Chapter 2
Definition of Cooperating Objects

The term "Cooperating Objects" was coined by the Embedded WiSeNts Research Roadmap [1] in November 2006. The Cooperating Objects Network of Excellence CONET used the same definition in the "Research Roadmap on Cooperating Objects" [2]. For the second edition of the roadmap "The emerging domain of Cooperating Objects" [3] the definition was revised to emphasise the cooperative aspects. We develop this definition further to improve its clarity as follows:

> Cooperating Objects are modular systems of autonomous, heterogeneous devices pursuing a common goal by cooperation in computations and in sensing and/or actuating with the environment.

As explained in the previous chapter, the domain of Cooperating Objects is a cross-section between (networked) embedded systems, ubiquitous computing and (wireless) sensor networks. There are, therefore, several flavours of Cooperating Objects depending on the degree in which they fulfill different Cooperating Object features. Some of them can process the context of cooperation intentionally, act on it and intentionally extend it, change it or stop it. As such they may possess the necessary logic to understand semantics and build complex behaviours, thus allowing the Cooperating Object to be part of a dynamic complex ecosystem.

As depicted in Fig. 2.1, there are several areas that share common ground with Cooperating Objects e.g. software agents, Internet of Things, Cyber-Physical systems etc. However what differentiates them is the mix of the degree of physical and feature elements that creates the right recipe for a specific area. For instance Cooperating Objects focuses mostly on the cooperation aspects while considering the rest of them only as enabling factors to achieve cooperation. Other approaches e.g. Internet of Things, focus mostly on the interaction and integration part while cooperation is optional. So the differentiating factor among all areas, is not the distinct characteristics but which of them they employ (depending on the scenario) and at which degree.

In the following sections each of the key features of this definition is discussed in detail.

P. J. Marrón et al., *The Emerging Domain of Cooperating Objects*,
SpringerBriefs in Cooperating Objects, DOI: 10.1007/978-3-642-28469-4_2,

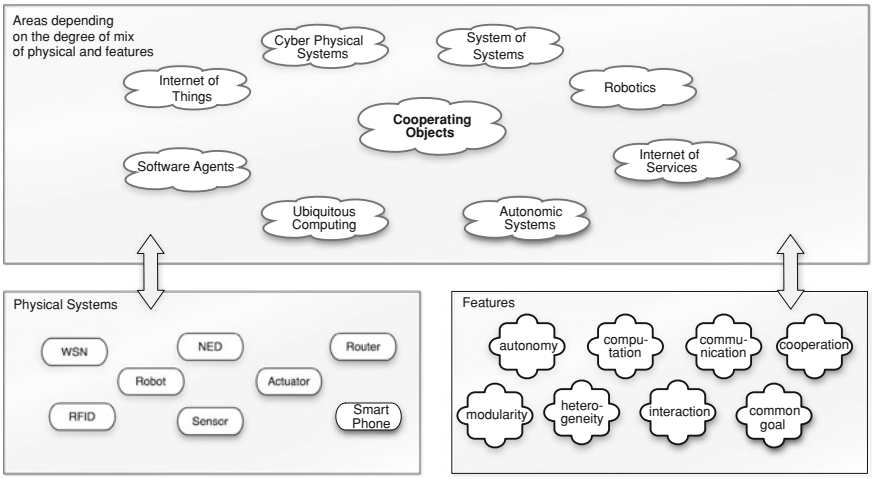

Fig. 2.1 Overview of various areas

2.1 Modularity

A Cooperating Object is composed of several devices that need to exhibit certain features according to the definition. Each of the devices contributes functionality to the overall Cooperating Object, but the modularisation helps to keep the single devices simple and maintainable.

The logical organisation of the single devices inside the Cooperating Object is not restricted by the definition and can range from completely flat to completely hierarchical. In the latter case, another Cooperating Object is included completely in a superordinate Cooperating Object that contains more devices or sibling Cooperating Objects. From the top-level Cooperating Object all participating devices and Cooperating Objects have a similar appearance since there is usually a single device representing the subordinate Cooperating Object to the superordinate one. Thus, in all following explanations a device can also be a complete Cooperating Object unless stated otherwise. The use of hierarchies depends on the type of cooperation but also on the need for scalability for which hierarchies are a proven principle in distributed systems.

Single devices can be exchanged if the replacement provides the same functionality. Thus, modularity is a prerequisite in a dynamic environment where the availability of all devices belonging to the current configuration cannot be guaranteed at all times. In contrast, it is quite common that devices are replaced by other devices during the lifetime of a Cooperating Object.

The modular design makes it also possible to replace a device by a more powerful one or to add new devices that extend the functionality. Thus, the Cooperating Object can be developed in an evolutionary fashion and adapted to new needs.

In contrast, a single device cannot be a Cooperating Object although it might consist of several components. In these monolithic systems the single components are not independent (see Autonomy in Sect. 2.2) and their dynamic exchange is impossible.

As for every modular system, the design of the interface and APIs is a crucial point. In general, the interface should protect the device itself while allowing others to use the provided functionality in a controlled way. Interfaces should adhere to standards if they exist so that the possibility for other matching modules is higher. They should not be tailored to a specific device since this creates a tightly coupled system which might only work in exactly one configuration, leading to an almost monolithic system.

2.2 Autonomy

Each device can decide on its own about its involvement in a Cooperating Object. If the Cooperating Object does not participate at all in the cooperation and coordination activities, it is not considered part of the Cooperating Object. Otherwise, it decides about the degree of participation. In general, a Cooperating Object can dedicate only a fraction of its resources or its functionality to the current Cooperating Object, thus leaving the possibility to serve multiple Cooperating Objects. Note that this is not restricted by the definition.

In contrast, the device can also exclusively "belong" to a single Cooperating Object. This is not a contradiction to the modularity principle as long as the device is not directed to another particular device, for example if they are statically wired, but can decide autonomously with which device it is cooperating. Nevertheless, the device might be designed for a special functionality so that it can only be able to participate with a special type of devices. However, it is still the autonomy of the device to choose with which particular instance of them.

The decision to cooperate or not can be based on various factors, e.g. the current and maximum number of Cooperating Objects a device is able to join, the energy level of the device, past interactions with the other devices requesting cooperation, compatibility of goals, and various application-specific variables. The single factors can be combined in a single cost function to evaluate the willingness to cooperate in an easier fashion. Different flavours of this willingness can exist: a device may simply be providing its information to all other devices that want to have it, i.e. this "helper device" always cooperates. On the other hand, a device can evaluate if it will benefit from cooperating, for example because the other devices could provide functionality in turn that it needs for its own tasks. Other incentive mechanisms can improve this mutual willingness to cooperate. Another possibility is that the requested task is compatible to an already executed task on the device and, thus, the cooperative work can be performed without or with only very little additional effort.

Autonomy also implies that there is no master that assigns the membership to a Cooperating Object or the functionality inside it. However, it is possible that the

devices of a Cooperating Object autonomously select a (temporary) master that configures the Cooperating Object. Usually, this master property circulates over time through the devices.

2.3 Heterogeneity

In the definition of Cooperating Objects, heterogeneity is a crucial point since it is more than heterogeneity in terms of, e.g. processing power or memory. In fact, a Cooperating Object must combine devices of different system concepts, i.e. Wireless Sensor Networks, embedded systems, robotics, etc. Since devices belonging to these different concepts often have different hardware characteristics the heterogeneity is also exhibited in this respect, but it is a consequence and not the actual meaning of "heterogeneity" in this definition.

Taking into account this meaning, it is obvious that a Cooperating Object is more than one of the system concepts presented in Chap. 1 that constitute the basis of Cooperating Objects. The novelty and at the same time the challenges of Cooperating Objects arise from the demand to strengthen the underdeveloped functional aspects of one the system concepts by combining it with other concepts without loosing the strengths.

Of course, the research challenges of the separate system concepts do not disappear merely by combining them, but they should be tackled in their areas. Cooperating Objects research should instead concentrate on the new challenges arising from the combination. However, there will be an overlap since problems can be equal for Cooperating Objects and an underlying system concept.

2.4 Computation

The majority of Cooperating Objects are computing devices cooperating with each other. (Electro-)Mechanical parts can be connected to these devices but do not count as extra devices (see Autonomy in Sect. 2.2). Due to the different nature of the single devices in a Cooperating Object (see Heterogeneity in Sect. 2.3) the computational capabilities can vary largely. However, a device must at least be able to take an autonomous decision about its involvement in a Cooperating Object and to communicate with other devices, which usually requires also computation.

Local computation can also be used to reduce the need for or the amount of communication, e.g. by compression or filtering, which can be a goal in energy-restricted scenarios. On the other hand, Cooperating Objects with less powerful devices have to distribute the computational workload onto several devices, which is an important self-configuration task.

2.5 Interaction with Environment

Cooperating Objects interact with the environment using sensors and/or actuators. Sensors convert physical or chemical quantities into a signal which is read by the connected computational device, and actuators transform a signal from this device into mechanical motion or physical quantities. It strongly depends on the application which sensors and actuators are involved. It can be as simple as a thermistor or an LED, but can also e.g. consist of an external digital signal processor or a machine controller. Nevertheless, the interaction should be substantial, especially with respect to actuators, i.e. actuation should have a changing effect on the environment.

The involvement of sensors and actuators makes Cooperating Objects real-world objects, i.e. there are no pure virtual Cooperating Objects. Certainly, not every device needs to have sensors or actuators, i.e. there can be computation-only devices, but sensing and/or actuation must be performed by some devices of the Cooperating Objects and this interaction with the environment must be a core functionality of the Cooperating Object and not just an optional side-effect.

2.6 Communication

Interestingly, the first of the five pragmatic axioms on human communication that Paul Watzlawick formulated in 1967 is also valid for Cooperating Objects: “one cannot not communicate” [4]. If there is no observable behaviour of a device at all, it implicitly denies all cooperation—be it by an autonomous decision or simply because the device failed. After all, communication is a requirement for cooperation, although not explicitly mentioned in the definition of Cooperating Objects.

If a device communicates there are three techniques of information exchange [5]: the most obvious technique is explicit communication, which can be performed using various means, e.g. wires, radio, light, sound. The content of the communication is manifold and can range from just the state of the single device to a common planning. Cooperating Objects do not restrict the possible communication patterns since it might still be necessary to communicate with a single device, all devices or a certain group of devices.

Besides explicit communication, there are two other techniques that work by observation using sensors. With passive action recognition the actions of other devices are observed, e.g. if an actuator moves. In contrast, the effects of actions of others can be sensed (“stigmergy”), e.g. the increase of temperature caused by a heater. Usually, these forms of communication show the lack of common interfaces for direct communication; nevertheless, the inclusion of such devices allows for interesting applications.

2.7 Common Goal

The ultimate reason for a Cooperating Object to exist is the common goal it tries to achieve. There should be a reason for pursuing the goal using Cooperating Objects: either the goal can only be achieved through Cooperating Objects or there is at least an improvement compared to a monolithic or centralised approach.

The form of goal definition is not restricted. Automated planning methods could be applied when the initial state is derived from the context and the possible actions are gathered from all available devices. However, due to the complexity of these planning algorithms, the goal is normally not explicitly stated at all, but represented as an application that describes also the required external interfaces. By finding devices that run the application and provide the needed interfaces the Cooperating Object is built up and the tasks are distributed among the devices. This task distribution has to take into account that some devices can execute certain tasks at all or in a more efficient way than other devices due to heterogeneity. However, it is the eventual result of the autonomous decision of each device.

If a device is only the delegate for a subordinate Cooperating Object the described planning and/or application building process for this task resulting in sub-tasks for the devices that are involved in the subordinate Cooperating Object.

Although the devices do not know the overall goal they execute a task to achieve it. Thus, each device has detailed knowledge only about its area of responsibility, but limited information about the whole Cooperating Object. However, the cooperation of the single devices make it possible to achieve the overall goal, which needs the full picture. Thus, the intelligence of the system lies distributed in the network.

Due to the potential long-running nature of Cooperating Object applications it is obvious that goals will change over time. This results in different tasks and, thus, in a reconfiguration of the Cooperating Object.

2.8 Cooperation

"Cooperation" is a widely and often used term, but its meaning differs between areas; and also inside an area there is not necessarily a common agreement. Moreover, the terms "cooperation" and "collaboration" are often used as synonyms. For example, in the Merriam Webster thesaurus [6] both words have the same definition: "the work and activity of a number of persons who individually contribute toward the efficiency of the whole". Therefore, we review some of the definitions before presenting our intention of "cooperation".

The Macmillan Dictionary [7] defines cooperation as "a situation in which people or organisations work together to achieve a result that will benefit all of them" and collaboration as "the process of working with someone to produce something". It can be argued that according to these definitions cooperation has a stronger motivation for common work than collaboration, but this does not hold for most other definitions.

In [8] both terms are defined with respect to agents. For cooperation, agents have defined roles that cannot be changed and, therefore, they must cooperate. The roles are assigned by the designer with a (single) goal in mind. In contrast, dynamic links between agents without predefined roles lead to collaboration. This way, agents can achieve their own goals by negotiating contracts with other agents. If there is a common goal both agents contribute shared effort to this goal. If no common goal can be negotiated but the single goals are well aligned, only resources can be shared in such a way that both agents can benefit. For Cooperating Object we do not adopt this distinction based on the difference between static and dynamic roles.

Three important categories of interactions between (computing) processes are described in [9]: cooperation is anticipated and desired interaction; competition is anticipated and acceptable, but undesirable interaction; and interference is unanticipated or unacceptable. Competition usually happens when access to computing resources needs to be serialised. With real-world objects, competition for resources of the real world occurs as well, e.g. cars driving on a crossing. Interference happens when the actions of unrelated devices collide and reduce the fulfilment of the goal. However, when combining the devices in a Cooperating Object they will coordinate before competition or interference can happen at all, thus making the overall process more efficient. It is also known from economic science that cooperation allows a more efficient use of resources than competition [10].[1]

According to [11], which deals with computer-based learning environments, "[c]ooperative work is accomplished by the division of labour among participants, as an activity where each person is responsible for a portion of the problem solving", i.e. the task is split in independent sub-tasks that can be performed individually. In contrast, collaboration needs "[…] the mutual engagement of participants in a coordinated effort to solve the problem together", i.e. there is no or only partial division of the work but a continued combined activity.

A similar direction is taken by [12], which establishes a hierarchy between coordination, cooperation and collaboration—we only consider the latter two. For cooperation, a mutual benefit should be gained, e.g. savings in time and cost, by sharing or partitioning work. For collaboration, a collective result should be achieved, which would not be possible alone. This result should be innovative, extraordinary or a break-through. A typical collaborative task would be brainstorming, i.e. a creative process.

For Cooperating Objects, we follow this view. According to our definition at the beginning of this chapter, there is a common goal that the devices try to achieve. As described in Sect. 2.7 the devices finally execute individual tasks. Nevertheless, some of these tasks might also need tight coordination due to the limitation of the participants or, for example, when two actuators driven by separate device need to interact. However, this is not comparable to an ongoing, creative, problem solving task as described by the last two definitions.

[1] Cooperation and competition can also be seen as orthogonal instead of opposed. Both span a continuum from weak to strong, and a relationship between two firms can be placed anywhere on this matrix. This phenomenon is called coopetition.

In our sense, cooperation is always intentional and driven by a goal. Without a goal and, thus, no tasks there is no need for cooperation at all. Although unintentional interaction might deliver the same results it does not happen in a controlled way which creates problems in case of errors. For example, reconfiguration is more difficult if the exact task that a device has performed is not known.

The participation of all devices in a Cooperating Object is needed to achieve the common goal, i.e. a Cooperating Object is more than just the sum of the single devices. Nevertheless, the common goal (see Sect. 2.7) does not imply benefits for all the cooperating devices. Some of them can be especially designed to help in cooperation, others can play a more active part in one cooperative task to profit more in another one. When autonomous and selfish objects decide autonomously if and how they cooperate the sum of the benefit must be positive. Otherwise, a device will eventually not agree to cooperate or not be asked to cooperate any more.

References

1. Marrón PJ, Minder D (eds) (2006) Embedded WiSeNts consortium, Embedded WiSeNts research roadmap. Logos, Berlin
2. Marrón PJ, Karnouskos S, Minder D (eds) (2009) Research roadmap on cooperating objects. European Commission, ISBN: 978-92-79-12046-6. Office for Official Publications of the European Communities. doi:10.2759/11566
3. Marrón PJ, Karnouskos S, Minder D, Ollero A (eds) (2011) The emerging domain of cooperating objects. Springer, Berlin. http://www.springer.com/engineering/signals/book/978-3-642-16945-8
4. Watzlawick P, Beavin JH, Jackson DD (1967) Pragmatics of human communication. W. W. Norton, New York
5. Siciliano B, Khatib O (eds) (2008) Springer handbook of robotics. Springer, Berlin
6. Merriam-Webster (2011) Merriam-Webster thesaurus. http://www.merriam-webster.com/thesaurus/
7. Macmillian (2011) Macmillan dictionary. http://www.macmillandictionary.com/
8. Sioutis C, Tweedale J (2006) Agent cooperation and collaboration. In: Gabrys B, Howlett R, Jain L (eds) Knowledge-based intelligent information and engineering systems. Lecture notes in computer science, vol 4252. Springer, Berlin, pp 464–471. doi:10.1007/11893004_60
9. Horning JJ, Randell B (1973) Process structuring. ACM Comput Surv 5:5–30. doi:10.1145/356612.356614
10. Kohn A (1992) No contest: the case against competition. Houghton Mifflin, New York
11. Roschelle J, Teasley SD (1995) The construction of shared knowledge in collaborative problem solving. In: O'Malley C (ed) Computer-supported collaborative learning. Springer, Berlin, pp 69–97
12. Ditkoff M, Moore T, Allen C, Pollard D (2005) The ideal collaborative team. http://www.ideachampions.com/downloads/collaborationresults.pdf