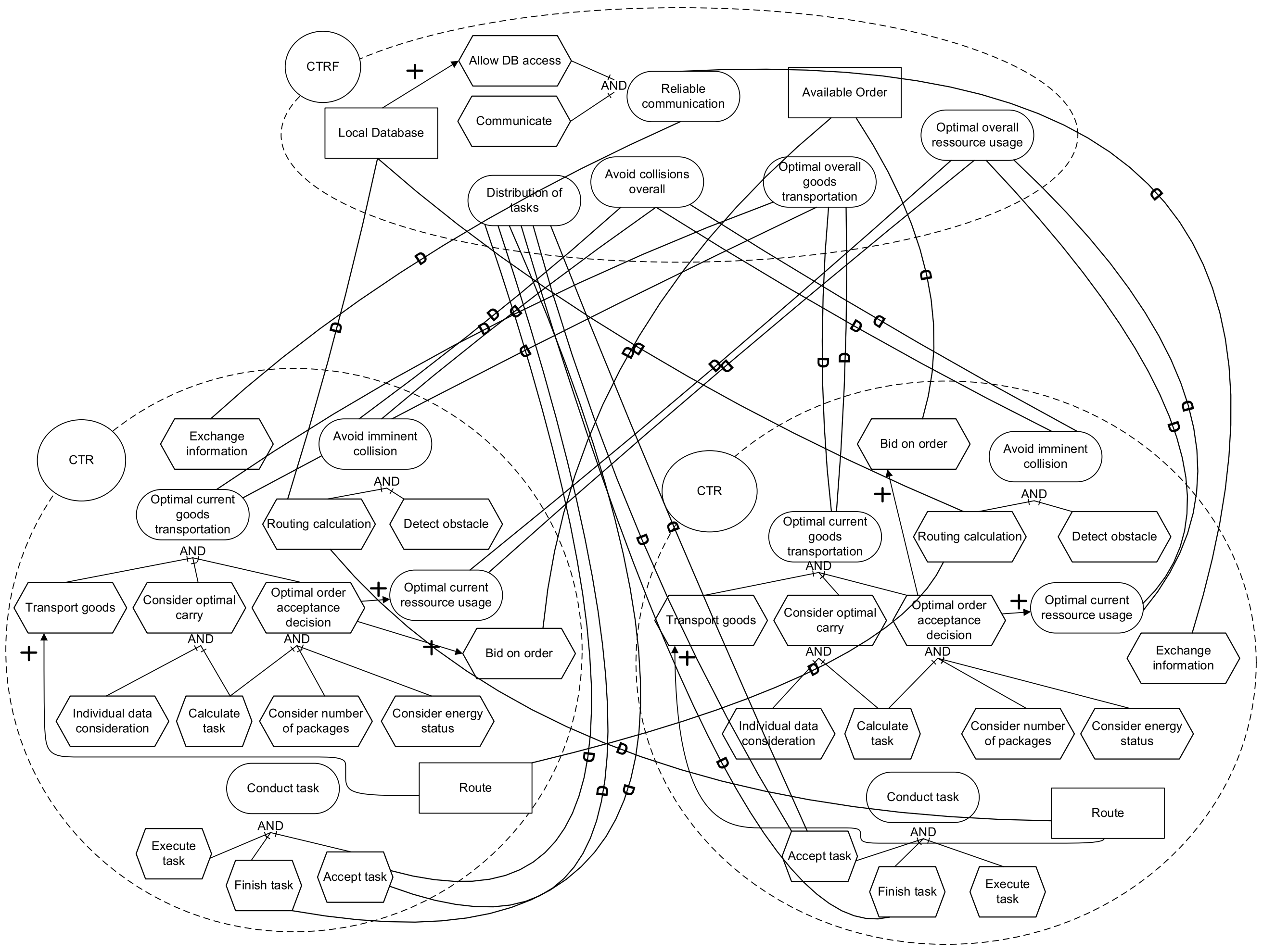


Fig. 25 Goal model collaborative transport robot fleet without extension

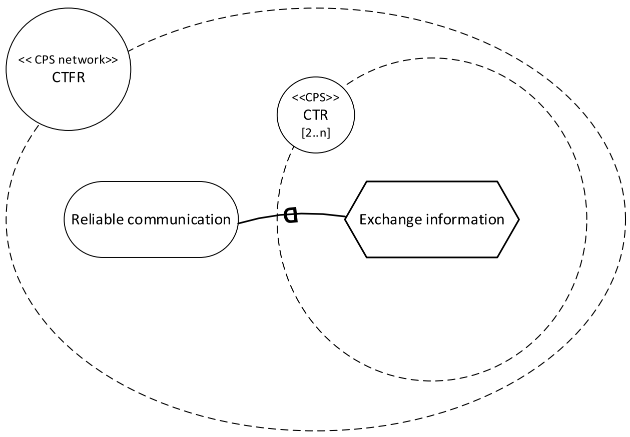


Fig. 26 CTR actor nested in CTRF actor

instance, for calculating a new route, a CTR depends on the current routes of the other partaking CTRs as otherwise the goal *avoid imminent collision* and consequently the network goal *avoid collisions overall* could not be reached.

As in the CACC case, there are goals that both the robot and the robot fleet share, such as *avoid collision overall* and *avoid imminent collision*. These differ in that the robot fleet wants to achieve the goals for all robots while the individual robot is primarily concerned with its own goals. But as these goals are interdependent, they are linked in the goal model by a bidirectional dependency.

With respect to RQ2.1 we can state that it is possible to document the goals of the CTRF and the goals of the CTR and relate them to each other. Hence, the proposed iStar extension is also applicable to the case of collaborative transport robots.

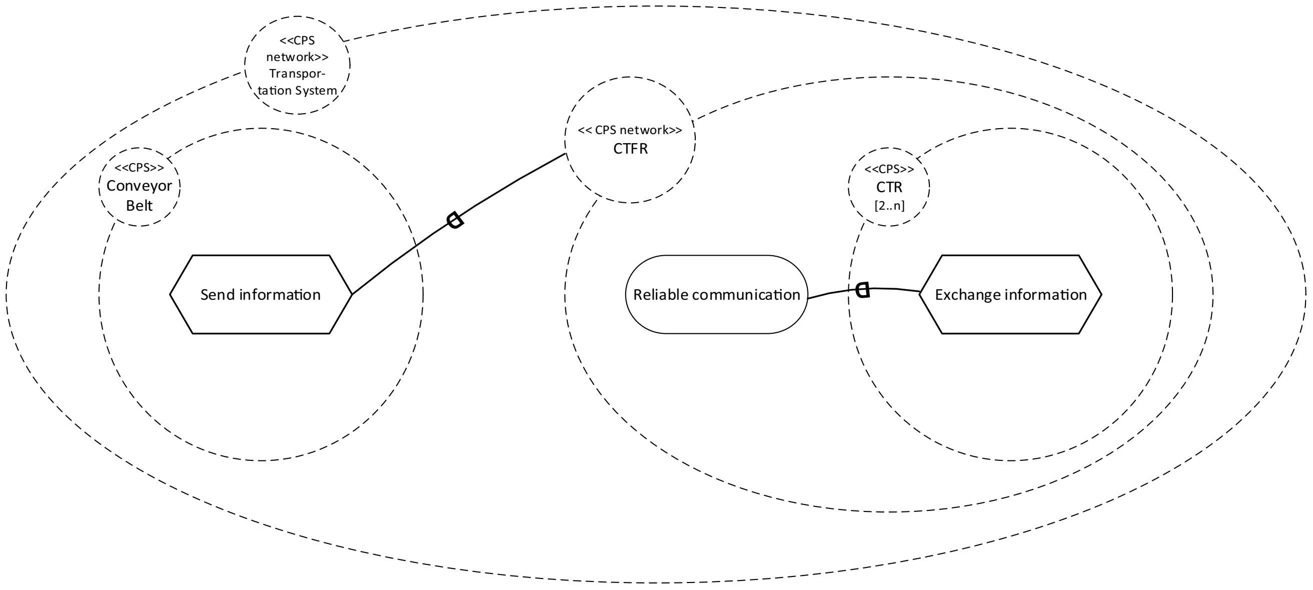


Fig. 27 Distinction between collaborative CPS belonging to the CPS network and collaborative CPS Not belonging to the CPS network

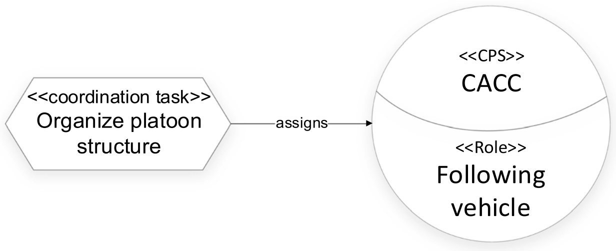

<<coordination task>> Organize platoon structure
assigns
<<CPS>> CACC
<<Role>> Following vehicle

Fig. 28 A coordination task in the CACC case example to assign the role of following vehicle

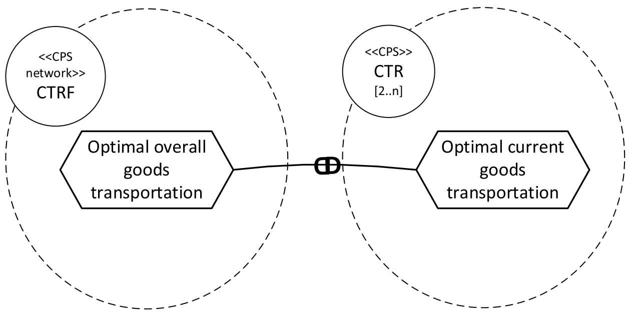

<<CPS network>> CTRF
Optimal overall goods transportation
<<CPS>> CTR [2..n]
Optimal current goods transportation

Fig. 29 Bidirectional dependency between two tasks

5.2.4 Comparison with original iStar notation for the collaborative transport robots case example (RQ2.2)

For the collaborative transport robots, we also investigate RQ2.2 by comparing the goal model shown in Fig. 24 to a goal model that does not use the extension. This goal model is shown in Fig. 25. Although this model is not as large and complex as the goal model for the CACC example, it can be easily seen, that the model is much larger and more complex than the CTR goal model that uses the extension. This is particularly due to the need for more dependency links. Again, two CTR actors are necessary to express the self-dependency between different identical robots. Therefore, we can state that Fig. 24 is more concise than Fig. 25. Also discussions with industry partners showed that industry professionals do not miss any information in the goal model using the extension compared to the other goal model but do find Fig. 24 more intuitive and comprehensible than Fig. 25, as the number of dependencies limits the overall readability.

5.3 Usefulness of proposed modeling elements

We will illustrate the usefulness of the individual modeling elements using excerpts from the models of the case examples. Furthermore, we discuss our major insights gained from the application and discussion with domain experts.

5.3.1 The use of actors (RQ3.1)

Particularly, the use of the nested representation of CPS network and collaborative CPS partaking was considered very helpful as this allows getting an intuitive picture of what the composition of the CPS network looks like. Figure 26 gives a brief fragment of the nested actor notation from the CTR example. As the CTR is a part of the CTRF, the CTR actor is modeled inside the CTRF actor. Still, the intentional elements of the CTR are separated from those of the CTRF by the actor boundary of the CTR.

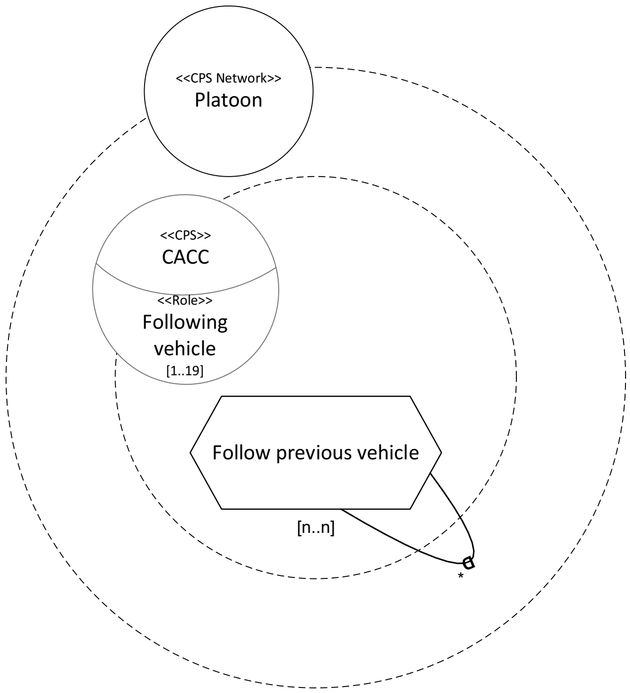


Fig. 30 Self-dependency link

Another advantage was not included in the original case example description but revealed during discussions. The CTRF typically does not operate on its own but also interacts with other systems in a smart factory, production machines, storage capacities, and even with other transportation systems. Hence, the nested representation is particularly suitable for displaying such systems separately from each other. As in Fig. 27, the conveyor belt is not part of the CTRF. It can still communicate with the CTR to announce goods in need of pickup. This way even further nesting might be useful to express different degrees of cohesion and collaboration. For instance, in the CTR case, a smart factory is composed of several collaborative CPS, some of which are assigned transportation tasks. The collaborative CPS with transportation tasks can, thus, be composed to the transport system of the smart factory. Hence, the conveyor belt and the CTRF can be nested into the transport system, which itself may be nested into the smart factory.

The extension also allows for defining a collaborative CPS (without defining a role) and a role this collaborative CPS can assume in the same goal model. The need to do so never arose in any of the case studies. In practice, it is usually relevant to either investigate the overall goals of a CPS (i.e. without considering a specific role) or to investigate issues that relate to the roles the CPS take.

5.3.2 The use of the coordination task (RQ3.2)

It has been shown that in the two case examples investigated, coordination tasks are less often needed as has been assumed upfront. However, the coordination task has shown useful to indicate that a certain role is assigned by a particular task, belonging to a particular actor. This helps engineers in defining responsibilities, i.e. which entity of the CPS network shall be responsible for the assignment of roles. This is shown in Fig. 28, the CACC in the role lead vehicle has the task to organize the platoon, i.e. it coordinates which CACC joins the platoon and which CACC needs to leave the platoon. Therefore, it is able to assign the role of a following vehicle to another CACC.

5.3.3 The use of bidirectional dependencies (RQ3.3)

The main benefit of the bidirectional dependency is seen in reducing the number of dependencies displayed. This is illustrated in Fig. 29, which displays two actors: the collaborative system network *CTRF* and the individual CTR. Both systems have the task to fulfill an optimal goods transportation. The task of the CTRF refers to the entire network of collaborative CPS and is therefore called *optimal overall goods transportation*, while the task of the CTR refers only to the robot itself and its current task, therefore it is called *optimal current goods transportation*. Both tasks depend on each other as the individual CTR can only reach an optimal transportation solution when the overall routes (i.e. also the routes of the other CTR) are optimized so that no collisions and backups occur. However, to achieve this the CTRF depends on each individual CTR to find optimal routes within the existing optimized overall routes. Hence, both tasks depend on each other. Using the bidirectional dependency, not only the number of dependency links is reduced, but as shown, the bidirectional dependency also indicates a very close relation between both tasks. This allows engineers to easily detect parts of the collaborative CPS network that can only be achieved in collaboration and must therefore be given particular care during implementation.

5.3.4 The use of self-dependencies (RQ3.4)

The major use of self-dependencies must be seen in its ability to reduce the number of actors shown in the CPS network as every system type or role is only displayed once regardless of how many instances are actually partaking in the system network. For instance, a collaborative system such as a platoon consists of several individual CACCs collaborating.

Fig. 31 Initial situation

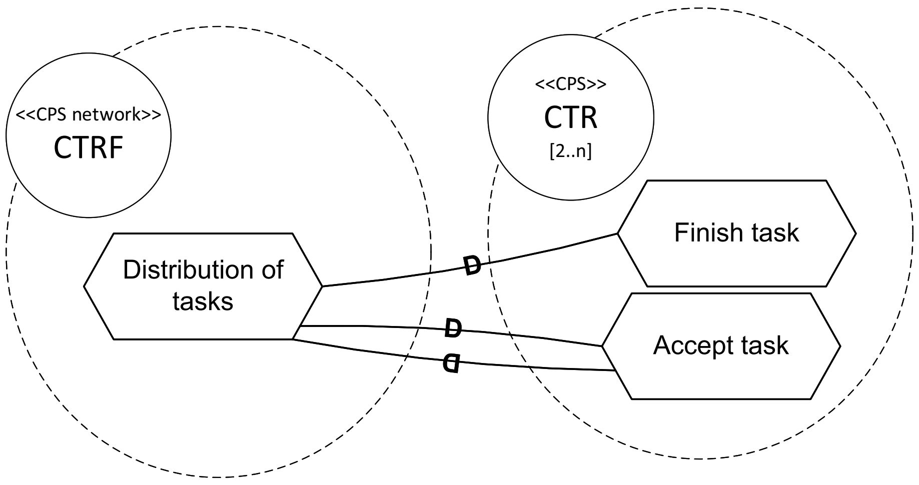


Fig. 32 Multidirectional dependency

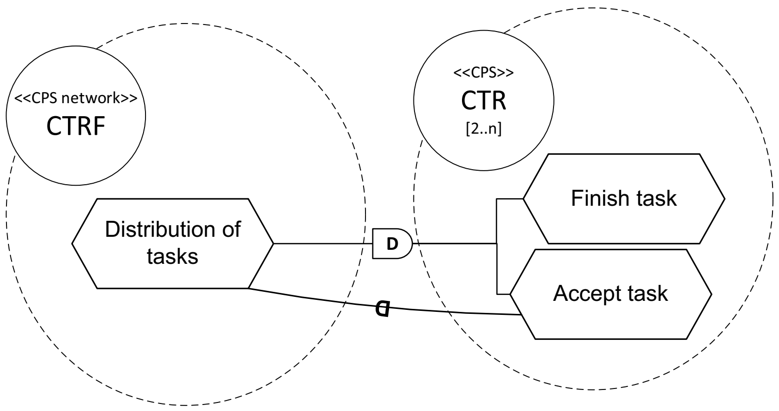
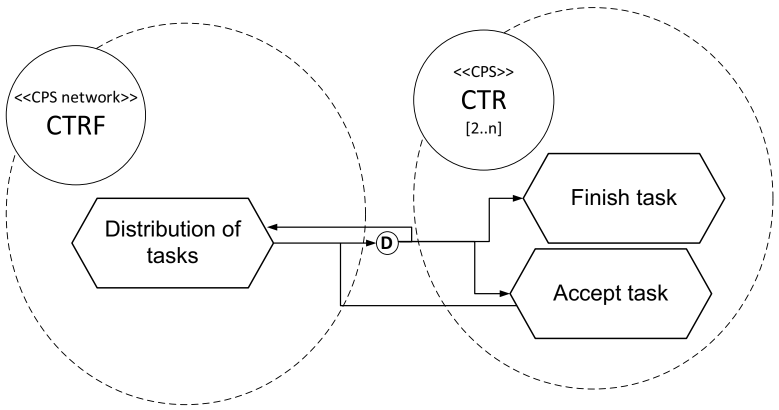


Fig. 33 Too complex representation of a multidirectional dependency



It is not feasible to represent all these configurations in models as CPS networks are dynamic, thus resulting in a large number of similar albeit slightly different configurations, nor is it feasible to represent large configurations such as platoons consisting of more than five vehicles in one model if each vehicle is depicted separately. Therefore, the CACC represents all instances of the CACC in a platoon.

In Fig. 30 it is shown that CACCs in the role *Following vehicle* are part of the platoon. And, to allow driving in a platoon formation, each following vehicle needs to follow its predecessor, i.e., the previous vehicle. Therefore, it depends on other following vehicles, which also need to each follow their predecessor. This is represented by a self-dependency. While this construct is seen as useful, care must be given to avoid misinterpretations, i.e. an individual system does not depend on itself but on other systems of the same type. However, using the asterisk was deemed very helpful as it indicates that it is not a normal dependency. People familiar with the self-*-properties directly—most likely out of the context of their domain knowledge—related this self-dependency not to the individual system on an instance level but as desired on a type level, i.e. that system of this type are self-dependent on other systems of this type.

5.3.5 The use of grouped dependencies (RQ3.5)

Grouped dependencies allow to further reduce the number of dependency links to be displayed. Instead of having multiple separate dependency links, dependencies are grouped, the dependency symbol is only shown once, and each involved intentional element connects only with one line to the symbol.

For an illustrative example, Fig. 31 shows the actors CTRF and CTR. The goal *Distribution of tasks* and the tasks *Finish task* and *Accept task* are connected through three dependency links with each other. For example, the task *Finish task* depends on the goal *Distribution of tasks*, since a CTR can only process and complete a task if it has been previously assigned to it. In addition, the goal *Distribution of tasks* depends on *Accept task*, since this goal is only achieved if a CTR who is assigned a transportation task also accepts this assignment. In this simplified model this may look comprehensible, but with an increasing number of actors and their goals and tasks, the number of dependencies can also increase.

Therefore, as shown in Fig. 32, the two actors are connected by using a grouped dependency. This allows us to express that the task *Distribution of tasks* depend on the tasks *Finish task* and *Accept task*. Please note that in this case we could also have reduced the complexity by using a bidirectional dependency between *Distribution of tasks* and *Accept task*.

Actually, in the early stages of the development of the extension, we aimed for always needing just one dependency link between two or more connected intentional elements. However, this was not achievable, as the resulting dependency constructs were complex and often misunderstood. We illustrate this in Fig. 33, which shows the initial idea to use only one multidirectional connector to connect all incoming and outgoing edges. As shown in Fig. 33, however, this is comparatively more difficult to understand than the example in Fig. 32.

5.3.6 The use of multiplicities for dependencies (RQ3.6)

Much akin to the discussion for the self-dependency, the multiplicities for dependencies were a necessary means to achieve displaying just one actor that represents all collaborative CPS of the same type and in the same role. Otherwise, it would not be possible to distinguish, e.g., the following two situations: (1) an intentional element of one collaborative CPS of a certain type depends on an intentional element of one collaborative CPS of another type, and (2) an intentional element of one CPS of a certain type depends on intentional elements of multiple/all CPS of another type that do exist. Therefore, the use of multiplicities is necessary. From our observations the use was quite intuitive as multiplicities are well known from UML class diagrams and other modeling languages and, thus, their use did not lead to any misinterpretations.

The use of multiplicities also shows the need to separate specification and analysis of goal models for collaborative CPS. For specification purposes, we need abstractions to reduce the complexity of the models and allow specification of CPS networks in a manageable fashion. Therefore, we use the concept of multiplicities to cope with the sheer number of configurations to be specified at design time. For analysis purposes, however, we need to ensure proper functionality in all situations. Thus, for runtime analysis all possible configurations need to be considered.

During specification we define what configurations may exist and thus need to be considered, however, we do not place emphasis on how these form or dissolve. The specification using multiplicities defines that the number of actors will vary in a known range at runtime but not how these variations occur at runtime. For example, in the robot case example, if we model that the fleet consists of three to eight identical robots that have the same role, it does not state how the fleet can actually vary between three and eight robots. If we want to state that, for instance, a robot might break down, we need to explicitly specify another actor type robot in the role “defect robot” and a coordination task can then be used to explicitly define how a robot can be assigned the role “defect robot”.

5.4 Remaining challenges

5.4.1 Limitations of the iStar extension (RQ4.1)

Despite the usefulness of the proposed extension and the overall applicability of the proposed extension, we have found some limitations. While, so far, we have briefly mentioned some remaining challenges in Sect. 4 and sketched limitations throughout Sects. 5.2 and 5.3 in this section we will discuss the most important limitations in more detail and provide insights into rationales.

5.4.1.1 Contribution links depending on the current CPS network configuration As outlined in Sect. 2.3.6 there is a need to allow modeling contributions where the value of the contribution depends on the current configuration. We use multiplicities for actors and dependencies to specify multiple configurations within one single goal model. Thus, multiple configurations are incorporated in one model and hence the nature of a contribution might be ambiguous. As briefly discussed in Sect. 4.2.3, we propose the use of configuration-dependent contribution value labels for these situations. This simple solution is a result of the inability to define this complex problem with a precise but at the same time comprehensible notation. Hence, the current solution for Challenge 6 is largely based on a tradeoff between expressiveness and proposing an easy to use iStar extension. We decided to go for simplicity to provide easy access for industry professionals, thus limiting the expressiveness of varying contribution links depending on the CPS network configuration.

Industry professionals have stressed the importance of investigating and analyzing these situations closer. So far, our solution to this need is by modeling concrete configurations in distinct models that allow detailed investigation and comparison. In doing so, the benefits of having only one model to maintain and analyze vanish and the effort needed increases. In addition, when it comes to automated support, the current solution is also not sufficient as a precisely defined contribution depending on the respective configurations is needed to allow for any kind of automation.

5.4.1.2 Missing support for in-depth analysis of concrete instance configurations There is not only a need to define and investigate the impact a contribution link has based on different configurations but more generally to investigate concrete investigations in-depth. Due to the use of abstractions in the specification (i.e. representing CPS of similar type with just one actor having multiplicities), it becomes difficult to reason about similar CPS that try to achieve different goals at the same time. For instance, two CTR might collaborate in a CTRF, while they have in principle the same goals due to the current context situation the robots try to achieve different goals. As an example, both robots might have different battery-levels. Depending on the current battery level, the goals to be fulfilled change. With lower battery-levels robots shall aim for resource preservation, while with higher battery-level the maximum number of transportation tasks shall be processed. Therefore, a means to generate and investigate concrete instance level configurations is needed. We have already shown the applicability, effectiveness, and usefulness of such generations for scenario model using ITU Message Sequence Charts (cf. [105, 106]). Due to the feedback we received from our industry partners, we are confident that this is transferable to goal models and will allow more in-depth analysis on the impact of certain configurations.

5.4.1.3 Interpreting complex relations involving multiplicities The interpretation of complex relations that involve multiplicities may be error prone. Due to the high amount of information to be processed for correctly interpreting the meaning of grouped dependencies with multiplicities involving actors with multiplicities, there is a risk of misinterpretation. However, the reduced size of the model itself due to the use of these constructs was very much appreciated.

Currently, we assume that the fulfillment of a dependency with multiplicities means that all depender elements are fulfilled if all dependee elements (AND-dependency), at least one dependee element (OR-dependency), or exactly one dependee element (XOR-dependency) are fulfilled. So far, we found this definition to be sufficiently comprehensive and intuitive. However, there might be the need to express that just one of the depender elements will be fulfilled or just a certain number. For instance, due to access restrictions a resource might be only accessible by exactly one CPS. Therefore, not all depending CPS can access this resource at the same time and therefore, only one CPS can fulfill its goals that depend on this resource. Furthermore, it is also conceivable that not all, at least one, exactly one dependee elements shall be fulfilled but a concrete number (or within a concrete range). This is, for instance, the case if measurements shall be validated across different members of a CPS network. To do so, it is not necessary for the measurement to be provided by all elements, but by at least two or three (as otherwise no meaningful detection of outliers is possible).

5.4.1.4 Goal fulfillment analysis and semantics of the iStar extension Semantics for iStar and GRL are typically defined based on goal fulfillments [107, 108]. Recommendation Z.151 refers to this as the "GRL model satisfaction analysis". For instance, the semantics of an AND-decomposition is defined such that the super-intentional element is fulfilled if all intentional elements it is composed of are fulfilled. Therefore, typically values for qualitative (e.g., high satisfaction, medium satisfaction, low satisfaction) and quantitative analysis (e.g., 0–100% contribution to the satisfaction) are defined. This allows, among others, automated analysis of the overall goal fulfillments (i.e. can the overall goals be sufficiently achieved) or calculating optimized goal fulfillments (i.e. which subgoals—under consideration of conflicts, etc.—should be fulfilled to reach the best possible goal fulfillment).

So far, we have focused on modeling collaborative CPS and manual analysis by human engineers. However, due to the complexity automated support based on clear semantics is desired. Particularly, goal fulfillment analyses can support engineers in identifying problematic CPS network configurations, etc. Thus, the aforementioned precise semantics for goal fulfillments will need to be established. Particularly, the semantics for goal fulfillment of multiplicities, i.e. the impact on the actors with multiplicities, and on the intentional elements involved with dependencies with multiplicities need to be precisely defined. At this point, we have gathered a broad understanding from the engineers about what it means when a goal is fulfilled. From what we have learned so far in collaborative CPS networks different degrees of goal fulfillment must be considered. For instance, the CTRF will not always fulfill a goal to 100% but in many cases to a point where it suffices. I.e. equal battery consumption across all CTRs will, depending on the configuration, not be achievable. However, for many of these configurations a less then optimal equality is also acceptable. Thus, when it comes to goal fulfillments and automated analyses thereof, more precise means are needed to express such complex situations depending on the configuration.

5.4.1.5 Circular dependencies Related to the aforementioned point, when analyzing goal fulfilment circular dependencies are a problem, as this typically can be interpreted as a deadlock, however, for collaborative CPS expressing such circular dependencies is important.

As could be seen from the application of the case examples, circular dependencies occurred regularly. Furthermore, we even introduced some elements (e.g., bi-directional dependencies) that contradict the fulfillability of the overall model in general as these are circular per definition. However, we deem these elements important. For instance, it is necessary to express that a CPS network cannot fulfill its goals if the goals of the individual systems are not fulfilled and vice versa. A CACC wanting to reduce the overall travel time depends on the platoon to reach this goal. However, the platoon depends on each CACC in the platoon trying to reach this goal as well. In other words, the platoon can only drive as fast as its slowest member.

The many circular dependency relations between multiple intentional elements of a CPS network and partaking CPS were not identified as problematic by industry professionals. Even more, they were highly appreciated as they express the inherently collaborative parts of the interplay between the individual CPS and the CPS network. Thus, these constructs are severely needed by industry professionals to foster their analysis in early stages. However, they are problematic for goal fulfillment analysis, which was also seen as desirable to support the development of collaborative CPS.

5.4.1.6 Tool support While we provide Visio stencils to create goal models for networks of collaborative CPS, the tool cannot prevent modelers from creating goal models that violate syntactical or well-formedness rules. Consequently, the responsibility for adhering to those rules lies completely with the modeler. This can be problematic for inexperienced modelers who are not that familiar with the rules and therefore more likely to create flawed goal models for networks of collaborative CPS.

Additionally, the tool does not provide automated analysis support for goal fulfillment. Goal models can be analyzed automatically to reason about goal fulfilment. This, however, is currently not implemented, leaving the requirements engineer with the task of having to analyze the goal models manually.

Currently, we are using Microsoft Visio as modeling tool, which also allows implementing feasibility checks and the goal fulfillment analysis via add-ins. Due to the popularity of Visio, we intend to keep and enhance this, instead of using another modeling tool which has already basic checks for goal models implemented. The main reason for this is the broad availability of Microsoft Office products in German industry. This leads to easy application as modelers already have sufficient experience with the tool.

5.4.2 Industry needs for future work (RQ4.2)

Based on the limitations discussed above, the need for future work arises. While we have already briefly discussed this need in Sect. 5.4.1. In this section, we briefly summarize the major needs identified. Particularly, we found needs for:

Contribution links depending on the configuration. The solution needs to allow for precise definition of the impact certain configurations or changes in a configuration have on the value of a contribution link and at the same time must be reasonably easy to model and comprehend that it is of value for manual analyzes and discussions in early development phases.

Providing support for interpreting complex relations involving multiplicities. Correctly interpreting dependencies with multiplicities and/or dependency groups can be difficult. While having groups and multiplicities allow for reducing the size of the goal model considerably (as otherwise each dependency would have to be modeled individually), there is an increased risk of misinterpretation. One possible solution to this issue could be the illustrative generation of model excerpts focusing on a particular dependency that allow for examining this dependency in the familiar style with no groups or multiplicities.

Automated goal fulfilment analysis and formal definition of semantics. Some new constructs (e.g., circular dependencies, grouped dependencies) hinder the use of established goal-fulfilment analysis approaches. To provide automated support for goal fulfilment analysis for goal models of networks of collaborative CPS, a precise definition of formal semantics is necessary. Particularly,

Table 3 Short summary of the principal findings for each research question

Research questions		Findings
RQ1.1	*Is the proposed iStar extension applicable to model a cooperative adaptive cruise control?*	The iStar extension is applicable and the application resulted in a valid model for the cooperative adaptive cruise control that has been evaluated by industry professionals as sufficient and helpful in the engineering process
RQ1.2	*Is the proposed iStar extension applicable to model collaborative transport robots?*	The iStar extension is applicable and the application resulted in a valid model for the collaborative transport robots that has been evaluated by industry professionals as sufficient and helpful in the engineering process
RQ2.1	*Does the use of the proposed iStar extension lead to a more concise yet still comprehensible model of the cooperative adaptive cruise control?*	The resulting model is more concise than a comparable model created without the extension. Particularly, the number of actors shown is reduced and the number of dependency links needed is significantly smaller. Furthermore, other approaches need to define a single model for each configuration, thus, the number of diagrams needed to describe the entire CPS network is also reduced considerably
RQ2.2	*Does the use of the proposed iStar extension lead to a more concise yet still comprehensible model of the collaborative transport robots?*	The resulting model is more concise than a comparable model created without the extension. Particularly, the number of actors shown is reduced and the number of dependency links needed is smaller. However, the effect is not as large as observed for RQ2.1. Nevertheless, also in this case, other approaches would need to define a single model for each configuration, thus, the number of diagrams needed to describe the entire CPS network is also reduced considerably
RQ3.1	*Is the use of collaborative CPS and the network of collaborative CPS as actors useful?*	The differentiation between collaborative CPS and the CPS network allows for expressing goals on different levels of abstraction and relating them to each other. I.e. it can be expressed how CPS network goals can be achieved based on the collaborative CPS' goals. Particularly, the use of stereotypes allows to easily distinguish both actor concepts and the use of nesting results in smaller models while at the same time making the hierarchy between CPS network and collaborative CPS intuitively clear
RQ3.2	*Is the use of the coordination task useful?*	The coordination task is useful as it allows to document changes of roles that may occur during runtime and indicate how they are triggered and who is responsible for changing the role of an actor. Thus, the coordination task concept allows to express a complex situation by using just one intentional element
RQ3.3	*Is the use of bidirectional dependencies useful?*	Bidirectional dependencies considerably reduce the size and complexity of the resulting models. Collaborative CPC are—as is quite obvious—collaborating and therefore, often rely on each other, furthermore often the CPS network relies on the individual CPS and vice versa. Thus, the bidirectional dependency reduces the number of dependencies used and adds the notion of mutuality which is not given by having two independent dependency links
RQ3.4	*Is the use of self-dependencies useful?*	The self-dependency allows to express that one system depends on another system of the same type and role. Thus, the self-dependency allows expressing different systems of the same type with just one actor element Consequently, the size of the goal model can considerably be reduced, and the clarity of the models is improved

Table 3 (continued)

Research questions		Findings
RQ3.5	*Is the use of grouped dependencies useful?*	Grouped dependencies can reduce the number of dependencies to be modeled and therefore reduce the size of the model and add to model clarity. However, it is to mention that in some cases the use of grouped dependencies can result in too complex to read dependencies. This is particular the case when dependent intentional elements are spatially distant. Therefore, this modeling element should not be used regardless of the layout of the model, but the current layout should be taken into account. However, in several situations the model complexity can considerably be reduced
RQ3.6	*Is the use of multiplicities for dependencies useful?*	As is the case for self-dependencies, this modeling element allows to model different systems of the same type with just one actor element. Consequently, the size of the goal model can considerably be reduced, and the clarity of the models is improved
RQ4.1	*What are limitations of the proposed iStar extension?*	As advanced automated support is desired, the proposed iStar extension is limited as no formal semantics are provided yet. This is particularly the case when it comes to circular dependencies and the proper interpretation of complex dependencies that involve multiplicities. Furthermore, by providing simplified type-level specifications using abstractions to allow for concise models, the ability to reason about concrete, potentially hazardous, instance configurations is limited
RQ4.2	*What are industry's needs for future work?*	There is particularly a need for revisiting the solution for Challenge 6 by semantically defining contribution links, where the contribution value depends on the configuration of the CPS network (i.e. is related to actor multiplicities). Furthermore, there is a need to define formal semantics for analyzing circular dependencies and to allow for automated goal fulfillment analysis. Furthermore, automated support for analyzing concrete instance-configurations is needed. These automated aspects can also be supported by adequately developing the tool support further

a solution for the needed *circular dependencies* must be provided.

Advanced tool support. To better support developers in creating goal models for networks of collaborative CPS, future tool support should also include checks for adherence to modeling rules as well as support for automated goal fulfillment analysis. As current tool support already provides stencils for creating goal models for networks of collaborative CPS, these analysis functionalities can be implemented as Visio add-ins, so that already created goal models can be analyzed,

6 Discussion

6.1 Summary and major findings

In this paper, we developed a GRL-compliant extension to the existing iStar goal modeling language for goal modeling of collaborative CPS and CPS networks. With the choice of iStar, we have adopted a widely used goal modeling approach. To do so, we integrated our extensions into the iStar metamodel and defined the concrete syntax to specify what the goal modeling extension looks like graphically considering best practices for model notation creation. Furthermore, the well-formedness rules were defined to describe constraints for the goal models. Our extension was evaluated using two industrial case examples: a CACC (cooperative adaptive cruise control system) from the automotive industry as well as a CTRF (collaborative transport robot fleet) from the industry automation domain.

For the main results of our evaluation we can state that:

- *RQ1:* Our evaluation shows that the iStar extension is applicable to industrial case examples of collaborative CPS. The resulting models were well received by industry professionals and rated as very helpful in the engineering process as the goals of a multitude of configurations to be considered can be easily expressed in manageable models.
- *RQ2:* The goal models with the extension include fewer actors and dependency lines compared to the goal mod-

els without the extension, although the same situation is shown in both. Therefore, the use of the iStar extension results in more concise goal models.

- *RQ3:* We have shown that each of the proposed modeling elements contributes to modeling complex situations in a clear and concise way and thus yields the creation of extensive and yet easily readable models. According to Moody's principle of complexity management, it was shown that the modeling elements of the extension are suitable to reduce the complexity of the iStar models when modeling collaborative CPS that interact in dynamic CPS networks.
- *RQ4:* Finally, we have investigated shortcomings of the extension and needs for future work. Among the remaining challenges, most notably is the proper definition of formal semantics that also consider circular and bidirectional dependencies, take multiplicities for contributions into account and, thus, allow for automated goal fulfillment analysis.

For a summary of the major findings for each sub-research question, please refer to Table 3.

6.2 Threats to validity

To evaluate our proposed extension, we used a commonly used evaluation approach (e.g., [17, 100, 109]). However, like all evaluation approaches, case study evaluations have some drawbacks [110, 111]. As recommended [112], we discuss those drawbacks in terms of conclusion, external, internal, and construct validity.

6.2.1 Conclusion Validity

Conclusion validity deals with drawing correct conclusions from the application results and findings. As case studies usually draw conclusions from few cases studies, conclusion validity must be considered rather low. To somewhat alleviate this threat, we conducted two case studies in different domains. We showed that both case examples can be modeled appropriately using the proposed extension. We furthermore showed that industry professionals found the created models easy to understand and helpful. However, at this point we cannot make any claims as to how well industry professionals can create goal models using the extension on their own.

6.2.2 External validity

External validity deals with the ability to generalize results to cases outside those studied. Collaborative CPS networks are of a diverse nature and exist in a variety of domains (such as energy, aviation, etc.) with specific characteristics. We cannot rule out the need for further adjustment to the extension for goal models of collaborative CPS for those domains. However, our case study has shown the applicability of the proposed extension for goal models of collaborative in two different domains, automotive and industry automation. We expect the proposed extension to be at least somewhat beneficial to the development of collaborative CPS from other domains.

Another remaining threat is if industry will ever use the extension on their own. Particularly, there is a threat that goal models at all will not be used by industry as recent studies have shown industry's reluctance to the use of goal modeling [26, 27]. While we cannot rule out this possibility, we want to highlight that we have shown for the automotive industry that goal models are welcomed when the introduction is accompanied with training and tutoring sessions [52]. Regarding the robot case example, the idea of using goal models was very well-received as it was a good match for how engineers thought of their robots. We found that the engineering was already centered around the goals, the individual robots have and around questions like when shall a robot fulfill which goal, etc. However, previously this was not made explicit and, therefore, the benefits of using goal models were quite obvious to our partners.

6.2.3 Internal validity

Internal validity deals with the ability to infer a causal relationship between treatment and outcome. As the goal models were largely created by the same persons that created the extension, a certain degree of bias cannot be dismissed completely. However, all goal models were frequently reviewed by industry professionals not involved in the development of the extension.

Due to being part of the CrESt-project, the timing of the workshops, data collection procedures, etc., were not completely under our control, but we made use of the means the project setup provided. Nevertheless, there was always sufficient space for industry feedback either in writing as comments to the models or during discussions. While we gave all participants the opportunity to give their opinion publicly or privately, we cannot rule out that some participants might have kept their opinions to themselves.

6.2.4 Construct validity

Construct validity deals with the generalizability of the results found for the particular case example to the underlying theory. I.e. in our case it must be questioned whether the case studies are indeed good representatives for collaborative CPS and whether the effects observed during application can be attributed to the proposed extension or whether these are only particular to the case example. Hence, there is a risk as the requirements for the iStar extension were based on findings from the evaluation case examples, that the proposed extension does only address specific issues for the two case examples under investigation, but that these are not representative for collaborative CPS at large.

One further aspect is the generalization beyond the use for specifying collaborative CPS. Therefore, it is to note that some of the modeling elements we use are not specific for collaborative CPS. Furthermore, we make also use of other proposed extensions that aimed at other system types. Consequently, we cannot state that the proposed extension is limited to the specification of collaborative CPS, nor can we state that there will be no collaborative CPS that cannot be modeled using our extension. However, the applicability on two case examples of collaborative CPS indicates that the proposed extension allows modeling collaborative CPS. Nevertheless, we assume that also other system types might be documented using the extension, particularly those we have briefly sketched in the related work section. However, we cannot make any reliable claim on this as this was not in the focus of our evaluation.

6.3 Inferences

In this paper, we have proposed a GRL-compliant iStar extension to support goal modeling of collaborative CPS that partake in dynamic CPS networks. The proposed modeling elements have been created based on needs identified in industrial applications of goal modeling and have been evaluated for their ability to solve these needs. In addition, the resulting overall goal models have shown valid, useful, and concise. Hence, we can state that the proposed extension is an adequate solution to an industrial problem situation. However, it must be questioned whether goal models in general are a valid approach for supporting the engineering of collaborative CPS. Particularly, for collaborative CPS it is the case that goal modeling is seen as an intuitive approach as it can be expressed that the individual CPS have their own goals to fulfill, which might be contradictory from one system to another as well as the overall goals of the CPS network. Insights derived from the workshops conducted with industry partners corroborate this claim. It was seen as very valuable to identify collaborative CPS and CPS network goals right from the beginning and already discuss dependencies and conflicts arising from the interplay of the individual CPS. Particularly, for the CTR case example it was confirmed that initial conceptual goal models can support the overall development as the industry partner involved follows a goal-oriented implementation approach. I.e. the defined goals are each instantiated by code and deployed on the robot. Additionally, key performance indicators (KPI) are defined to allow monitoring of the goal fulfillment of each goal and decision-making which goal fulfillment should be optimized in which situation.

Although the approach was only evaluated using two case examples, they have shown that the proposed extension is a valid and valuable solution at least for these. However, as the case examples were taken from different domains and the results were also discussed with partners working on other collaborative CPS and also stem from other domains, we are confident that the approach can be a valuable contribution in general. Particularly, the application of goal modeling for supporting the engineering of collaborative CPS seems very reasonable as discussing goal conflicts between individual collaborative CPS as well as between individual collaborative CPS and CPS network is vital for the engineering of these systems. Thus, the use of goal models can improve the engineering of these systems already in the early stages and – as, for instance, the application to the CTR case has shown – can also be used to structure the engineering process of these systems.

6.4 Future work

So far, we have identified limitations and needs for future work regarding the extension and its evaluation, which we will summarize in this section. As discussed in Sect. 5.4.1 some limitations to the proposed GRL-compliant iStar extension still exist. These lead to the need for further improvements to the extension as discussed in Sect. 5.4.2. In addition, we have discussed limitations originating from the threats to validity of the evaluation as outlined in Sect. 6.2, which have been identified as needs for future evaluation efforts in Sect. 6.3.

Thus, two major research directions exist that need to be coped with in future work:

- *Extending and improving the proposed GRL-compliant iStar extension for collaborative CPS.* Most notably there still exists a need for a formal definition of semantics to allow for automated analyses and reasoning about goal fulfillment relations. In addition, industry needs exist regarding the documentation of contribution links with values depending on the different configurations as well as extended tool support.
- *Extending the evaluation of the proposed GRL-compliant iStar extension for collaborative CPS.* In the evaluation of the proposed extension, we have shown that the extension can be used to adequately model the goals for the two selected industry case examples. We have further shown that industry professionals regard the extension as helpful. Beside the need for further evaluation using different case examples, it is also of interest to study the use of the extension by industry professionals not only as

interpreters of the models but their ability to create goal models using the extension themselves.

7 Conclusion

In this paper, we have presented a GRL-compliant iStar extension for collaborative CPS. Collaborative CPS form CPS networks in which they can achieve goals that cannot be achieved by individual CPS on their own [22]. In previous work we have investigated how suitable GRL/iStar is to model such collaborative CPS that form CPS networks [23]. We found that goal modeling – particularly using GRL – is a promising approach to specify collaborative CPS and analyze the interdependencies between the individual CPS and the CPS network. However, we also found that some specific characteristics of collaborative CPS and CPS networks are not sufficiently covered by the iStar modeling language so far. Therefore, in this paper we defined requirements for extending GRL/iStar to allow for consideration of these aspects. Based on these requirements, we have developed a GRL-compliant iStar extension and shown its applicability and usefulness by employing two industrial case examples. We used a cooperative adaptive cruise control system that dynamically forms platoons at runtime from the automotive industry and autonomous transport robots that form fleets of robots to fulfill transportation tasks in smart factories from the industry automation domain.

While we have shown the applicability of the approach to industrial case examples and made the case for its usefulness as seen by industry partners, we have also identified remaining challenges for future work. In this paper we focused on defining an appropriate extension to foster graphical modeling in the development of collaborative CPS. This means that we mainly addressed communication aspects, support for early comprehension and representation of complex relations within the CPS network, and manual analyses of goal relations. This was well-received by industry partners and has been shown to be applicable and useful for collaborative CPS and CPS networks. Thus, we believe this extension is a good starting point for further advanced analysis techniques to support requirements engineering of collaborative CPS and CPS networks. This is substantiated by the discovered desire for automated support in analyzing goal fulfillment relations and for identifying and in-depth analysis of concrete potentially hazardous instance-level configurations. Therefore, in the next step, a thorough definition of goal fulfillment semantics is needed. These must also consider challenging model elements such as circular and bidirectional dependencies or contributions whose value depends on the respective configuration.

Acknowledgments We thank Elham Mirzaei, Martin Neumann, and Jan Stefan Zernickel from InSystems Automation GmbH; Jochen Nickles and Markus Sauer from Siemens AG; Frank Houdek from Daimler AG; and Sebastian Schröck and Peter Heidl from Robert Bosch GmbH for their support. Furthermore, we like to thank the anonymous reviewers that provided us with very helpful feedback and comments, thus supporting us in improving this paper.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was partially funded by the German Federal Ministry for Education and Research (BMBF) under grant no. 01IS16043V

References

1. van Lamsweerde A, Letier E (2004) From object orientation to goal orientation: a paradigm shift for requirements engineering. In: Wirsing M, Knapp A, Balsamo S (eds) Radical innovations of software and systems engineering in the future. Springer, Berlin Heidelberg, pp 325–340
2. Bresciani P, Perini A, Giorgini P et al (2004) Tropos: an agent-oriented software development methodology. Auton Agents Multi-Agent Syst 8:203–236. https://doi.org/10.1023/B:AGNT.0000018806.20944.ef
3. Ali R, Dalpiaz F, Giorgini P (2010) A goal-based framework for contextual requirements modeling and analysis. Requir Eng 15:439–458. https://doi.org/10.1007/s00766-010-0110-z
4. Cheng BHC, Sawyer P, Bencomo N, Whittle J (2009) A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: Schürr A, Selic B (eds) Model driven engineering languages and systems. Springer, Berlin Heidelberg, pp 468–483
5. Mylopoulos J, Chung L, Yu E (1999) From object-oriented to goal-oriented requirements analysis. Commun ACM 42:31–37. https://doi.org/10.1145/291469.293165
6. Ghanavati S, Rifaut A, Dubois E, Amyot D (2014) Goal-oriented compliance with multiple regulations. In: 2014 IEEE 22nd international requirements engineering conference (RE). pp 73–82
7. Grau G, Franch X, Maiden NAM (2008) PRiM: An i*-based process reengineering method for information systems specification. Inform Softw Technol 50:76–100. https://doi.org/10.1016/j.infsof.2007.10.006
8. Cardoso ECS, Almeida JPA, Guizzardi G, Guizzardi RSS (2009) Eliciting goals for business process models with non-functional requirements catalogues. In: Halpin T, Krogstie J, Nurcan S et al (eds) Enterprise, business-process and information systems modeling. Springer, Berlin, Heidelberg, pp 33–45
9. Horkoff J, Yu E (2016) Interactive goal model analysis for early requirements engineering. Requir Eng 21:29–61. https://doi.org/10.1007/s00766-014-0209-8

10. Cleland-Huang J, Settimi R, BenKhadra O et al (2005) Goal-centric traceability for managing non-functional requirements. In: Proceedings of the 27th international conference on software engineering. ACM, New York, NY, USA, pp 362–371
11. Kavakli V, Loucopoulos P (1999) Goal-driven business process analysis application in electricity deregulation. Inf Syst 24:187–207. https://doi.org/10.1016/S0306-4379(99)00015-0
12. Liaskos S, Alexei, Yu Y et al (2006) On goal-based variability acquisition and analysis. In: 14th IEEE international requirements engineering conference (RE'06). pp 79–88
13. Yijun Yu, Leite JCSP, Mylopoulos J (2004) From goals to aspects: discovering aspects from requirements goal models. In: Proceedings. 12th IEEE international requirements engineering conference, 2004. pp 38–47
14. van Lamsweerde A, Darimont R, Letier E (1998) Managing conflicts in goal-driven requirements engineering. IEEE Trans Softw Eng 24:908–926. https://doi.org/10.1109/32.730542
15. Fuxman A, Liu L, Mylopoulos J et al (2004) Specifying and analyzing early requirements in Tropos. Requir Eng 9:132–150. https://doi.org/10.1007/s00766-004-0191-7
16. Matulevičius R, Mayer N, Mouratidis H et al (2008) Adapting secure tropos for security risk management in the early phases of information systems development. In: Bellahsène Z, Léonard M (eds) Advanced information systems engineering. Springer, Berlin, Heidelberg, pp 541–555
17. Mouratidis H, Giorgini P (2007) Secure tropos: a security-oriented extension of the tropos methodology. Int J Soft Eng Knowl Eng 17:285–309. https://doi.org/10.1142/S0218194007003240
18. Rolland C, Souveyet C, Achour CB (1998) Guiding goal modeling using scenarios. IEEE Trans Softw Eng 24:1055–1071. https://doi.org/10.1109/32.738339
19. Goldsby HJ, Sawyer P, Bencomo N et al (2008) Goal-based modeling of dynamically adaptive system requirements. In: 15th Annual IEEE international conference and workshop on the engineering of computer based systems (ecbs 2008). pp 36–45
20. Andersson B, Johannesson P, Zdravkovic J (2009) Aligning goals and services through goal and business modelling. Inf Syst E-Bus Manag 7:143–169. https://doi.org/10.1007/s10257-008-0084-2
21. Dalpiaz F, Franch X, Horkoff J (2016) iStar 2.0 language guide. https://arxiv.org/abs/1605.07767 [cs]
22. Mosterman PJ, Zander J (2016) Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. Softw Syst Model 15:5–16. https://doi.org/10.1007/s10270-015-0469-x
23. Daun M, Stenkova V, Krajinski L et al (2019) 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. pp 1600–1609
24. Yu ESK (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of ISRE '97: 3rd IEEE international symposium on requirements engineering. pp 226–235
25. International Telecommunication Union (2018) Recommendation Z.151 (10/18): user requirements notation (URN)—language definition. International Telecommunication Union, Geneva, Switzerland
26. Mavin A, Wilkinson P, Teufl S et al (2017) Does goal-oriented requirements engineering achieve its goal? In: 2017 IEEE 25th international requirements engineering conference (RE). pp 174–183
27. Wagner S, Fernández DM, Felderer M et al (2019) Status quo in requirements engineering: a theory and a global family of surveys. ACM Trans Softw Eng Methodol 28:9:1-9:48. https://doi.org/10.1145/3306607
28. Horkoff J, Aydemir FB, Cardoso E et al (2019) Goal-oriented requirements engineering: an extended systematic mapping study. Requir Eng 24:133–160. https://doi.org/10.1007/s00766-017-0280-z
29. Kavakli E (2004) Modeling organizational goals: analysis of current methods. In: Proceedings of the 2004 ACM symposium on applied computing. ACM, New York, NY, USA, pp 1339–1343
30. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20:3–50. https://doi.org/10.1016/0167-6423(93)90021-G
31. van Lamsweerde A (2009) Requirements engineering: from system goals to UML models to software specifications, 1st edn. Wiley, Hoboken
32. Amyot D, Horkoff J, Gross D, Mussbacher G (2009) A lightweight GRL profile for i* modeling. In: Heuser CA, Pernul G (eds) Advances in conceptual modelling—challenging perspectives. Springer, Berlin Heidelberg, pp 254–264
33. Amyot D, Mussbacher G (2011) User requirements notation: the first ten years, the next ten years. JSW 6:747–768. https://doi.org/10.4304/jsw.6.5.747-768
34. Horkoff J, Elahi G, Abdulhadi S, Yu E (2008) Reflective analysis of the syntax and semantics of the i* framework. In: Song I-Y, Piattini M, Chen Y-PP et al (eds) Advances in conceptual modeling—challenges and opportunities. Springer, Berlin Heidelberg, pp 249–260
35. Brings J, Daun M, Bandyszak T et al (2019) Model-based documentation of dynamicity constraints for collaborative cyber-physical system architectures: findings from an industrial case study. J Syst Archit 97:153–167. https://doi.org/10.1016/j.sysarc.2019.02.012
36. Teruel MA, Navarro E, López-Jaquero V et al (2011) CSRML: a goal-oriented approach to model requirements for collaborative systems. In: Jeusfeld M, Delcambre L, Ling T-W (eds) Conceptual modeling—ER 2011. Springer, Berlin, Heidelberg, pp 33–46
37. Kim KD, Kumar PR (2012) Cyber-physical systems: a perspective at the centennial. Proc IEEE 100:1287–1308. https://doi.org/10.1109/JPROC.2012.2189792
38. Fitzgerald J, Larsen PG, Verhoef M (2014) From embedded to cyber-physical systems: challenges and future directions. In: Fitzgerald J, Larsen PG, Verhoef M (eds) Collaborative design for embedded systems. Springer, Berlin, Heidelberg, pp 293–303
39. Lee EA (2008) Cyber physical systems: design challenges. In: 2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC). pp 363–369
40. Stankovic JA, Lee I, Mok A, Rajkumar R (2005) Opportunities and obligations for physical computing systems. Computer 38:23–31. https://doi.org/10.1109/MC.2005.386
41. Ponsard C, Massonet P, Rifaut A et al (2005) Early verification and validation of mission critical systems. Electron Notes Theor Comput Sci 133:237–254. https://doi.org/10.1016/j.entcs.2004.08.067
42. Fallah YP, Huang C, Sengupta R, Krishnan H (2010) Design of cooperative vehicle safety systems based on tight coupling of communication, computing and physical vehicle dynamics. In: Proceedings of the 1st ACM/IEEE international conference on cyber-physical systems. ACM, New York, NY, USA, pp 159–167
43. Fallah YP, Huang C, Sengupta R, Krishnan H (2011) Analysis of information dissemination in vehicular ad-hoc networks with application to cooperative vehicle safety systems. IEEE Trans Veh Technol 60:233–247. https://doi.org/10.1109/TVT.2010.2085022

44. Sha L, Gopalakrishnan S, Liu X, Wang Q (2008) Cyber-physical systems: a new frontier. In: 2008 IEEE international conference on sensor networks, ubiquitous, and trustworthy computing (sutc 2008). pp 1–9
45. Lee EA (2010) CPS foundations. In: Design automation conference. pp 737–742
46. Gonçalves E, de Oliveira MA, Monteiro I et al (2019) Understanding what is important in iStar extension proposals: the viewpoint of researchers. Requir Eng 24:55–84. https://doi.org/10.1007/s00766-018-0302-5
47. Moody D (2009) The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans Softw Eng 35:756–779. https://doi.org/10.1109/TSE.2009.67
48. Brings J, Daun M, Weyer T, Pohl K (2020) Goal-based configuration analysis for networks of collaborative cyber-physical systems. In: Proceedings of the 35th annual ACM symposium on applied computing. Association for Computing Machinery, Brno, Czech Republic, pp 1387–1396
49. Daun M, Salmon A, Tenbergen B et al (2014) Industrial case studies in graduate requirements engineering courses: The impact on student motivation. In: Bollin A, Hochmüller E, Mittermeir RT et al (eds) 27th IEEE conference on software engineering education and training, CSEE&T 2014, Klagenfurt, Austria, April 23–25, 2014. IEEE, pp 3–12
50. Tenbergen B, Daun M (2019) Industry Projects in Requirements Engineering Education: Application in a University Course in the US and Comparison with Germany. In: 52nd Hawaii International Conference on System Sciences
51. Daun M, Brings J, Obe PA et al (2017) Teaching conceptual modeling in online courses: coping with the need for individual feedback to modeling exercises. In: Washizaki H, Mead N (eds) 30th IEEE conference on software engineering education and training, CSEE&T 2017, Savannah, GA, USA, November 7–9, 2017. IEEE, pp 134–143
52. Daun M, Keller K, Brings J (2017) Teaching goal modeling to engineering professionals—an experience report. In: Franch X, Snoeck M, Guizzardi RSS, Jureta I (eds) Proceedings of the 5th symposium on conceptual modeling education and the 2nd international iStar teaching workshop co-located with the 36th international conference on conceptual modeling (ER 2017), Valencia, Spain, November 6–9, 2017. CEUR-WS.org, pp 38–47
53. Lewis GA, Morris E, Place P et al (2009) Requirements engineering for systems of systems. In: 2009 3rd annual IEEE systems conference. pp 247–252
54. Kopetz H, Bondavalli A, Brancati F et al (2016) Emergence in cyber-physical systems-of-systems (CPSoSs). In: Bondavalli A, Bouchenak S, Kopetz H (eds) Cyber-physical systems of systems. Springer, Cham, pp 73–96
55. Cavalcante E, Batista T, Bencomo N, Sawyer P (2015) revisiting goal-oriented models for self-aware systems-of-systems. In: 2015 IEEE international conference on autonomic computing. pp 231–234
56. Garro A, Tundis A (2015) On the reliability analysis of systems and SoS: the RAMSAS method and related extensions. IEEE Syst J 9:232–241. https://doi.org/10.1109/JSYST.2014.2321617
57. Silva E, Cavalcante E, Batista T et al (2014) On the characterization of missions of systems-of-systems. In: Proceedings of the 2014 European conference on software architecture workshops. ACM, New York, NY, USA, pp 26:1–26:8
58. Silva E, Batista T, Cavalcante E (2015) A mission-oriented tool for system-of-systems modeling. In: Proceedings of the third international workshop on software engineering for systems-of-systems. IEEE Press, Piscataway, NJ, USA, pp 31–36
59. Silva E, Batista T, Oquendo F (2015) A mission-oriented approach for designing system-of-systems. In: 2015 10th system of systems engineering conference (SoSE). pp 346–351
60. Silva E, Batista T (2018) Formal modeling systems-of-systems missions with mKAOS. In: Proceedings of the 33rd annual ACM symposium on applied computing. ACM, New York, NY, USA, pp 1674–1679
61. Garcés L, Nakagawa EY (2017) A process to establish, model and validate missions of systems-of-systems in reference architectures. In: Proceedings of the symposium on applied computing. ACM, New York, NY, USA, pp 1765–1772
62. Rogers A, Ramchurn SD, Jennings NR (2012) Delivering the smart grid: challenges for autonomous agents and multi-agent systems research. In: Proceedings of the twenty-sixth AAAI conference on artificial intelligence. pp 2166–2172
63. Wooldridge M (1997) Agent-based software engineering. IEE Proc Softw Eng 144:26–37. https://doi.org/10.1049/ip-sen:19971026
64. Rao AS, Georgeff MP (1991) Modeling rational agents within a BDI-architecture. In: Proceedings of the second international conference on principles of knowledge representation and reasoning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 473–484
65. Rao AS, Georgeff MP (1995) BDI Agents: from theory to practice. In: Proceedings of the first international conference on multi-agent systems (ICMAS-95), San Francisco. pp 312–319
66. Grau G, Cares C, Franch X, Navarrete FJ (2006) A comparative analysis of i*agent-oriented modelling techniques. In: Proceedings of the eighteenth international conference on software engineering and knowledge engineering (SEKE'06). pp 1–7
67. Vrbaski M, Mussbacher G, Petriu D, Amyot D (2012) Goal models as run-time entities in context-aware systems. In: Proceedings of the 7th workshop on Models@Run.Time. ACM, New York, NY, USA, pp 3–8
68. Bergenti F, Rimassa G, Somacher M, Botelho LM (2003) A FIPA compliant goal delegation protocol. In: Huget M-P (ed) Communication in multiagent systems: agent communication languages and conversation policies. Springer, Berlin, Heidelberg, pp 223–238
69. Braubach L, Pokahr A, Moldt D, Lamersdorf W (2005) Goal representation for BDI agent systems. In: Bordini RH, Dastani M, Dix J, El Seghrouchni Fallah A (eds) Programming multi-agent systems. Springer, Berlin Heidelberg, pp 44–65
70. Partsakoulakis I, Vouros G (2002) Roles in collaborative activity. In: Vlahavas IP, Spyropoulos CD (eds) Methods and applications of artificial intelligence. Springer, Berlin, Heidelberg, pp 449–460
71. Vally J-D, Courdier R (1998) A conceptual "role-centered" model for design of multi-agents systems. In: Ishida T (ed) Multiagent platforms. Springer, Berlin, Heidelberg, pp 33–46
72. Wooldridge M, Jennings NR, Kinny D (1999) A methodology for agent-oriented analysis and design. In: Proceedings of the third annual conference on autonomous agents. ACM, New York, NY, USA, pp 69–76
73. Kendall EA (2000) Role modeling for agent system analysis, design, and implementation. IEEE Concurr 8:34–41. https://doi.org/10.1109/4434.846192
74. Kinny D, Georgeff M, and Rao A (1996) A methodology and modelling technique for systems of BDI agents. In: Van de Velde W, Perram JW (eds) Agents Breaking Away, pp. 56–71
75. Odell J, Nodine M, Levy R (2004) A metamodel for agents, roles, and groups. In: Agent-oriented software engineering V. Springer, Berlin, Heidelberg, pp 78–92

76. Beydoun G, Low G, Henderson-Sellers B et al (2009) FAML: a generic metamodel for MAS development. IEEE Trans Softw Eng 35:841–863. https://doi.org/10.1109/TSE.2009.34
77. Adam E, Strugeon EGL, Mandiau R (2008) Flexible hierarchical organisation of role based agents. In: 2008 Second IEEE international conference on self-adaptive and self-organizing systems workshops. pp 186–191
78. Adam E, Mandiau R (2007) Flexible roles in a holonic multi-agent system. In: Mařík V, Vyatkin V, Colombo AW (eds) Holonic and multi-agent systems for manufacturing. Springer, Berlin, Heidelberg, pp 59–70
79. Giorgini P, Mylopoulos J, Sebastiani R (2005) Goal-oriented requirements analysis and reasoning in the Tropos methodology. Eng Appl Artif Intell 18:159–171. https://doi.org/10.1016/j.engappai.2004.11.017
80. Zhong C, DeLoach SA (2011) Runtime models for automatic reorganization of multi-robot systems. In: Proceedings of the 6th international symposium on software engineering for adaptive and self-managing systems. ACM, New York, NY, USA, pp 20–29
81. Thangarajah J, Padgham L, Winikoff M (2003) Detecting and exploiting positive goal interaction in intelligent agents. In: Proceedings of the second international joint conference on autonomous agents and multiagent systems. ACM, New York, NY, USA, pp 401–408
82. Cheong C, Winikoff M (2005) Hermes: implementing goal-oriented agent interactions. In: Programming multi-agent systems. Springer, Berlin, Heidelberg, pp 168–183
83. Cheong C, Winikoff M (2005) Hermes: designing goal-oriented agent interactions. In: Müller JP, Zambonelli F (eds) Agent-oriented software engineering VI. Springer, Berlin, Heidelberg, pp 16–27
84. Cheong C, Winikoff M (2005) Hermes: a methodology for goal oriented agent interactions. In: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems. ACM, New York, NY, USA, pp 1121–1122
85. Gonçalves E, Castro J, Araújo J, Heineck T (2018) A systematic literature review of iStar extensions. J Syst Softw 137:1–33. https://doi.org/10.1016/j.jss.2017.11.023
86. Teruel MA, Tardío R, Navarro E et al (2014) CSRML4BI: a goal-oriented requirements approach for collaborative business intelligence. In: Yu E, Dobbie G, Jarke M, Purao S (eds) Conceptual modeling. Springer, Cham, pp 423–430
87. Teruel MA, Navarro E, López-Jaquero V et al (2017) A comprehensive framework for modeling requirements of CSCW systems. J Softw Evol Process 29:e1858. https://doi.org/10.1002/smr.1858
88. Ellis CA, Gibbs SJ, Rein G (1991) Groupware: some issues and experiences. Commun ACM 34:39–58. https://doi.org/10.1145/99977.99987
89. Ali R, Dalpiaz F, Giorgini P (2014) Requirements-driven deployment. Softw Syst Model 13:433–456. https://doi.org/10.1007/s10270-012-0255-y
90. Silva C, Borba C, Castro J (2011) A goal oriented approach to identify and configure feature models for software product lines. WER
91. Borba C, Silva C (2009) A Comparison of goal-oriented approaches to model software product lines variability. In: Heuser CA, Pernul G (eds) Advances in conceptual modelling—challenging perspectives. Springer, Berlin, Heidelberg, pp 244–253
92. Guzman A, Martínez Rebollar A, Vargas F et al (2016) A methodology for modeling Ambient Intelligence applications using i* framework. In: iStar 2016 ninth international i* workshop 1674:61–66
93. Marosin D, Ghanavati S (2017) Principle-based goal-oriented requirements language. In: Proper HA, Winter R, Aier S, de Kinderen S (eds) Architectural coordination of enterprise transformation. Springer, Cham, pp 235–247
94. Gailly F, España S, Poels G, Pastor O (2008) Integrating business domain ontologies with early requirements modelling. In: Song I-Y, Piattini M, Chen Y-PP et al (eds) Advances in conceptual modeling—challenges and opportunities. Springer, Berlin, Heidelberg, pp 282–291
95. van Arem B, van Driel CJG, Visser R (2006) The impact of cooperative adaptive cruise control on traffic-flow characteristics. IEEE Trans Intell Transp Syst 7:429–436. https://doi.org/10.1109/TITS.2006.884615
96. Han S-Y, Chen Y-H, Wang L, Abraham A (2013) Decentralized longitudinal tracking control for cooperative adaptive cruise control systems in a platoon. In: 2013 IEEE international conference on systems, man, and cybernetics. IEEE, Manchester, pp 2013–2018
97. Salehie M, Tahvildari L (2009) Self-adaptive software: landscape and research challenges. ACM Trans Auton Adapt Syst 4:14:1-14:42. https://doi.org/10.1145/1516533.1516538
98. OMG (2014) Object constraint language. OMG
99. Hölldobler K, Roth A, Rumpe B, Wortmann A (2017) Advances in modeling language engineering. In: Ouhammou Y, Ivanovic M, Abelló A, Bellatreche L (eds) Model and data engineering. Springer, Cham, pp 3–17
100. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14:131. https://doi.org/10.1007/s10664-008-9102-8
101. Daun M, Brings J, Obe PA et al (2019) Using view-based architecture descriptions to aid in automated runtime planning for a smart factory. In: IEEE international conference on software architecture companion, ICSA Companion 2019, Hamburg, Germany, March 25–26, 2019. IEEE, pp 202–209
102. Bandyszak T, Daun M, Tenbergen B et al (2020) Orthogonal uncertainty modeling in the engineering of cyber-physical systems. IEEE Trans Autom Sci Eng. https://doi.org/10.1109/TASE.2020.2980726
103. Bhatt RM, Tang CP, Krovi VN (2009) Formation optimization for a fleet of wheeled mobile robots—a geometric approach. Robot Auton Syst 57:102–120. https://doi.org/10.1016/j.robot.2006.12.012
104. Schlingloff B-H (2018) Specification and verification of collaborative transport robots. In: 2018 4th international workshop on emerging ideas and trends in the engineering of cyber-physical systems (EITEC). IEEE, Porto, pp 3–8
105. Stenkova V, Brings J, Daun M, Weyer T (2019) Generic negative scenarios for the specification of collaborative cyber-physical systems. In: Conceptual modeling—38th international conference, ER 2019, proceedings. Springer, p in press
106. Daun M, Brings J, Weyer T (2020) Do instance-level review diagrams support validation processes of cyber-physical system specifications: results from a controlled experiment. In: Proceedings of the international conference on software and system processes, ICSSP 2020, Seoul, Republic of Korea. IEEE/ACM
107. Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R (2003) Formal reasoning techniques for goal models. In: Spaccapietra S, March S, Aberer K (eds) Journal on data semantics I. Springer, Berlin Heidelberg, pp 1–20
108. Amyot D, Ghanavati S, Horkoff J et al (2010) Evaluating goal models within the goal-oriented requirement language. Int J Intell Syst 25:841–877. https://doi.org/10.1002/int.20433
109. Lockerbie J, Maiden NAM, Engmann J et al (2012) Exploring the impact of software requirements on system-wide goals: a method

using satisfaction arguments and i* goal modelling. Requir Eng 17:227–254. https://doi.org/10.1007/s00766-011-0138-8

110. Runeson P, Höst M, Rainer A, Regnell B (2012) Case study research in software engineering: guidelines and examples. Wiley, Hoboken

111. Yin RK (2018) Case study research and applications: design and methods, 6th edn. Sage Publications Ltd., Los Angeles

112. Wohlin C, Runeson P, Höst M et al (2012) Experimentation in software engineering, 2012th edn. Springer, New York