Article

Digital Twin Generation: Re-Conceptualizing Agent Systems for Behavior-Centered CPS Development

Christian Stary[1,*]

[1] Johannes Kepler University Linz, Business Informatics-Communications Engineering; Christian.Stary@jku.at

* Correspondence: Christian.Stary@jku.at; Tel.: +43 732 2468 4320

Abstract: Cyber-Physical Systems form the new backbone of digital ecosystems. Their design can be coupled with engineering activities to facilitate dynamic adaptation and (re-)configuration. Behavior-oriented technologies enable highly distributed and while coupled operation of systems. Utilizing them for digital twins as self-contained design entities with federation capabilities makes them promising candidates to develop and run highly functional CPS. In this paper we discuss mapping CPS components to behavior-based digital twin constituents mirroring integration and implementation through subject-oriented models. These models, inspired by agent-oriented system thinking can be executed and increase transparency at design and runtime. Patterns recognizing environmental factors and operation details facilitate configuration of CPS. Subject-oriented runtime support enable dynamic adaptation and federated use.

Keywords: Digital Twin; Behavior Modeling; Cyber-Physical Systems; Design-Integrated Engineering; Subject Orientation

1. Introduction

Cyber-Physical System (CPS) design involves a variety of disciplines, leading to contributions of various teams of engineers with diverse backgrounds (Derler et al., 2013). Many system properties influence the design in more than one discipline, while the lack of clearly defined interfaces between disciplines is likely to hinder collaboration of the involved developers (ibid.) According to Hehenberger et al.'s survey study (Hehenberger et al., 2016), CPS-design requires high-level integrated design methods. System modelling forms the ground for the structure and operation of finally developed CPSs. The CPS design process requires to focus on (i) physical components, (ii) digital ones, and (iii) on their integration, preferably in digital representations, termed digital twins. Physical components are those CPS parts that are present in the physical world. According to their capabilities they recognize events and can set actions physically. Digital components represent all computation-relevant CPS parts. Design is finally about integrating these components in a way that the required CPS functionality can be accomplished through their interplay in a socio-technical system.

Hehenberger et al. (2016) concluded that CPS designers directly start working on functional and architecture issues, rather than developing a conceptual model, i.e. digital twin representations. They also conclude from existing work that modeling languages should allow for dynamically modeling architectures with specific reference to CPSs. CPS design should lead to a CPS representation 'as an assembly of components and the associated interfaces between them', that allow 'to model to determine the important functional and system parameters and hence realise the potential optimization.' Khaitan et al., (2014) refer to a CPSs as a "system of systems" due to their heterogeneity and complexity. Such an understanding should help throughout design to model domain specific, continuous interaction and adaptation according to changing configurations. Their list of modeling techniques and programming tools for designing CPSs include formal semantics approaches and architecture styles, such as (multi-)agent and event-based semantic models. For capturing heterogeneity a mixtures of models of computation are advised, whereas the concurrent and dynamic nature seems to favor actor- or agent-based design. Since simulation is a major concern,

the automated execution of models is of benefit. Merlo et al. (2019) have addressed design engineering methods that integrate the user / consumer into design processes and affect modeling. The proposed integrated approach contains the ordered phases 'exploration' as divergent top-down design, 'synthesis' as selection and unit tests, and 'development' as convergent bottom-up design. This system's engineering approach recognizes the role 'customer' managing the CPS, and 'user'. Since there are distinct types of users the user-centered approach ensures their specific requirements for design.

Agent-based systems have become wide-spread in complex environments (Weiss, 2013). While the Oxford American Dictionary provides a fundamental meaning for 'agent' as "a person or thing that takes an active role or produces a specified effect" (http://oaadonline.oxfordlearnersdictionaries.com/dictionary/agent) agents in socio-technical environments, such as business organizations, are active entities causing an environment to change (effects). They might act "on behalf of another, for example, managing business, financial, or contractual matters, or provide a service" (ibid.), such as intelligent recommending. In line with this understanding, Sterling et al. (2009) define an agent as "an entity that performs a specific activity in an environment of which it is aware and that can respond to changes." (p. 7) Wooldridge (2013) seconds "an agent will have a repertoire of actions available to it. This set of possible actions represents the agent's effectoric capability: its ability to modify its environment" (p. 6). Its actions have preconditions associated with them. They capture "the possible situations in which they can be applied" (ibid.) – all attributes qualifying them for CPS design.

Agent behavior has been termed 'intelligent' when an action could be programmed deciding which of an agent's actions is performed in order to best meet the demands of a situation or best satisfy delegated objectives to the agent. As such, agents are decision-making systems that are embedded in an environment. Like CPS they are computer systems that are capable of autonomous action in some environment in order to meet objectives assigned to them by system designers, also termed goal-oriented behavior. A repertoire of actions can be executed "to modify the environment, which may appear to respond non-deterministically to the execution of these actions" (ibid.), either proactively or reactively, either stand-alone or in cooperation with other agents. Hence, software agents (or CPS components) can act autonomously without any direct intervention of humans or other software agents and have full control over their internal state. Each component decides for itself whether or not to perform an action upon request from another agent or human user.

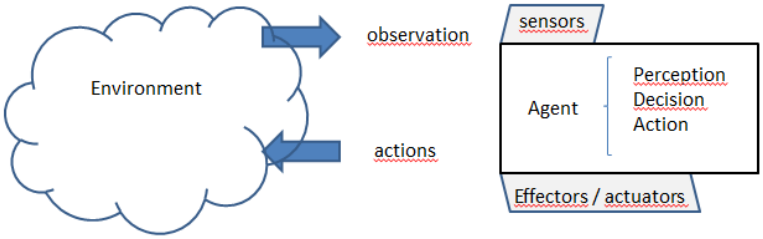

Fig. 1 Conceptual view of CPS component as agent

Fig 1 reveals that CPS conceptualized as agent-based systems can be coupled by sensors to receive inputs from the environment, and have actuators allowing them to create effects in their environment.

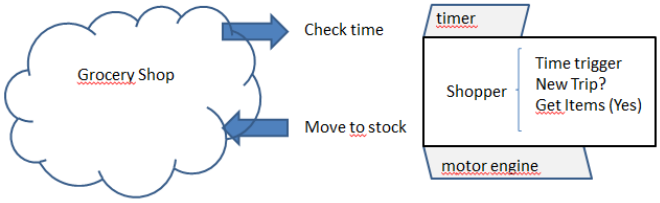

Fig. 2 Grocery shopper example

Fig. 2 and 3 show a typical application of an agent model, namely a shopper agent in a grocery store. It is a behavior abstraction serving the need of an active system component, on one hand reacting on input signals, on the other hand actively influencing other system components. The grocery shopper is triggered by time signals and develops its behavior in the shopping environment according to the internal decisions taken.

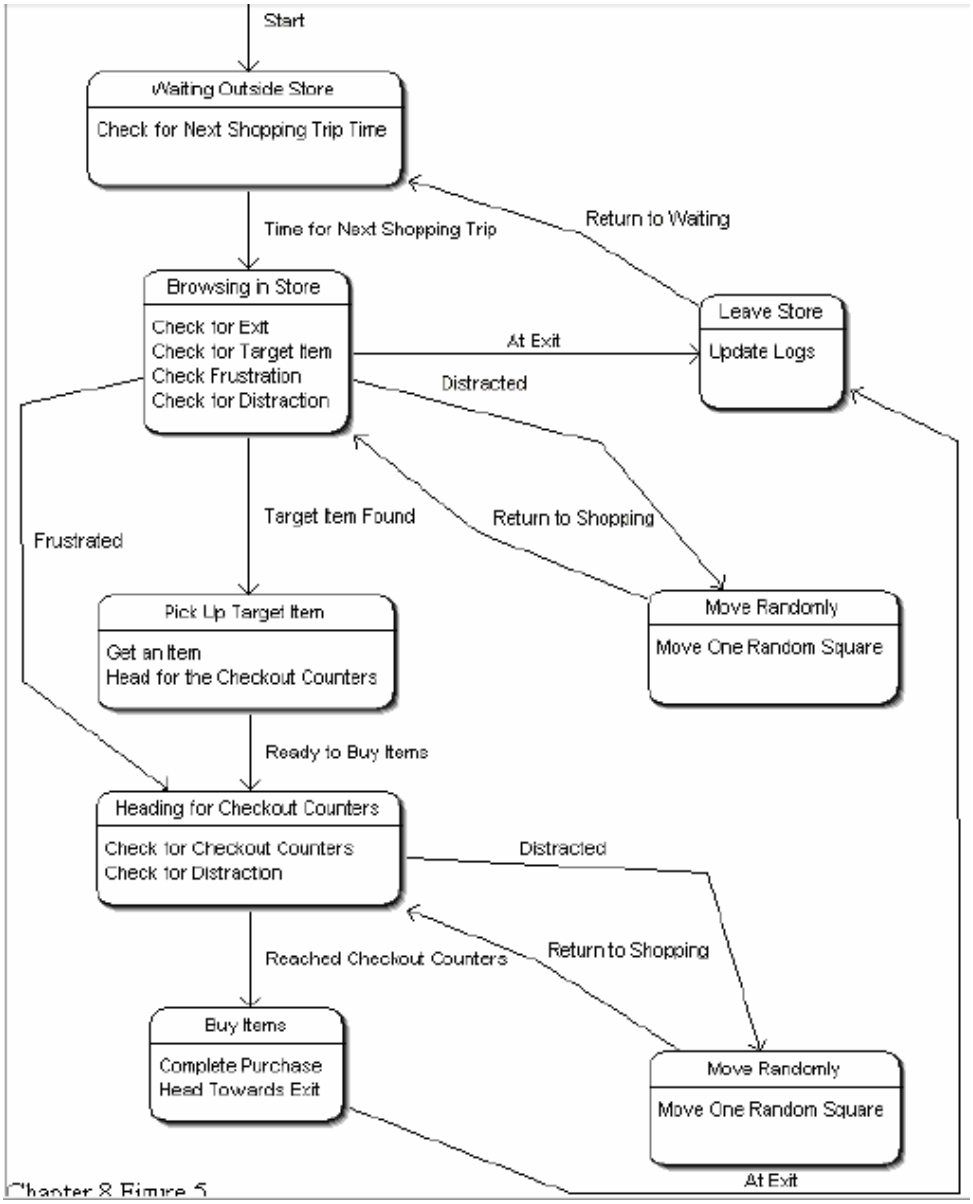

Fig. 3 Shopper behavior encapsulation in grocery store – behavior representation by rules in flowcharts, taken from North et al. (2007)

Such behavior abstractions can be used for representing CPS through digital twins, i.e. allowing for behavior encapsulations that (i) can be executed automatically for probing and simulation, and (ii) abstract from physical and digital runtime evidence. They suggest approach has been established as communication-oriented paradigm in Business Process Management and termed Subject-oriented Business Process Management (S-BPM) (Fleischmann et al., 2012). It provides an approach for both behavior abstraction and dynamic adaptation by actors and CPS components. In this way, and based on automated execution of validated processes, CPS development becomes more effective than using traditional life cycles (as given, e.g., in Weske, 2012). Digital Twins benefit from behavior descriptions

of each component's intelligence (subject behavior diagrams) while preserving overall system performance (subject interaction diagram).

CPS as dynamically adaptable systems can be represented by intertwining agent and subject-oriented approaches while implementing mutual benefits, such as stakeholder-specific abstraction of agent behavior and CPS component abstraction. We offer conclusions for intertwined subject- and agent-oriented CPS engineering. The conceptualization for CPS development aims at:

1. increasing acceptability of behavior-centered approaches in the CPS development driven by Digital Twins as complete digital representation
2. standardization of designs for the sake of pattern-based CPS development, in particular with respect to a semantic infrastructure.

This paper is structured as follows: Section 2 provides some background on current digital twin specification, agent technologies, and subject orientation. Section 3 explores the benefits of subject-oriented conceptualization for CPS as environments based on agent-like behavior encapsulations. Section 4 describes how adaptive behavior of CPS components modeled in an agent-based way can be handled using S-BPM. Section 5 concludes the paper.

2. Digital Twins as Design Representations and Behavior Encapsulation Approaches

We follow the recently consolidated conceptualization of Digital Twins proposed by Wagner et al. (2019). It reflects the nature of CPS, as a digital twin consists of respective parts physical and digital product, and 'connected data that tie and indissolubly connect the physical and virtual product'. It is considered an integrated simulation for optimizing a CPS while mirroring the lifecycle of its actual system pendant. Mirroring means to fully capture the potential or actual physical manufactured product from its micro or atomic level to the macro level through models. Beyond referring to the product a digital twin can contain a production system itself.

However, Digital Twins are mainly used in the product design stages to gain quantitative data for efficient and optimal product generation – see also value chain for product life cycles in the figure below. Digital Twins increasingly contain digital production information, so that production systems can be simulated. Besides manufacturing steps (machine) tools, and process parameters are handled in the course of optimization. Linking product and production twins into digital production twins affects product engineering process, even addressing human control and interaction issues. Wagner et al. (2019) envision enriched use of Digital Twins along product engineering to ensure coherent and efficient flow of information. They also recognize the lack of a consistent framework for holistic application scenarios, particularly addressing the integration of product designers' work and production planner concepts.

The latter has already been addressed recently by Papacharalampopoulos et al., 2020. They propose to study what-if scenarios through experimenting with varying process parameters. They are handled by real time Digital Twins, thus requiring executable models representing relevant design parameters. 'The implementation of a digital twin of a manufacturing process in a real production line should take into consideration, besides the actual development of the consisting model, its validation in terms of very specific concepts and functionalities' (ibid, p. 111). Since real-time estimation distinguish a physics model of a product from a Digital Twin, a Digital Twin facility should have additional capabilities for real-time decision making. In the course of optimization, human interventions when experimenting with different designs become transparent and traceable.

Consequently, a digital twin becomes a virtual instance of a physical system (twin) that is not only continually updated with the latter's operation data on performance and status throughout the entire physical system's life cycle, but could control the system's behavior (cf. Liu et al., 2020). The models representing the digital twin could address CPSs from different perspectives and in different layers, based on networked micro and macro elements.

A CPS is finally characterized by a physical asset and its Digital Twin (cf. Lu et al., 2020) – see also figure. 'In contrast, a Digital Twin is limited to the digital model, not the twinning physical asset, though a Digital Twin cannot live without its twining asset in the physical space. In other words,

Digital Twin represents the prerequisite for the development of a CPS.' (ibid., p. 2). As such, it can serve as an accurate representation of the CPS design process.

Agent technologies provide the fundamental capability of flexible autonomous action in dynamic, unpredictable and open environments, while keeping the benefits of modular, component-based system design. In particular for semantic computing, agents can support the automation and semi-automation of information identification, collection, and processing tasks, in cooperation with other components. Typical applications are (product) recommender systems based on customer preference models (cf. Murray et al., 2009). However, agent technologies in applications could play a manifold role relevant for utilizing them in CPS engineering. Agents can

- represent a metaphor when designing complex, distributed intelligent systems in terms of autonomous units of action
- provide a set of technologies for intelligent systems
- allow for homomorphically modeling real-world systems, in order to meet human-centered needs in artificial systems, such as economics.

What needs to be negotiated for CPS design is the degree of autonomy. On one hand, components (subjects, agents, modules) should be able to respond dynamically to changing circumstances in a self-contained way, on the other hand, a group of components is dedicated to achieve some overarching objectives, including the goals a CPS is designed for. Once the degree of autonomy has been identified, the aggregation of different functionalities that might have previously been distinct (such as planning, doing and coordinating) can be achieved in a conceptually embodied and situated way.

Finally, the provided set of technologies needs to relate directly to these abstractions in the design and conceptualization of systems composed of individual components. These components interact to achieve a common goal. Hence, the consideration of societal or macro-level, e.g., organizations includes issues of software engineering, information architecting, communication infrastructures, and workflow support. Current agent technologies meet these objectives in terms of distributed planning and decision-making, bidding, communication, coordination and negotiation, as well as learning.

Business engineering which builds the context of CPS development, is mainly driven by functional decomposition of value chains when behavior is encapsulated through processes (cf. Davis, 2001). However, in subject-oriented BPM processes are viewed as emerging from both the interaction between these subjects and the local behaviors encapsulated within the individual subjects. Subjects operate in parallel and can exchange messages asynchronously or synchronously. It is a view of processes as autonomous, concurrent behaviors of distributed entities. A subject is a behavioral role assumed by an "actor"; i.e. an entity that is capable of performing actions. The entity can be a human, a piece of software, a machine (e.g., a robot), a device (e.g., a sensor), or a combination of these. Subjects can execute local actions that do not involve interacting with other subjects (e.g., calculating a price and storing a postal address), and communicative actions that are concerned with exchanging messages between subjects, i.e. sending and receiving messages.

Subjects are one of five core symbols used in S-BPM. Based on these symbols, two types of diagrams can be produced to conjointly represent a process: Subject Interaction Diagrams (SIDs) and Subject Behavior Diagrams (SBDs). SIDs provide a global view of the process, including the subjects involved and the messages they exchange. The SID of a simple ordering process is shown in Fig. 1. Subject Behavior Diagrams (SBDs) provide a local view of the process from the perspective of individual subjects. They include sequences of states representing local actions and communicative actions including sending messages and receiving messages. State transitions are represented as arrows, with labels indicating the outcome of the preceding state (see Fig. 5 & 6).

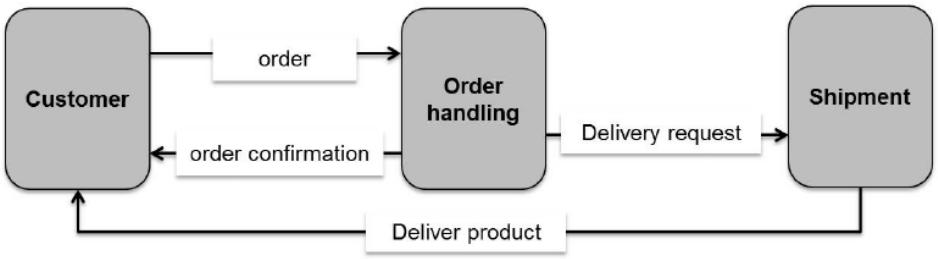

Fig. 4 Order Handling – Subject Interaction Diagram

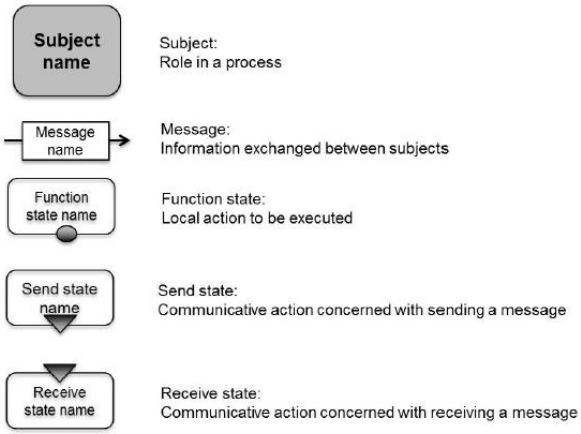

Fig. 5 Diagrammatic Elements for agent-based Subject-oriented Behavior Encapsulation

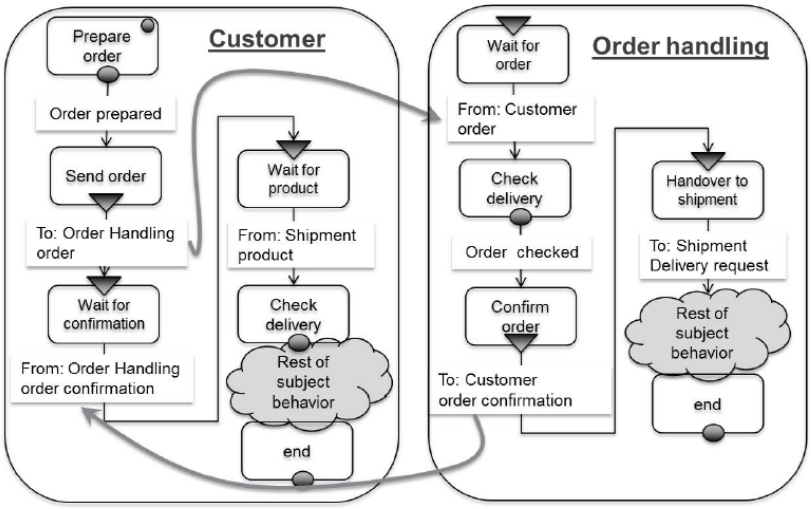

Fig. 6 Order Handling – Subject Behavior Diagrams ‘Customer’ and ‘Order Handling’

Given this potential of current status of development, agent-based and subject-oriented bahvior encapsulation can be intertwined either on the basis of

- predefined communications protocols (possible when using SIDs) and standardized behavior specifications (enabled by SBDs), however limiting scalability of systems, or

- semi-structured communication languages, such as FIPA ACL, and design methodologies such as GAIA (Wooldridge et al., 2000).

The latter allow meeting ad-hoc, non-deterministic, and domain-specific requirements, as Nealen et al., 2003 have shown for healthcare. The ultimate stage of scalability could be reached through dynamic and intelligent formation of components and system architectures beyond domains, referring to adaptivity (cf. Sterling et al., 2009).

By means of agent-oriented concepts intelligence is distributed, as Nealen et al. (2003) explains that the basic properties of intelligent agents are 'autonomy, proactivity, social ability, while the features of multi-agent systems are 'management of distributed information, communication and coordination between separate autonomous entities' (p. 9). For business applications transparent specification and processing efficiency have been brought into play (cf. Sterling et al., 2009, p. 6): "There is a need and expectation for instantaneous responses, which will be achieved only by efficient implementations in light of the complexity. Efficiency may well determine the possibility of solutions—for example, file sharing or downloading large video files. A more abstract fourth attribute is purposefulness. In light of the complexity and changing nature of the environment, it will be difficult—if not impossible—for all requirements to be stated. It is better to work at a higher level and to explain purposes in terms of goals, and, in certain circumstances, to have the system determine the appropriate path of action. This approach can aid system design and clarity, which leads to the next attribute.

The final attribute is a little bit different and perhaps less obvious; namely, the software should be understandable, at least in its design and overall purpose. We need ways of thinking about and describing software that simplify complexity, at least with regard to describing behavior. The desire for understandability is influenced by the software engineering perspective undertaken within this book. There are many potential advantages of understandable software, including better instructions for how to use it." Hence, complex systems, once their components and relationships are understood, can be more easily designed, maintained, and developed further.

In summary, more explicit representations of intelligence facilitates dynamic arrangements of components and environments, like required for CPS development from a stakeholder perspective. .

3. CPS Behavior Encapsulations

We start out with the concept of 'agent' and proceed with multi-agent environments, in order to represent dynamic business process settings in a subject-oriented way.

3.1 CPS Components as Self-contained Objects = Subjects

Behavior encapsulations stemming from agent systems are considered (re-)active entities, processing inputs from an environment and producing output according to their processing scheme, which cause an environment to change. As particularity they might interact before they react on environmental inputs (depending on the precondition associated with their actions) involving other agents. Trying to identify the best action in the current situation, agents decide which of their actions is performed according to the inputs provided. Finally, agents can act proactively, requesting data from the environment to check whether further interventions are required.

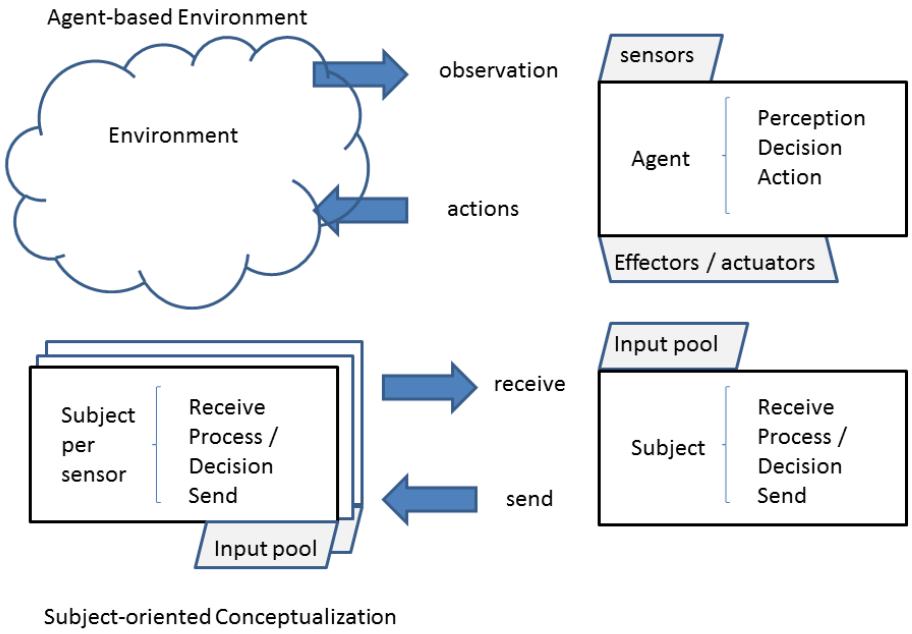

Fig. 7 Agent-based Environments (upper part) conceptualized through Subject Representations (lower part) as used for Digital Twin representation – each sensor and effector/actuator becomes a subject

The upper part of Fig. 7 conceptualizes agents as interfacing their environment (according to Wooldridge, 2013 as given in Fig. 1). The lower part of Fig. 7 shows subjects establishing a subject-based environment comprising the conceptual agent model. Each subject in that environment is able to receive inputs (stored in an input pool) that are processed. Its internal behavior allows decision making before either proceeding with actions and/or sending messages to other subjects.

The fundamental difference between agent-oriented environment representations (as shown in Fig. 1 according to Wooldridge, 2013) and subject orientation ones is that the latter allow representing explicitly all effects an agent's reaction or action has. It models effectors/actuators of the agent environment regardless of whether an agent or subject acts in a reactive, pro-active, or social way. When behaving reactively, agents receive inputs from their environment that trigger appropriate actions in response to external events and changes that occur in the environment. When behaving pro-actively agents take initiative and perform self-contained, goal-directed actions. When acting in a social way, agents interact via cooperation, coordination, and negotiation with other agents (and possibly with humans) to perform specific tasks according to a common goal.

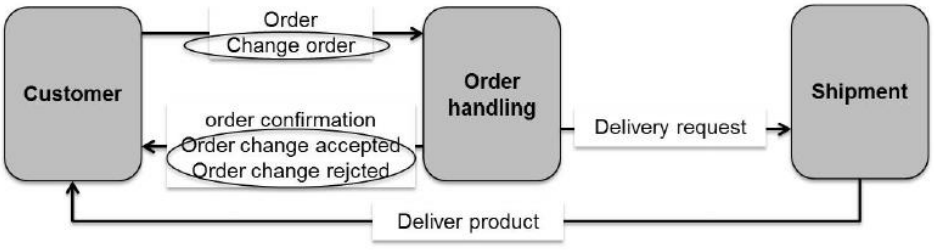

Fig. 8 Subject-Interaction Diagram including change of order

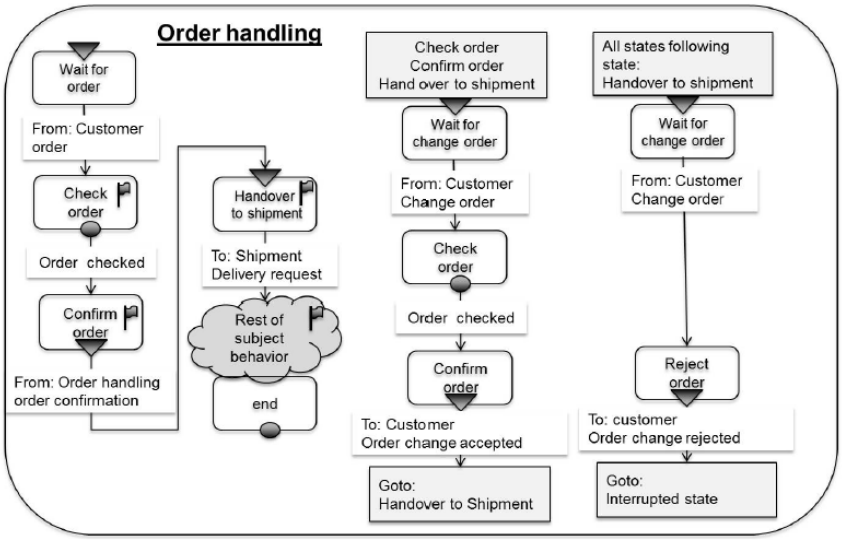

Fig. 9 Order Handling SBD including change of order

Fig. 8 and 9 show subject-oriented representations of a reactive agent. Both, on the level of Subject-Interaction Diagrams (SIDs as the one in Fig. 8) and on the level of Subject-Behavior Diagrams (SBDs, e.g., in Fig. 9) changes of customer orders can be specified. According to S-BPM the subject Order handling 'waits' for the respective input (message) to start decision making. In this way, the pro-activeness of the resulting business organization becomes visible and behavior patterns can be adapted accordingly.

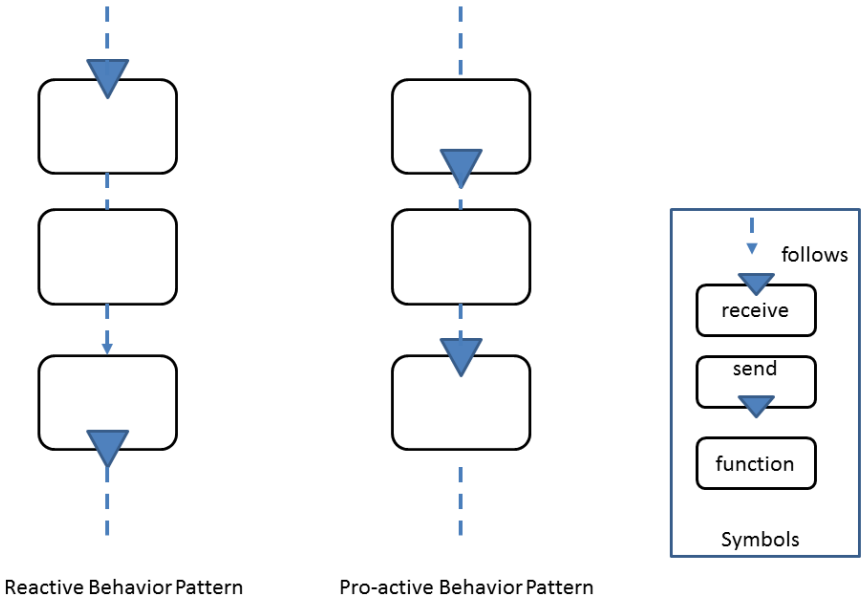

Fig. 10 Subject-oriented representation of pro-active and reactive behavior patterns

Fig. 10 shows patterns for pro-active and reactive behaviors in subject-oriented notation. Reactive behavior is based on temporal sequences of loosely coupled receive – do – send states (see above for order changes). Pro-active behavior requires asking for inputs, namely sending requests, while performing regular tasks, and awaiting response (see Fig 11 for handling orders).

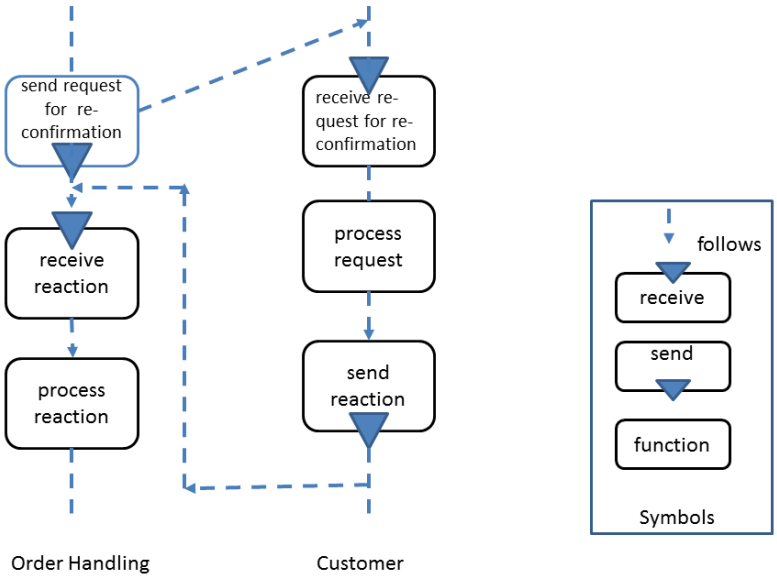

Fig. 11 Pro-active behavior pattern of the subject Order Handling

Such a pattern-based approach can be a first step towards representing adaptive behavior. Stratulat et al. (2009, p. 813) recognizes the challenge of capturing interaction between agents, respecting autonomy and self-organization of system components: "Agents are considered as active autonomous components with respect to the control of their behavior and to the relationship they have with other components of the system, i.e. the other agents and the environment in which they act. The various studies in the MAS [Multi-Agent System] domain showed that the interaction in an agent-based system has multiple facets that are difficult to grasp. The consequence of this is that the basic concepts and principles of MAS have been identified and studied much of the time in isolation from the others and the various perspectives that are adopted are often orthogonal and mutually exclusive (e.g. internal vs. public meaning of the communication, rational vs. reactive internal architectures, agent-based vs. environment-mediated interaction, agent-oriented vs. organization-centered MAS, etc.). However, it seems that if we want to have a deeper understanding and a more complete perspective of the interaction process some efforts should be made to rethink all the basic elements and to integrate as many as possible of them into a unifying framework." Therefore, in their paper, they describe a general framework and an abstract model "of what constitutes a first step towards an integral view of agent-based interaction" (*ibid*.).

Sterling et al. (2009, 9f) highlighted several qualities of agents: (i) Purposefulness, pursuing a goal and determining paths of action accordingly, (ii) controlled autonomy, as it is able to pursue its own goals seemingly independently, (iii) situatedness of agents, i.e. being aware of its surrounding environment. It must be capable of perceiving changes and responding appropriately.

The introduced subject-oriented elements meet all qualities while explicitly modeling the goal-relevant part of the environment as subjects:

- A subject is a behavior abstraction for a specific purpose. It might stem from a functional role, type of application, or intention to capture behavior on an abstract level.
- A subject has controlled autonomy, as it encapsulates behavior to pursue a specific goal independently.
- A subject is situated in an environment of subjects, and, most importantly, it is aware of this environment of the other subjects it is exchanging messages with in this environment. This mechanism allows not only perceiving changes and responding appropriately, but acquiring information about behavior of other subjects of the environment.

In particular, the latter property helps to pro-actively collect information of relevance to change behavior – see section 4.

3.2 CPS = Multi-Agent Encapsulsations plus Relevant Environment Subjects

In a multi-agent system a collection of software agents collaborate in an external environment via their sensors and effectors (cf. Fig. 4). Hereby, Nealon et al. (2003), p. 2, refer to the underlying system characteristics, namely communication: Agents "can communicate with users or other agents. Thus, they can exchange information, engage in complex negotiations, and coordinate their activities to cooperate in the joint resolution of a problem. Agents usually have reasoning, planning and learning capabilities that allow them to display an intelligent behaviour."

However, abstraction of multi-agent systems has so far received little attention (Cohen et al. 2009). In particular agent-based modeling is still under development (Heath et al., 2009). Most approaches serve implementation and architecture purposes (cf. Cheng et al., 2009, Seghrouchni et al., 2009), and only few for strategic business development via simulation (e.g. North et al. (2007)). Hahn et al. (2009) recognize the demand for further research: "Various agent-oriented methodologies and metamodels exist to design and develop multiagent systems (MAS) in an abstract manner. Frequently, these frameworks specialise on particular parts of the MAS and only few works have been invested to derive a common standardisation. This limits the impact of agent-related systems in commercial applications." In their work they argue for developing "a metamodel for agent systems that abstracts from existing agent-oriented methodologies, programming languages, and platforms and could thus be considered as platform-independent". Their metamodel allows defining an abstract syntax of domain-specific modelling languages for MAS to provide for code generation out of generated designs. They apply the principles of model-driven development (MDD) and utilize two model transformations to bring the generated models into textual code that can be executed.

Wooldridge (2013) considers multi-agent systems not only as systems composed of multiple interacting intelligent agents that interact to solve problems that are beyond the individual capabilities or knowledge of each individual, but also as decentralized systems without global control, and asynchronous computation. However, the dispute on the level of granularity seems not to be resolved yet. For instance, Cohen et al. (2009) proposes reducing the state space of MAS to a tractable size. They have presented an abstraction technique for MAS preserving temporal-epistemic properties. When the state space of an MAS is too large for a model checker to compute, it is sometimes possible to reduce it by simplifying the local states and actions of agents before feeding the system to a model checker. If the model checker reports that a design requirement (expressed in the temporal epistemic logic) holds, they conclude that the requirement holds also for the original system. Thus, they abstract a multi-agent system by simplifying the local states, the local protocol and the local evolution function of each agent. They model multi-agent systems in the interpreted systems framework (Fagin et al., 1994), where each agent is described by

- a set of possible local states it can be in,
- a set of actions it can perform,
- a local protocol selecting actions depending on the current local state, and
- a local evolution function specifying how an agent evolves from one local state to another depending on its own action and the actions of other agents.

This approach is in line with subject orientation. Each Subject Behavior Diagram (SBD) captures all possible local states a subject can be in, namely in terms of send, receive, and act. They represent the actions it can perform as well as the interfaces to other subjects. The local protocol is given by triggering actions internally (do) or externally (send). In this way the protocol for subject interaction is defined: receiving a message triggers an internal action of the addressed subject. The inputs to subjects trigger the evolution procedure in terms of proceeding from one local state to another depending on its own activities and the actions of other subjects (i.e. incoming messages from other subjects).

In the framework developments initially the environment has been left out (cf. Raimondi et al., 2007) which is not the case for subject-oriented representations. The explicit embodiment of subjects

(agents) in their environment is rather seen as a prerequisite for further developing intelligent behavior, and thus, intelligence of systems, such as organizations.

Proponents of MAS claim that a model of an agent should be "constructed to aid in building the system that we have in mind. To paraphrase Parnas's well-known characterization of specifications, a model should be as complex as it needs to be to reflect the issues the system is being built to address, but no more complex" (Sterling et al., 2009, 14). Given the example provided by the authors (UML), models should not only be 'intuitively understandable', but also support the construction of software systems at a level that allows model checking and reviewing in terms of system correctness before code is generated and the system is implemented. Consequently, interfaces between components, such as classes, subjects, or agents need to be defined. Additionally, the passing mechanisms of information need to be specified while "the models must have sufficient detail to be useful, but not so much detail as to overwhelm." (*ibid*., 15)

North et al. (2007) go one step further, considering modeling as an essential prerequisite for simulation. An MAS model therefore has to contain:

- A set of agents (part of the user-defined model)
- A set of agent relationships (part of the user-defined model)
- A framework for simulating agent behaviors and interactions (provided by a toolkit)

For simulation, unlike other modeling approaches, modeling is driven by the perspective of autonomous decision-making units. Therefore an agent representation, such as subject-oriented models, needs to capture rules of behavior operating affecting attributes. The decision rules vary for every intelligent component (subject) in terms of sophistication and representations of the external world. Simulation is based on "local" interaction among components, since no central authority or controller exists for

- how the system operates
- how the system is modeled
- how the system/model moves from one state to another.

These objectives can be met by MAS systems represented by subjects:

- Each subject has its individual intelligent behavior, represented by state transitions triggered by inputs and dedicated decision rules.
- The interaction among subjects is not disturbed by any supervising authority. It is driven by individual subjects sending and receiving messages. Hence, the overall business system corresponds to choreographed self-contained units (subjects).

Consequently, any optimization of a system composed by subjects is based on the intertwining of individual behaviors (subjects) through communication. North et al. (2007) propose rules based on theories of individual rational behavior. Hereby, individuals collaborate with others when it is in their best economic interest to do so. They collaborate to survive. Furthermore, decision making should be based on simple rules, in order to let rule structures evolve. In this way, bounded rationality can be taken into account. Reasoning regarding goals is progressively refined by means of procedures accounting for the limited knowledge and abilities of individual decision makers. In line with that we propose to use modular decision-making patterns as shown in Fig. 12. X,Y, and Z represent situative parameters, such as accepting, rejecting or ask for refining an order, which trigger different behaviors, expressed in the subsequent branches (Send according to X, e.g., accepting the order, Act according to Y, e.g., preparing the request for refinement, and Send according to Z, e.g., rejecting an order).

Such structures of MAS can be used to map the shopper agent's behavior effectively to execution patterns. Since subject models can be executed after validation and in a concurrent way (cf. Fleischmann et al., 2012) they allow for simulation of handling complex situations.

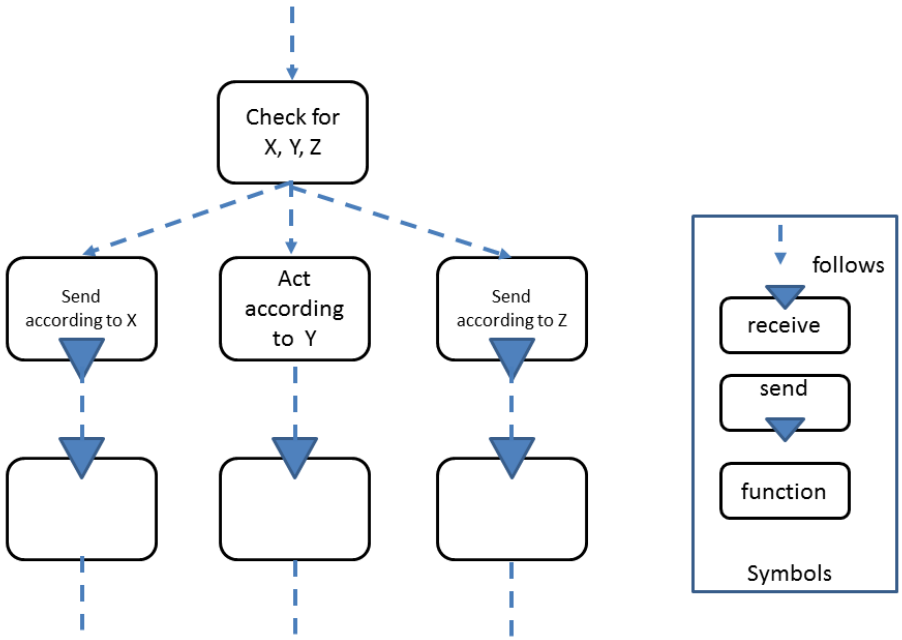

Fig. 12 Decision making patterns

For convenience, exception handling is supported by execution support facilities that can be used at design time. In this way, non-standard behavior can be distinguished from routine ones. In Fig 13 the message guard pattern that has been applied when addressing the change of orders in Fig. 3 requires hissing a 'red' flag that triggers substitutive procedures (returning to regular SBD sequence) or complementary behavior (leaving the originally executed SBD).

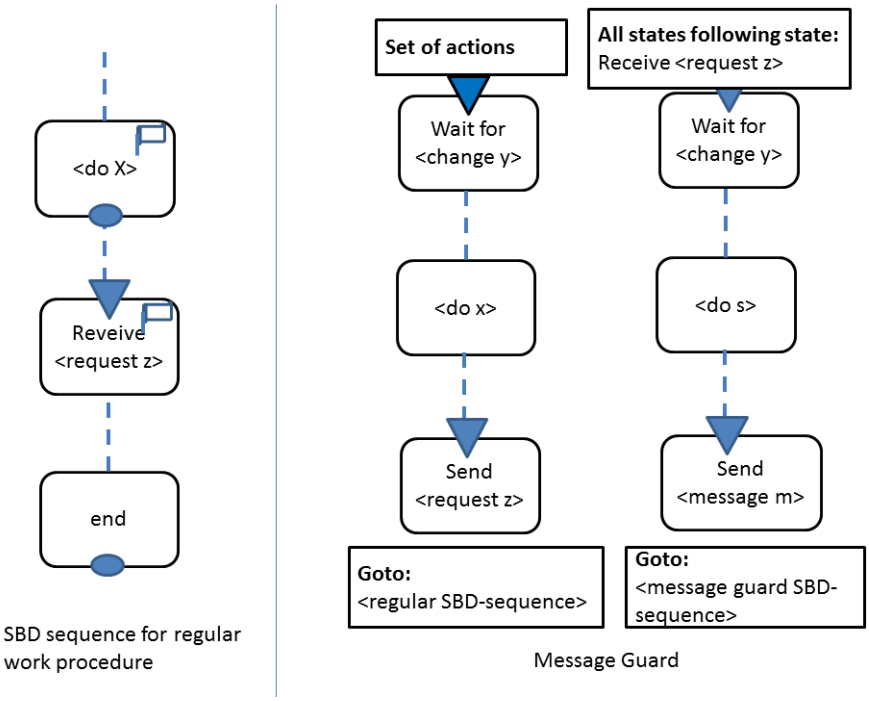

Fig. 13 Intelligent behavior specifications - Handling changes dynamically in subject-oriented systems

4. Communication-based Engineering

In this section we focus on the subject-oriented conceptualization with respect to adaptive behavior based on sense data. For adaptation subjects need to (i) become informed, and (ii) selective with respect to their behavior. The approach is inspired by Gero et al.'s (2000, 2002) proposed architecture for situated agents, implementing their architecture by dedicated communication patterns. Hereby, the subject's 'sensors are monitor subjects that (actively) search for relevant changes in the environment and produce direct input for an acting subject. In addition, a decision support subject (termed hypothesizor by Gero et al.) can be invoked by an acting subject. It is provided with monitored information and situation-sensitive data to identify mismatches between the current and desired situation. It is supposed to support the acting subject 'deciding on actions that when executed are likely to reduce or eliminate that mismatch' (Gero et al., 2003, 102). Based on the results of the decision support process, the acting subject can decide which sequence of operations to execute.
Fig 14 provides the corresponding interaction schema for communication-based engineering of agent-based environments. It shows that an <acting subject> can ask for both, monitoring the situation, and decision support based on the monitored information. An <acting subject> is any subject in an environment that needs to take an action to accomplish a work task. The Monitor subject embodied in the environment either accepts requests to collect data on the current situation or automatically receives data and pushes it to the <acting subject>. The Decision Support subject is available for consultancy with respect to selecting the next action. It requires all available information on a certain situation.

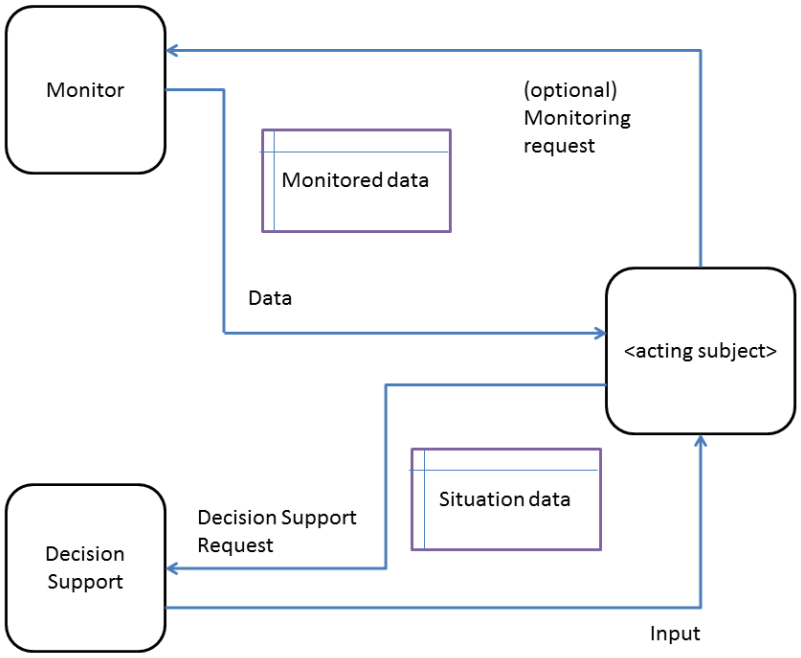

Fig. 14 An <acting subject> can ask for monitoring a situation and decision support based on the monitored information

A Monitoring subject can either receive and operate on environment data automatically or monitor the behavior of other subjects. Fig 15 shows both types of monitors. The environment monitor is a signal receiver which processes them to deliver data, e.g., a temperature sensor. The social monitor actively requests data from other subjects, e.g., whether certain data values have changed. Both could iterate for refining the results.

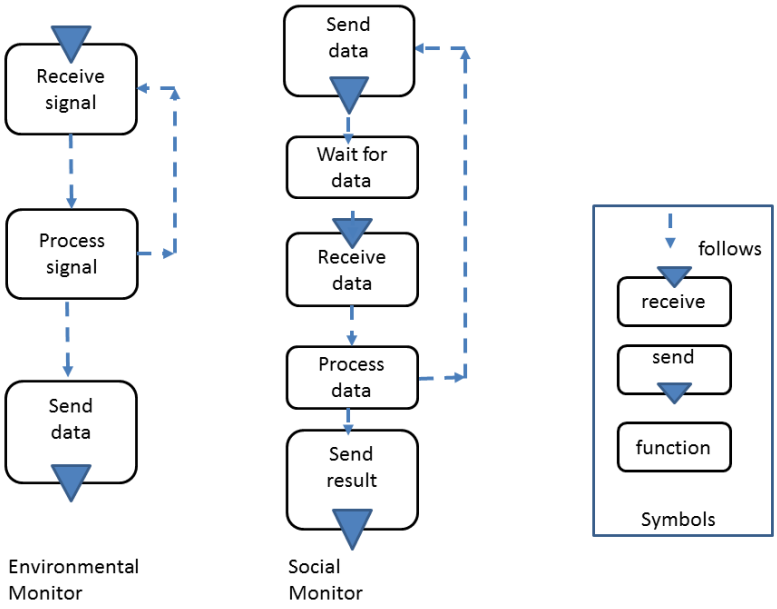

Fig. 15 Push- and pull monitoring a situation

The communication of the <acting subject> can be conceptualized accordingly. Fig. 16 details an <acting subject> sending a request and waiting for the result (case of social monitoring).

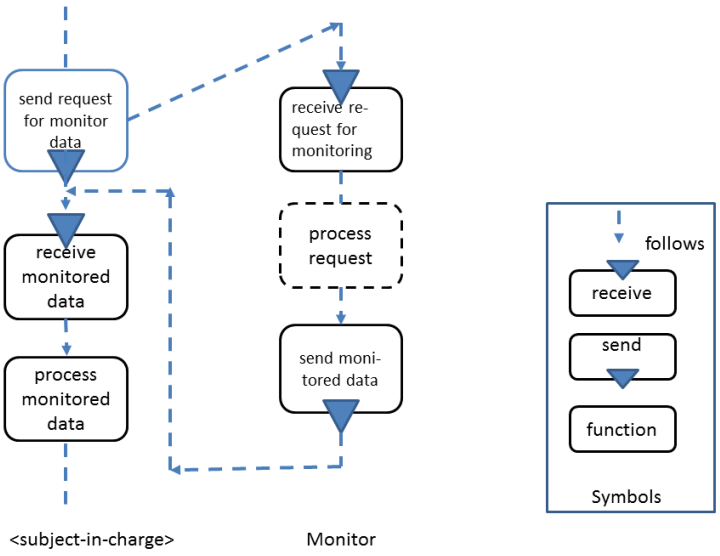

Fig. 16 A <subject-in-charge> interacts with Monitor (encapsulating how the request is processed)

The request for monitoring can be modeled before any function state, thus providing for each critical function a preprocessing sequence. It considers environmental data beyond subject interaction (sensors, effectors/actuators) and can be captured also in behavior descriptions.

Once an <acting subject> has received monitor data it can act in response to environment data. The input data from Monitor are then processed along the subject's behavior. In case a subject's action requires decision support, it needs to activate the Decision Support subject. The initial step hereby is requesting inference on situation-specific data, not only using the monitored data, but also data created or processed by the <acting subject> itself. The input data are sent to Decision Support as they describe the current situation of the <acting subject>. Processing the request for decision support

requires a business rule engine that holds for the business at hand. The received results are processed according to available sets of rules and the situation data provided by the <acting subject>. The results are finally delivered to the <acting subject> - see Figure 17.

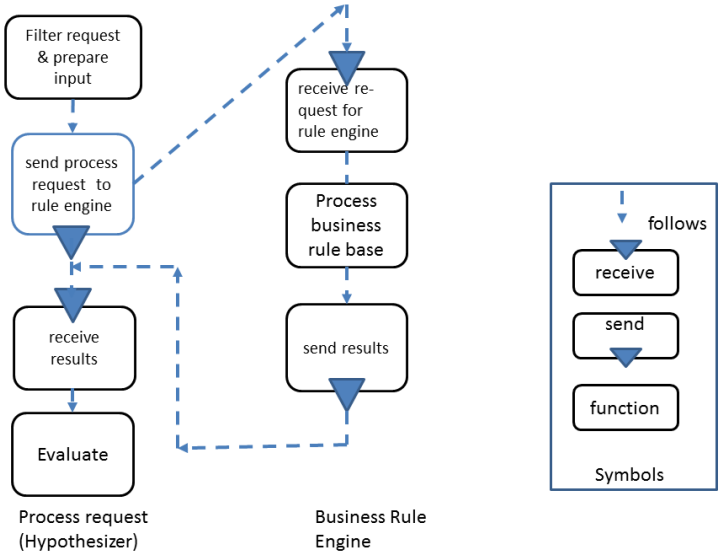

Fig. 17 Rule processing

The evaluated results are checked and a decision pattern as shown in Fig. 12 could be triggered to complete a work task. In Fig 18 we assume a shopper subject looking at its shopping list and deciding to buy another product while in a store (cf. Fig. 3). The Monitor subject is activated to find out whether the other product is on stock. In order to avoid frustration in case it is not on stock, an alternative product is looked up – a Decision Support subject with respect to consumer behavior is activated and checked whether it is available, until the shopper can be satisfied or informed about a lack of further alternatives.

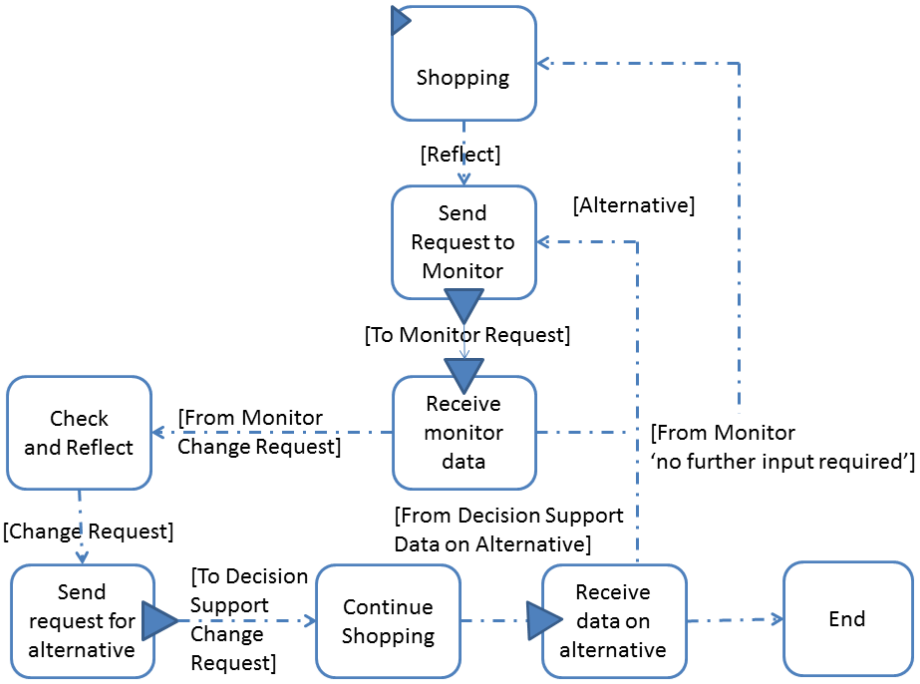

Fig. 18 Grocery case involving stock monitoring and decision support

Given monitoring and decision support, an acting subject can perform in a reflexive way, using input data from the environment, either preprocessed from interaction behavior from other subjects or from observer components, such as sensors. Moreover, an acting subject can also activate a decision support subject. In this case, monitored data from the environment can be enriched with situation-sensitive data by the decision support subject for further processing, e.g., according to general business rules. The results lead to informed decision taking of the acting subject. Hence, a CPS based on such intelligent components, can be provided with emergent behavior sequences.

5. Discussion and Conclusion

CPS can be designed in a behavior-centered way to support accomplishing tasks in dynamic and complex systems. In this paper we have revisited their structure and behavior from an agent-based perspective. We suggest using subject-oriented representations of Digital Twins for CPS system development for the sake of comprehensive, effective and efficient system modeling and implementation. We have identified major model patterns for behavior-centered Digital Twin development. They meet fundamental qualities of modular intelligent systems, namely behavior abstraction for a specific purpose, and controlled autonomy. The subject-oriented modeling approach delivers executable twins, hence even allowing dynamic allocation of physical and digital runtime elements to a CPS.

Author Contributions:

Funding: This research received no external funding.

Acknowledgments: The discussions with several i2pm.net members and proponents of S-BPM, in particular Albert Fleischmann and Werner Schmidt, are acknowledged.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Alter, S. (2008) Service system fundamentals: Work system, value chain, and life cycle, in: IBM Systems Journal, 47(1), 71-85
2. Bradshaw, J.M., Carvalho, L.M., Bunch, L., Eskridge1, T., Feltovich, P.J., Forsythe, C., Hoffman, R.R., Johnson, M., Kidwell, D., Woods, D.D. (2012) Coactive Emergence as a Sensemaking Strategy for Cyber Operations, IHMC Technical Report, Pensacola, FL
3. Brambilla, M., Fraternali, P., Vaca, C. (2012) BPMN and design patterns for engineering social BPM solutions, in: Business Process Management Workshops, eds: Daniel, F., Barkaoui, K., Dustdar, S., LNBIP 99 Springer, Berlin, 219-230
4. Buhler, P., Vidal, J.M. (2004) Enacting BPEL4WS Specified Workflows with Multiagent Systems, in: Proceedings Workshop on Web Services and Agent-Based Engineering, New York City
5. Buhler, P., Greenwood, D. , Reitbauer, A. (2005) A Multiagent Web Service Composition Engine, in: Proceedings First International Workshop on Engineering Service Compositions, Amsterdam, The Netherlands, IBM Research Report RC23821, IBM Research Division
6. Castro, J., Kolp, M., Mylopoulos, J. (2001) A requirements-driven development methodology, in: Proceedings CAiSE 2001, eds: Dittrich, K.R., Geppert, A., Norrie, M.C. LNCS 2068, Springer, Heidelberg, 108-123
7. Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (Eds.) (2009) Software Engineering for self-adaptive systems, LNCS 5525, Springer, Berlin
8. Cohen, M., Dam, M., Lomuscio, A., Russo, F. (2009) Abstraction in model checking multi-agent systems, in: Proceedings 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), eds.: Decker, Sichman, Sierra and Castelfranchi, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org)
9. Davis, R. (2001) Business Process Modelling with ARIS: A Practical Guide, Springer, Berlin
10. Derler, P., Lee, E. A., Tripakis, S., & Törngren, M. (2013). Cyber-physical system design contracts. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems* (pp. 109-118).

11. Dixon, J., Jones, T. (2011) Hype Cycle for Business Process Management, 2011, Gartner Research, Stamford, CT
12. Elammari, M.; Issa, Z. (2013) Using Model Driven Architecture to Develop Multi-Agent Systems, in press
13. Elammari M., Lalonde W. (1999) An Agent-Oriented Methodology: High-Level and Intermediate Models, in: Proceedings of AIOS 99, Heidelberg, Germany, 1999.
14. Fagin, R., Halpern, J.Y., Vardi, M.Y, Moses, Y. (1995) Reasoning about knowledge, MIT Press, Cambridge, MA
15. FIPA – Foundation for Intelligent Physical Agents. Standards available at http://www.fipa.org
16. Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E. (2012) Subject-Oriented Business Process Management, Springer, Berlin
17. Fleischmann, A., Schmidt, W., Stary, C., Kannengießer, U. (2013) Subject-Oriented Modeling and Execution of Multi-Agent Business Processes, in: Proc. WI-IAT 2013, Int. Conf. on Intelligent Agent Technology, IEEE/WIC/ACM.
18. Gero, J.S., Fujii, H. (2000) A computational framework for concept formation for a situated design agent, in: Knowledge-Based Systems, 13(6), pp. 361-368.
19. Gero, J.S:, Kannengiesser, U. (2002) The Situated-Function-Behaviour-Structure framework, in: J. S. Gero (ed.), Artificial Intelligence in Design'02. Kluwer, Dordrecht, pp. 89-104.
20. Gero, J. S., Kannengiesser, U. (2003) Function-Behaviour-Structure: A Model for social situated agents, in: Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions, International Joint Conference on Artificial Intelligence, IJCAI, 101-107.
21. Grefen, P., Mehandjiev, N., Kouvas, G., Weichhart, G., & Eshuis, R. (2009) Dynamic business network process management in instant virtual enterprises, in: Computers in Industry, 60(2), 86-103
22. Hahn, C., Madrigal-Mora, C., Fischer, K. (2009) A platform-independent metamodel for multiagent systems, in: Autonomous Agents and Multi-Agent Systems, 18(2), 239-266
23. Heath, B., Hill R., Ciarallo, F. (2009) A survey of agent-based modeling practices (January 1998 - July 2008), Journal of Artificial Societies and Social Simulation, 12(4), 9-43
24. Hehenberger, P., Vogel-Heuser, B., Bradley, D., Eynard, B., Tomiyama, T., & Achiche, S. (2016). Design, modelling, simulation and integration of cyber physical systems: Methods and applications. *Computers in Industry*, *82*, 273-289.
25. Herrmann, Th. (2012) Kreatives Prozessdesign. Konzepte und Methoden zur Integration von Prozessorganisation, Technik und Arbeitsgestaltung, Springer, Berlin
26. Huhns, M., Stephens, L. (2001) Automating Supply Chains, in: IEEE Internet Computing, 5(4), 90-93
27. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Odgers, B., Alty, J.L. (2000) Implementing a Business Process Management System using ADEPT: A real-world case study, in: International Journal of Applied Artificial Intelligence, 14(5), 421–463
28. Jennings, N.R., Wooldridge, M. (2001) Agent-Oriented Software Engineering, in: Handbook of Agent Technology, ed.: J. Bradshaw, AAAI/MIT Press
29. Khaitan, S. K., & McCalley, J. D. (2014). Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems Journal*, *9*(2), 350-365.
30. Laclavık, Z., Balogh, M., Babık, L., Hluchy (2006) AGENTOWL: Semantic knowledge model and agent architecture, in: Computing and Informatics, Vol. 25, 421–439
31. Liu, C., Jiang, P., & Jiang, W. (2020). Web-based digital twin modeling and remote control of cyber-physical production systems. *Robotics and Computer-Integrated Manufacturing*, *64*, 101956.
32. Lu, Y., Liu, C., Kevin, I., Wang, K., Huang, H., & Xu, X. (2020). Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, *61*, 101837.
33. McBurney, P., Parsons, M. (2009) Dialogue games for agent argumentation, in: I. Rahwan, G. R. Simari (eds.), Argumentation in Artificial Intelligence, DOI 10.1007/978-0-387-98197-0 13, _c, Springer Science+Business Media, LLC, 261-280
34. Merlo, C., Abi Akle, A., Llaria, A., Terrasson, G., Villeneuve, E., & Pilniere, V. (2019). Proposal of a user-centred approach for CPS design: pillbox case study. *IFAC-PapersOnLine*, *51*(34), 196-201.
35. Miller J., Mukerji J. (2003) MDA Guide, Version 1.0.1.OMG, 2003, from http://www.omg.org/docs/omg/03-06-01.pdf
36. Milner, R. (2006) Ubiquitous computing: Shall we understand it? The Computer Journal 49, 383-389
37. Milner, R. (2009) The space and motion of communicating agents, Cambridge University Press, Cambridge

38. Murray, K. B., Häubl, G. (2009). Personalization without interrogation: Towards more effective interactions between consumers and feature-based recommendation agents. Journal of Interactive Marketing, 23(2), 138-146.
39. Nealon, J. L., Moreno, A. (2003) Agent-based applications in health care, in: Applications of Software Agent Technology in the Health Care Domain, eds: Nealon J, Moreno A., Birkhäuser, Basel, 3-18.
40. Nikraz M., Caire G., Bahri P. (2006) A Methodology for the Analysis and Design of
41. Multi-Agent Systems using JADE, in: International Journal of Computer Systems Science and Engineering, 21(2)
42. North, M.J., Macal, C.M. (2007) Managing business complexity: Discovering strategic solutions with agent-based modeling and simulation, Oxford University Press, Oxford
43. Papacharalampopoulos, A., Stavropoulos, P., & Petrides, D. (2020). Towards a digital twin for manufacturing processes: Applicability on laser welding. *Procedia CIRP*, *88*, 110-115.
44. Pedell, S., Miller, T., Sterling, L., Vetere, F., Howard, S. (2011) Substantiating agent-based quality goals for understanding socio-technical systems, Proceedings of the First International Workshop on Agent-Based Modeling for Policy Engineering (AMPLE 2011), pp. 97-112
45. Pohl, K. (2010) Requirements Engineering: Fundamentals, principles, and techniques. Springer, Heidelberg
46. Raimondi, F., Lomuscio., A. (2007) Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. Journal of Applied Logic, 5(2):235–251
47. Seghrouchni, A.E.F., Dix, J., Dastani, M., Bordini, R.H. (eds.) (2009) Multi-agent programming. languages, tools and applications, Springer, Heidelberg
48. Smarsly, K., Law, K.H. (2011) Advanced Structural Health Monitoring based on Multi-Agent Technology, Stanford University, Research Report, Stanford
49. Sterling, L., Taveter, K. (2009) The art of agent-oriented modeling, MIT Press, MA
50. Stratulat, T., Ferber, J.,. Tranier, J. (2009) MASQ: Towards an integral approach to interaction Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, 813-820
51. Tuerel, M.A., Navarro, E., López-Jaquero, V., Montero, F. (2011) CSRML: A goal-oriented approach to model requirements for collaborative systems, in: Proceedings ER 2011, LNCS 6998, Springer, Berlin, 33-46
52. Wagner, R., Schleich, B., Haefner, B., Kuhnle, A., Wartzack, S., & Lanza, G. (2019). Challenges and potentials of digital twins and Industry 4.0 in product design and production for high performance products. *Procedia CIRP*, *84*, 88-93.
53. Weiss, G. (ed.) (2013) Multi-agent Systems, MIT Press, Cambridge, MA.
54. Weske, M. (2012). Business process management: concepts, languages, architectures. Springer, Berlin. Wooldridge, M., Jennings, N. R., Kinny, D. (2000) The Gaia methodology for agent-oriented analysis and design, in: Autonomous Agents and Multi-Agent Systems, 3(3), 285-312
55. Wooldridge, M. (2002) An introduction to Multiagent systems. John Wiley, Chichester
56. Wooldridge, M. (2013) Intelligent Agents, in: Multi-agent Systems, 2nd edition, ed.: Weiss, G., MIT Press, Cambridge, MA, p. 1-50