Towards Data- and Norm-aware Multiagent Systems

Matteo Baldoni(✉)[1], Cristina Baroglio[1], Diego Calvanese[2], Roberto Micalizio[1], and Marco Montali[2]

[1] Università degli Studi di Torino — Dipartimento di Informatica
c.so Svizzera 185, I-10149 Torino (Italy)
firstname.lastname@unito.it

[2] Free University of Bozen-Bolzano — KRDB Research Centre
Piazza Domenicani 3, I-39100 Bolzano, Italy
lastname@inf.unibz.it

Abstract. We recall the key abstractions and models on which the major approaches to software specification rely, using Meyer's forces of computation as dimensions of comparison. Based on the identified strengths and lacks, we introduce data-awareness and of norm-awareness as recommended properties, explaining the advantages they bring about. We show that multiagent systems are a good candidate for the development of a data- and norm-aware programming, tracing directions for the realization of multiagent systems that are data and norm-aware. Finally, we report and comment some proposals from the multiagent systems literature that, though developed independently and not inserted in an organic framework, already face specific aspects that are relevant to bring about norm and data-awareness.

1 Introduction

One of the key characteristics of agents is their situatedness [41, 50, 51], i.e. the fact that an agent is immersed in an environment, be it social or physical, that it perceives, senses, and acts upon. Despite the centrality of situatedness, most studies in the research area on multiagent systems are focussed only on features of agents, while those that put forward the need of representing the environment either (1) disregard the plurality of data, thus typically relying on a propositional representation, as explained in [36], or (2) do not provide a representation of the process by which data evolve in a form that can be reasoned about, as we underline in this work.

We advocate that, in order for agents to be capable of dealing with richer data representations that go beyond the propositional case, it is necessary to rely on an information system through which data can, for instance, be aggregated or information can be extracted (*data awareness*). The environment, for what concerns its being used by the agents, should be specified on top of building blocks that amount to semantically meaningful chunks of data, which evolve as a consequence of the agents' actions. The description of how data evolve should

be provided by the environment to its agents as a body of norms. This would allow agents to deliberate how to act and which goals to pursue also in terms of expectations about the evolution of the environment (*norm awareness*). Gathering from proposals like [6, 18], we propose to describe the environment in terms of data information models and data lifecycles, that are to be made available to the agents in their deliberation process. A data information model specifies the structure of the information, a data lifecycle, instead, specifies data state transitions. Finally, it is capitol that data-awareness and norm-awareness are realized in a way that does not compromise the agents' deliberative capabilities. Problems may, in fact, arise when no bound is placed on the number of tuples that can be added to database relations as the computation goes on [5, 36].

Let us make a couple of examples. In a propositional setting, it is common to consider an order as pertaining to an interaction session. Combining different orders of a same client into a single shipping procedure would be positive in various respects (to reduce pollution, to save money, to make the client happy by receiving everything in one box), but the exhibition of such a behavior requires to distill information from the data specifying the different orders, to associate the orders to the single client, and to know that all orders follow a same evolution, whose description should be available to the agents in a form that can be reasoned about. Only such kind of awareness would provide the agents the means to adapt their behavior to the cases which are captured by the actual data. Similarly, in a warehouse that received various orders concerning items of a same kind, and that will undergo some packaging process, it would be more efficient to first pick all the items up (probably they will be on the same shelf) and only after start to pack them up. Instead, in a propositional setting the pick-and-pack can only occur one item at a time, introducing a considerable waste of time. Of course, it is always possible to hard-code some optimization procedure in the agents' behaviors but the interesting thing would be that the agents adapted autonomously, after reasoning on data, without any hard-coding.

To explain our point, we start the paper by recalling the key abstractions and models on which the major approaches to software specification rely, including both the ones developed by the research area on multiagent systems and those proposed by other research communities. We provide an organic view by relying on Meyer's three forces of computation [33] as reference dimensions, along which all the considered proposals are positioned. To this aim, Section 2 introduces Meyer's forces of computation, while Section 3 overviews approaches to software specification, ranging from functional decomposition to multiagent systems.

Based on the strengths and lacks emerged in this part of the paper, Section 4 introduces data-awareness and norm-awareness as recommended properties, showing that multiagent systems are a good candidate for the development of a data- and norm-aware programming. It explains the advantages brought about by this vision, tracing directions to the realization of multiagent systems that are data and norm-aware. Section 5, then, reports and comments some proposals in the multiagent systems literature that, though developed indepen-

dently and not inserted in an organic framework, face specific aspects that are relevant to bring about norm and data-awareness. Conclusions end the paper.

2 Meyer's forces: Processor, Action and Object

We decided to use *Meyer's forces of computation* as a common ground for comparing the different proposals because they provide a neutral touchstone, unrelated to any specific programming approach or modularization mechanism. According to Meyer, three forces are at play when we use software to perform some computations (see Figure 1): *processors*, *actions*, and *objects*. A processor can be a *process* or a *thread* (in the paper we use both the terms processor and process to refer to this force); actions are the *operations* that make the computation; objects are the *data* to which actions are applied.

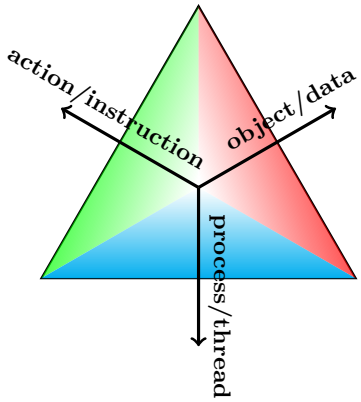

Fig. 1. Meyer's three forces of computation [33, Chapter 5, page 101].

A software system, in order to execute, uses processes to apply certain actions to certain objects. The form of the actions depends on the considered level of granularity: they can be instructions of the programming language as well as they can be major steps of a complex algorithm. Moreover, the form of actions conditions the way in which processes operate on objects. Some objects are built by a computation for its own needs and exist only while the computation proceeds; others (e.g., files or databases) are external and may outlive individual computations. In the following we analyse the most important proposals concerning software modularization, showing how they (sometimes implicitly) give more or less strength to Meyer's forces, and the drawbacks that follow.

3 From Functional Decomposition to MAS

It becomes apparent that processor and object are the two principal forces along which most approaches to software modularization have been developed so far, while the action force remained subsidiary to one or another.

Functional Decomposition. The top-down *functional decomposition* is probably the earliest approach to building modularized software; it relies on a model that puts at the center the notion of process; namely, the implementation of a given function is based only on a set of actions made of instructions, provided by the programming language at hand, possibly in combination with previously defined functions [33]. Top-down functional decomposition builds a system by stepwise refinement, starting with the definition of its abstract function. Each refinement step decreases the abstraction of the specification. With reference to Figure 1, the approach disregards objects/data, just considered as data structures that are instrumental to the function specification and internal to processes. Actions are defined only in terms of the instructions provided by the programming language and of other functions built on top of them (subroutines), into which a process is structured. All in all, this approach is intuitive and suitable to the development of individual *algorithms*, in turn aimed at solving some specific *task*, but does not scale up equally well when *data are shared among concurrent processes* because it lacks abstractions to explicitly account for such data and their corresponding management mechanisms.

Object-Orientation. The *Object-Oriented* approach to modularization results from an effort aimed at showing the limits of the functional approach [33]. Objects (data) often have a life on their own, independent from the processes that use them. Objects become, then, the fundamental notion of the model. They provide the actions by which (and only by which) it is possible to operate on them (*data operations*). This approach, however, disregards processes and their modularization both internally and externally to objects. Internally, because objects provide actions but have a *static nature*, and are inherently passive: actions are invoked on objects, but the decision of which operations to invoke so as to evolve such objects is taken by external processes. This also implies that there is no decoupling between the *use of an object* and the *management of that object*. Externally, because the model does not supply conceptual notions for composing the actions provided by objects into processes, and there is no conceptual support to the specification of tasks, in particular when concurrency is involved.

Actor Model, Active Objects. The key concept in the *actor model* [30] (to which *active objects* are largely inspired) is that *everything is an actor*. Interaction between actors occurs only through *direct asynchronous message passing*, with no restriction on the order in which messages are received. An actor is a computational entity that, in response to an incoming message, can: (1) send a finite number of messages to other actors; (2) create a finite number of new actors; (3) designate the behavior to be used in response to the next incoming message.

These three steps can be executed in any order, possibly in parallel. Recipients of messages are identified by opaque addresses. Interestingly, in [30] Hewitt et al. state that "We use the ACTOR metaphor to emphasize the inseparability of control and data flow in our model. Data structures, functions, semaphores, monitors, [...] and data bases can all be shown to be special cases of actors. All of the above are objects with certain useful modes of behavior." The actor model *decouples* the sender of a message from the communications sent, and this makes it possible to tackle asynchronous communication and to define control structures as patterns of passing messages.

Many authors, such as [34, 46, 37], noted that the actor model does not address the issue of *coordination*. Coordination requires the possibility for an actor to have expectations on another actor's behavior, but the mere asynchronous message passing gives no means to foresee how a message receiver will behave. For example, in the object-paradigm methods return the computed results to their callers. In the actor model this is not granted because this simple pattern requires the exchange of two messages; however, no way for specifying patterns of message exchanges between actors is provided. The lack of such mechanisms hinders the verification of properties of a system of interacting actors. Similar problems are well-known also in the area that studies enterprise application integration [1] and service-oriented computing [45], that can be considered as heirs of the actor model and where once again interaction relies on asynchronous message passing. There are in the literature proposals to overcome these limits. For instance for what concerns the actor model. [37] proposes to use Scribble protocols and their relation to finite state machines for specification and runtime verification of actor interactions. Instead, in the case of service-oriented approaches, there are proposals of languages that allow capturing complex business processes as service compositions, either in the form of orchestrations (e.g. BPEL) or of choreographies (e.g. WS-CDL).

The above problem can better be understood by referring to Meyer's forces. The actor model supports the realization of object/data management processes (these are the internal behaviors of the actors, that rule how the actor evolves), but it does not support the design and the modularization of processes that perform the object use, which would be *external* to the actors. As a consequence, generalizing what [14] states about service-oriented approaches, the modularization supplied by the actor model, while favoring component reuse, does not address the need of connecting the data to the organizational processes: data remains hidden inside systems.

Business Processes. Business processes have been increasingly adopted by enterprises and organizations to conceptually describe their dynamics, and those of the socio-technical systems they live in. Modern enterprises [13] are complex, distributed, and aleatory systems: complex and distributed because they involve offices, activities, actors, resources, often heterogeneous and geographically distributed; aleatory because they are affected by unpredictable events like new laws, market trends, but also resignations, incidents, and so on. In this light, *business processes* help to create an explicit representation of how an enterprise

works towards the accomplishments of its tasks and goals. More specifically, a business process describes how a set of interrelated activities can lead to a precise and measurable result (a product or a service) in response to an *external event* (e.g., a new order) [49]. Business processes developed for understanding how an enterprise work can then be refined and used as the basis for developing software systems that the enterprise will adopt to concretely support the execution of its procedures [13, 24]. In this light, business processes become *workflows* that connect and coordinate different people, offices, organizations, and software in a compound flow of execution [1]. Among the main advantages of this process-centric view, the fact that it enables analysis of an enterprise functioning, it enables comparison of business processes, it enables the study of compliance to norms (e.g. [27]), and also to identify critical points like bottlenecks by way of simulations (e.g., see iGrafx Process[3] for Six Sigma). The adoption of a service-oriented approach and of web services helps implementing workflows that span across multiple organizations, whose infrastructures may well be heterogeneous and little integrated [1, 45].

On the negative side, business processes, by being an expression of the process force, show the same limits of the functional decomposition approach. Specifically, they are typically represented in an activity-centric way, i.e., by emphasizing which flows of activities are acceptable, without providing adequate abstractions to capture the data that are manipulated along such flows. Data are subsidiary to processes.

Artifact-centric Process Management. The *artifact-centric approach* [6, 18, 14] counterposes a data-centric vision to the activity-centric vision described above. *Artifacts* are concrete, identifiable, self-describing chunks of information, the basic building blocks by which business models and operations are described. They are business-relevant objects that are created and evolve as they pass through business operations. They include an *information model* of the data, and a *lifecycle model*, that contains the key states through which the data evolve, together with their transitions (triggered by the execution of corresponding tasks). A change to an artifact can trigger changes to other artifacts, possibly of a different type. The lifecycle model is not only used at runtime to track the evolution of artifacts, but also at design time to understand who is responsible of which transitions.

On the negative side, like in the case of the actor model, business artifacts disregard the design and the modularization of those processes that operate on them. Moreover, verification problems are much harder to tackle than in the case where only the control-flow perspective is considered. In fact, the explicit presence of data, together with the possibility of incorporating new data from the external environment, makes these systems infinite-state in general [14].

Agents and Multiagent Systems. In [41, 51], *agents* are defined as entities that observe their environment and act upon it so as to achieve their own goals. Two

[3] `http://www.igrafx.com/`.

fundamental characteristics of agents are *autonomy* and *situatedness*. Agents are autonomous in the sense that they have a sense-plan-act deliberative cycle, which gives them control of their internal state and behavior; autonomy, in turn, implies proactivity, i.e., the ability of an agent to take action towards the achievement of its (delegated) objectives, without being solicited to do so. Agents are situated because they can sense, perceive, and manipulate the environment in which operate. The environment could be physical or virtual, and is understood by agents in terms of (relevant) data. From a programming perspective, it is natural to compare agents to objects. Agent-oriented programming was introduced by Shoham as "a specialization of *object-oriented programming*" [42]. The difference between agents and static objects is clear. Citing Wooldridge [51, Section 2.2]: (1) objects do not have control over their own behavior[4], (2) objects do not exhibit flexibility in their behavior, and (3) in standard object models there is a single thread of control, while agents are inherently multi-threaded. Similar comments are reported also by other authors, like Jennings [31]. However, when comparing agents to actors, the behavioral dimension is not sufficient: [51, page 30] reduces the difference between agents and active objects, which encompass an own thread of control, to the fact that "active objects are essentially agents that do not necessarily have the ability to exhibit *flexible* autonomous behavior". In order to understand the difference between the agent paradigm and objects it is necessary to rely on both the abstractions introduced by the agent paradigm, that are that of agent and that of environment [50]. Such a dichotomy does not find correspondence in the other models and gives a first-class role to both Meyer's process and object force (see Figure 2). Processes realize algorithms aimed at achieving objectives, and this is exactly the gist of the agent abstraction and the rationale behind its proactivity: agents exploit their deliberative cycle (as control flow), possibly together with the key abstractions of belief, desire, and intention (as logic), so as to realize algorithms, i.e., processes, for acting in their environment to pursue their goals[5]. Contrariwise, active objects and actors do not have goals nor purposes, even though their specification includes a process. As we said, they are a manifestation of the object force. In the agent paradigm the manifestation of the object force is the environment abstraction. The environment does not exhibit the kind of autonomy explained for agents even when its definition includes a process. Its being reactive rather than active makes the environment more similar to an actor whose behavior is triggered by the messages it receives, that are all served indistinctly.

The A&A meta-model. Despite the centrality of situatedness in the definition of agents, most of the research in multiagent systems typically focuses on the abstraction of agent only, completely abstracting away from the notion of environment. Proposals like [22, 50] overcome this limit by introducing first-class

[4] This is summarized by the well-known motto "Objects do it for free; agents do it because they want it".

[5] Summarizing, objects "do it" for free because they are data, agents are processes and "do it" because it is functional to their objectives.

abstractions for the environment, to be captured alongside agents themselves. In particular, [50] states that "the environment is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources." This proposal brought to important evolutions like the A&A meta-model [38] and its implementation CArtAgO [39].

Normative Multiagent Systems. A fundamental step towards raising the value of the action force is brought by *normative multiagent systems* [32, 8], which take inspiration from mechanisms that are typical of human communities, and have been widely studied in the research area on multiagent systems. According to [8] a normative multiagent system is: "a multiagent system together with normative systems in which agents on the one hand can decide whether to follow the explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms". Initially the focus was posed mainly on *regulative norms* that, through obligations, permissions, and prohibitions, specify the patterns of actions and interactions agents should adhere to, even though deviations can still occur and have to be properly considered [32]. More recently, regulative norms have been combined with *constitutive norms* [7, 15, 19], which support the creation of institutional realities by defining institutional actions that make sense only within the institutions they belong to. A typical example is that of "raising a hand", which counts as "make a bid" in the context of an auction. Institutional actions allow agents to operate within an institution. Citing [19], the impact on the agent's deliberative cycle is that agents can "reason about the social consequences of their actions". In this light, going back to Meyer's forces, if agents are abstractions for processes and environments for objects, then *norms* are abstractions of the *action force* (see Figure 2) because norms model actions and, thus, condition the way in which processes operate on objects. In fact, norms specify either institutional actions, or the conditions for the use of such actions, consequently regulating the acceptable behavior of the agents in a system. This view is also supported by the fact that norms concern "doing the right thing" rather than "doing what leads to a goal" [48].

4 Need of Data and Norm Awareness

Reality is complex even in simple settings because it involves data, and data are related and compose semantically meaningful chunks of information. The realization of systems where a set of autonomous and heterogeneous parties can interact effectively, leveraging the richness of the data they create and manipulate through their actions, requires, on the one hand, data-awareness and, on the other hand, a specification of the rules by which data evolve, that agents should take into account to decide if and how to act (norm-awareness). These two kinds of awareness should be seamlessly integrated in the system through appropriate abstractions.

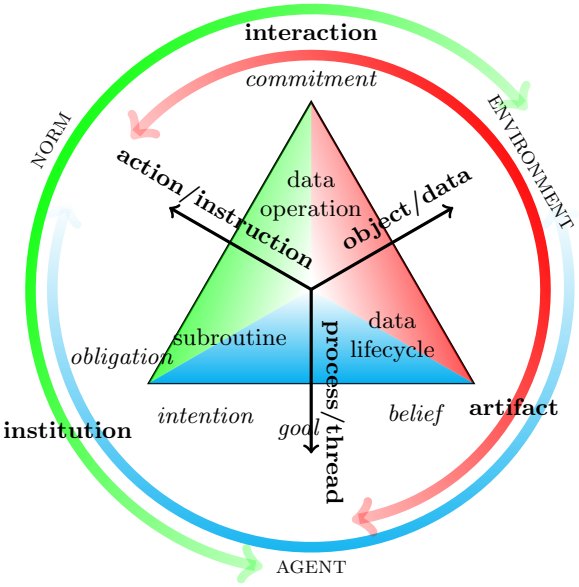

Fig. 2. Rereading Meyer's forces.

Of the many approaches to the specification and modularization of software that we have discussed, multiagent systems are particularly promising. One key aspect in this respect is the fact that, differently than in the other approaches, the action force is not ancillary to the process force nor to the object force. Actions are the capabilities agents have to modify their environment. The process force is mapped onto a cycle in which the agent observes the world (updating its beliefs), deliberates which intentions to achieve, plans how to achieve them, and finally executes the plan [11]. Beliefs and intentions are those components of the process abstraction that, with reference to Figure 2, create a bridge respectively towards the object/data force (i.e., the environment) and the action force. Beliefs concern the environment. Intentions lead to action [51], meaning that if an agent has an intention, then the expectation is that it will make a reasonable attempt to achieve it. In this sense, intentions play a central role in the selection and the execution of action. This independence of the action force from the other two is what enables the use of norms as an abstraction of the action force and, so, to model the specification of data lifecycles by way of norms. Note that, even though in general data-awareness and norm-awareness are orthogonal to BDI notions, it is natural to think of agents as BDI agents for a seamless integration of all the aspects of deliberation, including the awareness of data and of their lifecycles.

While in functional decomposition actions are produced by refining a given goal through a top-down strategy, intentions are a means by which the action force is put in relation to the process force. Thus, while in other approaches actions are hard-coded, so to say, in the process, an agent's deliberative process is independent of the actions it uses and, in particular, it can concern also actions by other agents. So, for instance, consider a setting where the order lifecycle is available to the agents in a way that can be reasoned about. An agent, who is handling part of the lifecycle of an order, may conclude that, since it has to pick up three items in the warehouse, since each such item will have to be packed, since all packagings are performed by a same other agent, and since one of its goals is saving energy, it is preferable to first pick them all up and only then deliver them to the other agent. Data-awareness here is awareness that three items of a same kind are requested. Norm-awareness that items are picked because each of them is part of some order, whose lifecycle says that after being picked they will be packed. Again data-awareness allows our agent to know that the orders are different and that all parcels are to be made by a same other agent.

Notice that approaches that rely on the object force do not provide the abstractions that allow realizing the warehouse example because they do not foresee an abstraction like that of agent (not even of process). Consequently, object-orientation associates operations to data, but the paradigm did not push the study towards a normative representation. Similarly, while business artifacts provide both a rich description of their data and their lifecycle, they do not provide any link to a corresponding normative understanding, thus making impossible for the agents (could any be defined) to leverage this knowledge for reasoning about how to act. On the other hand, artifacts in the A&A model are radically different from the business artifacts because they do not come with an explicit information model for data, and they do not exhibit data lifecycles. Thus, this information cannot be exploited at design time, nor at runtime, to reason about which actions should be taken towards the achievement of the agent goals.

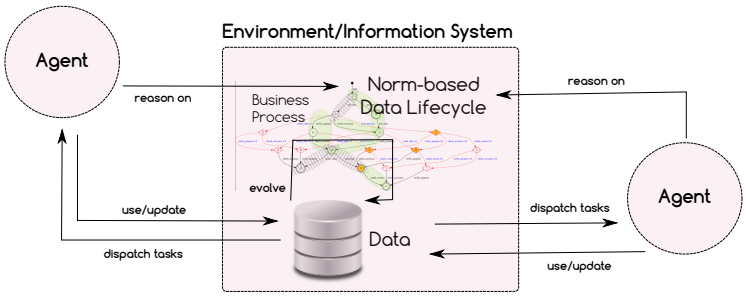

Fig. 3. Data-aware and norm-aware multiagent system.

Another reason that makes agents promising is that agents already show the capability of tackling norms. This is due to the fact that, since in the agent paradigm each agent is an independent locus of control, coordination means are deemed as essential towards regulating the overall behavior of the system. As it is well underlined in [31], the agent-based model allows to naturally tackle the issue of coordination by introducing the concepts of interaction protocol [16], and that of *norm* [26, 48]. These concepts are at the heart of the design of multiagent systems. The deliberative cycle of agents is affected by the norms and by the obligations these norms generate as a consequence of the agents' actions. In principle, each agent is capable to adapt its behavior to (local or coordination) changing conditions, e.g., by re-ranking its goals based on the context or by adopting new goals, and free to do it or not. Institutions and organizations are a way to realize functional decomposition in an agent setting. Intuitively, an institution is an organizational structure for coordinating the activities of multiple interacting agents, that typically embodies some rules (norms) to govern participation and interaction. In general, an organization adds to this societal dimension a set of organizational goals, and powers to create institutional facts or to modify the norms and obligations of the normative system [7]. Agents, playing roles, must accomplish the organizational goals respecting the norms. The limit is that, despite the centrality of norms, a holistic proposal where constitutive norms are used to specify both agent actions and data operations, and where regulative norms are used to create expectations on the overall evolution of the system (agents behavior and environment evolution) is yet to be developed.

Data and Norm-aware Multiagent Systems. A data-aware and norm-aware multiagent system, see Figure 3, should involve a group of agents and of business artifacts with the following characteristics. Agents interact with each other and with the environment by creating and modifying data which belong to an information system and that are reified by business artifacts. The conceptual model of the information system is described in terms of the norms that regulate the evolution of such data. Norms express data lifecycles, i.e. they capture how data pass from one state to another as a consequence of actions that are performed by some agent. The conceptual model is available to the interacting agents in a form that allows agents to reason on it. The agents are aware of the current state (of the lifecycle) of the data, and thus of the tasks expected of them and of their parties. At design time, norms would provide a programming interface between agents and their environment, given in terms of those state changes that are relevant in the environment.

5 Steps towards Data and Norm Awareness

Data- and norm-awareness, in the sense introduced in this paper, are not yet realized in multiagent systems but the literature already contains independent efforts that tackle specific aspects of this direction of research, which, thus, fit in the picture we have drawn. Interestingly, many of such works focus on social

commitments which emerge as currently occupying a central position in the junction between norms and data.

A first example is provided by the JaCaMo+ platform [3], which allows Jason agents [10] to engage commitment-based interactions [43], in turn reified as CArtAgO [38] artifacts (both agents and artifacts are first-class elements in the design of the multiagent system). JaCaMo+ artifacts implement the social state of the interaction, which is the social environment in which agents act, and provide the roles that are then enacted by the agents. The explicit representation of the social state enables the realization of a data-aware approach, where the data are the events occurring in the social state, while social commitments provide the information necessary to agents in their interaction. A social commitment $C(x, y, s, u)$ captures that agent x (debtor) commits to agent y (creditor) to bring about the consequent condition u when the antecedent condition s holds. Antecedent and consequent conditions are conjunctions or disjunctions of events and commitments. The interesting point about commitments is that they have a lifecycle [47]: a commitment is *null* right before being created; *active* when it is created; active has substates *conditional* (as long as the antecedent condition did not occur), and *detached* (when the antecedent condition occurred, the debtor is engaged in the consequent condition of the commitment); an active commitment can become: *pending* if suspended; *satisfied*, if the engagement is accomplished; *expired*, if it will not be necessary to accomplish the consequent condition; *terminated* if the commitment is canceled when conditional or released when active; and finally, *violated* when its antecedent has been satisfied, but its consequent will be forever false, or it is canceled when detached (the debtor will be considered liable for the violation). JaCaMo+ explicitly represents the states of the commitments, allowing the agents to take also this information into account in their reasoning. Commitments in JaCaMo+ belong to the social state and are shared by the interacting agents as resources. So, they are information, that is created and evolves along the interaction with event occurrence, and that contributes to the specification of the environment in which the agents operate. In this light, the social state can be seen as a special kind of business artifact in the sense of [6, 18, 14]. JaCaMo+ allows specifying agent programs as Jason plans, whose triggering events amount to the change of the state of some commitment [2]. Suppose, to make an example, that the commitment goes to the state "detached" and that this event triggers a plan in the agent which is the debtor of that commitment: the connection between the commitment and the associated plan is not only causal (event triggers plan), but rather the plan is explicitly attached to the commitment, in the sense that its aim is to satisfy the consequent condition of the commitment (norm-awareness).

An independent proposal, i.e. [20], then shows how commitments lifecycle can be captured by a set of norms. It explains the advantages of this view which are: (1) enabling agents to take into account the evolution of commitments in their reasoning; (2) allowing the customization of the commitment lifecycle to the needs of particular application contexts. This proposal fits the understanding

of norm-awareness we have explained and it provides evidence of the advantages of a norm-centered description of data evolution.

de Brito *et al.* [21] explain the limits of current approaches to artificial institutions, e.g. [25], basically residing in the fact that proposals always remain at an abstract level that does not account for the tight connection between the institution and its environment. So, for instance, the institution will say that "the winner of an auction is obliged to pay its offer, otherwise it is fined" without specifying aspects such as what an agent should do to become the winner of the auction, how payments are made, or how a fine is applied. The work overcomes the limit of the traditional approaches by allowing a specification of regulations that is based on facts occurring in the environment (an aspect that we interpret as data-awareness). The important consequence is that in this way, the institution does not depend on agents informing about norm violation, goal achievement, role adoption, etc. for the relevant information is obtained from the environment. In [21], situated artificial institutions are specified in terms of norms and constitutive rules. Norms are based on status functions, like winner, payment. Constitutive rules state the conditions for an element of the environment to carry a status function. For example, if the environment has an automatic teller machine implemented by an artifact, an operation in such artifact could count as the payment.

Other works make proposals for going beyond the propositional representation, which characterizes most studies on multiagent systems, underlining the importance of putting information in the centre. In particular, the Cupid language [17] provides a sophisticate and information-centric representation that distinguishes between a schema (what occurs in a specification) and its instances (what transpires and is represented in a database), reserving the term commitment only for schemas. This avoids the inadequacy of first-order in representing commitment instances by relying on relational database queries. The advantages, brought to the analysis of properties of a data-aware approach are proved in DACMAS [36], which incorporates commitment-based MASs but in a data-aware context. In general, in presence of data transition systems become typically infinite-state [14]. On the one hand, this is due to the fact that there is no bound on the number of tuples that can be added to database relations as the computation goes on. On the other hand, even when the number of tuples does not exceed a certain threshold, it is possible to populate them using infinitely many different data objects. Interestingly, when a DACMAS is state-bounded, i.e., the number of data that are simultaneously present at each moment in time is bounded, verification of rich temporal properties becomes decidable. Notably, this shows that, by suitably controlling how data are evolved in the system, it is possible to make agents data-aware without compromising their reasoning capabilities [5, 36]. A language for representing norms that guarantees a priori the decidability of property analysis would be a great advancement being the tool that agents need to reason and decide which action to take, thus leveraging their autonomy. JaCaMo [9], simpAL [40], JaCaMo+ [2] are existing platforms for the development of MAS that have the right potential for developing the view depicted in

Figure 2. The next step would be the introduction of information-centric artifacts, whose lifecycle and data evolution are realized by way of query languages that, as for DACMAS [36], guarantee decidability when certain constraints are met. For commitment-based platforms, the Cupid [17] language would provide analogous features.

From an ontological perspective, Guarino and Guizzardi [28, 29] discussed the importance of relationship reification and its connection with events/ processes. This work provides further foundation to our vision in connection to the specification of the conceptual model of an environment, seen as an information system. What this proposal currently lacks of is a methodology that will help designers to specify conceptual models. The literature on Agent-Oriented Software Engineering, on the other hand, proposes many methodologies. Briefly, SODA [35] is an agent-oriented methodology for the analysis and design of agent-based systems, adopting a layering principle and a tabular representation. It focuses on inter-agent issues, like the engineering of societies and environment for MAS, and relies on a meta-model that includes both agents and artifacts. GAIA [52] is a methodology for developing a MAS as an organization. Tropos [12] is a requirements-driven methodology for developing multiagent systems, while [23, 4] allow building declarative business process specifications in a norm-oriented fashion, see for instance. Although the last two methodologies do not consider data lifecycles in general, but rather rely on commitments and constraints, they are good candidates for extensions to a vision where norms, that capture the evolution of data, are composed into the specifications of multiagent system that are data and norm-aware. One viable direction to reach this purpose is to gather from the proposal in [44] for the realization of norm-governed socio-technical systems. Suitable methodologies should also be provided for programming the agents. In this respect, a starting point could be CoSE [2], a commitment-driven methodology for programming agents in presence of business-artifacts, that reify relationships captured as commitments.

6 Final Remarks

In this work, we have discussed the need for data-aware and norm-aware multiagent systems. In particular, we identified the importance of providing norm-based representations of the data lifecycles and of specifying the conceptual model of the underlying information system in terms of such norms. We have also commented some recent works that, independently, move along this direction facing one aspect or another.

One of the reasons of going towards data- and norm-awareness is the conviction that this will bring benefits to the design and implementation of software. The capability given to agents to take into account the data lifecycles in their reasoning process will provide the capability of reasoning about abnormal conditions in the environment, and decide how to react to them. This will enrich the already available capability agents can be equipped with of reasoning about deviations from their expected behavior. So, in principle, the robustness of the

system, intended as the ability to react appropriately to abnormal conditions, would be increased. The fact that data structure and lifecycles are explicitly represented in a way that can be reasoned about makes also the agents and their environment more decoupled, reducing the need of customizing agent programs when the environment changes. This increases both the extendibility and the reusability of all the components of the MAS. Last but not the least, data-awareness joint with a norm-based representation will enable a fully fledged range of verifications, and will also help modularizing the verification of properties inside a MAS, with a positive impact on the correctness of software.

Acknowledgements. The authors would like to thank the anonymous reviewers for the helpful comments. This work was developed during the sabbatical year that Matteo Baldoni and Cristina Baroglio spent at the Free University of Bolzano-Bozen. It was partially supported by the *Accountable Trustworthy Organizations and Systems (AThOS)* project, funded by Università degli Studi di Torino and Compagnia di San Paolo (CSP 2014).

References

1. Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services*. Springer, 2004.
2. Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Empowering agent coordination with social engagement. In Marco Gavanelli, Evelina Lamma, and Fabrizio Riguzzi, editors, *AI*IA 2015, Advances in Artificial Intelligence - XIVth International Conference of the Italian Association for Artificial Intelligence, Ferrara, Italy, September 23-25, 2015, Proceedings*, volume 9336 of *Lecture Notes in Computer Science*, pages 89–101. Springer, 2015.
3. Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Leveraging Commitments and Goals in Agent Interaction. In D. Ancona, M. Maratea, and V. Mascardi, editors, *Proc. of XXX Italian Conference on Computational Logic, CILC*, 2015.
4. Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti, and Federico Capuzzimati. Engineering commitment-based business protocols with the 2CL methodology. *JAAMAS*, 28(4):519–557, 2014.
5. Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi. A computationally-grounded semantics for artifact-centric systems and abstraction results. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 738–743. IJCAI/AAAI, 2011.
6. Kamal Bhattacharya, Nathan S. Caswell, Santhosh Kumaran, Anil Nigam, and Frederick Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems Journal*, 46(4):703–721, 2007.
7. Guido Boella and Leendert W. N. van der Torre. Regulative and constitutive norms in normative multiagent systems. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 255–266. AAAI Press, 2004.

8. Guido Boella, Leendert W. N. van der Torre, and Harko Verhagen. Introduction to normative multiagent systems. In Guido Boella, Leendert W. N. van der Torre, and Harko Verhagen, editors, *Normative Multi-agent Systems, 18.03. - 23.03.2007*, volume 07122 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
9. Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747 – 761, 2013.
10. Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, 2007.
11. Michael E. Bratman. What is intention? In P. Cohen, J. Morgan, and M. Pollack, editors, *Intensions in Communication*, pages 15–31. MIT Press, Cambridge, MA, 1990.
12. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
13. David M. Bridgeland and Ron Zahavi. *Business Modeling: A Practical Guide to Realizing Business Value*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.
14. Diego Calvanese, Giuseppe De Giacomo, and Marco Montali. Foundations of data-aware process analysis: a database theory perspective. In Richard Hull and Wenfei Fan, editors, *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 1–12. ACM, 2013.
15. Amit K. Chopra and Munindar P. Singh. Constitutive interoperability. In *Proceedings of the 7th Int. J. Conf. on Autonomous agents and multiagent systems, Volume 2*, pages 797–804. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
16. Amit K. Chopra and Munindar P. Singh. Agent communication. In Gerhard Weiss, editor, *Multiagent Systems, 2nd edition*. MIT Press, 2013.
17. Amit K. Chopra and Munindar P. Singh. Cupid: Commitments in relational algebra. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2052–2059. AAAI Press, 2015.
18. David Cohn and Richard Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *IEEE Data Eng. Bull.*, 32(3):3–9, 2009.
19. Natalia Criado, Estefania Argente, Pablo Noriega, and Vicent Botti. Reasoning about constitutive norms in bdi agents. *Logic Journal of IGPL*, 2013.
20. Mehdi Dastani, Leendert van der Torre, and Neil Yorke-Smith. Commitments and interaction norms in organisations. *J. Autonomous Agents and Multiagent Systems*, pages 1–43, 2015.
21. Maiquel de Brito, Jomi Fred Hübner, and Olivier Boissier. A conceptual model for situated artificial institutions. In Nils Bulling, Leendert W. N. van der Torre, Serena Villata, Wojtek Jamroga, and Wamberto Weber Vasconcelos, editors, *Computational Logic in Multi-Agent Systems - 15th International Workshop, CLIMA XV, Prague, Czech Republic, August 18-19, 2014. Proceedings*, volume 8624 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2014.
22. Yves Demazeau. From interactions to collective behaviour in agent-based systems. In *Proceedings of the 1st. European Conference on Cognitive Science*, pages 117–132, Saint-Malo, 1995.

23. Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolving cross-organizational business processes. *ACM Trans. Softw. Eng. Methodol.*, 19(2), 2009.
24. Antonio Di Leva and Salvatore Femiano. The BP-M* Methodology for Process Analysis in the Health Sector. *Intelligent Information Management*, 3(2):56–63, 2011.
25. Mark d'Inverno, Michael Luck, Pablo Noriega, Juan A. Rodríguez-Aguilar, and Carles Sierra. Communicating open systems. *Artif. Intell.*, 186:38–94, 2012.
26. Jack P. Gibbs. Norms: The problem of definition and classification. *American Journal of Sociology*, 70(5):586—594, 1965.
27. Guido Governatori. Law, logic and business processes. In *Third International Workshop on Requirements Engineering and Law, RELAW 2010, Sydney, NSW, Australia, September 28, 2010*, pages 1–10. IEEE, 2010.
28. Nicola Guarino and Giancarlo Guizzardi. "We Need to Discuss the Relationship": Revisiting Relationships as Modeling Constructs. In Jelena Zdravkovic, Marite Kirikova, and Paul Johannesson, editors, *Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings*, volume 9097 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2015.
29. Nicola Guarino and Giancarlo Guizzardi. Relationships and events: Towards a general theory of reification and truthmaking. In *AI*IA 2016, Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, December 2016, Proceedings*, Lecture Notes in Computer Science. Springer, 2016. to appear.
30. Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular ACTOR formalism for artificial intelligence. In Nils J. Nilsson, editor, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, August 1973*, pages 235–245. William Kaufmann, 1973.
31. Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.
32. Andrew J.I. Jones and José Carmo. Deontic logic and contrary-to-duties. In Dov Gabbay, editor, *Handbook of Philosophical Logic*, page 203–279. Kluwer, 2001.
33. Bertrand Meyer. *Object-oriented Software Construction (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.
34. John C. Mitchell. *Concepts in programming languages*. Cambridge University Press, Cambridge, New York (N. Y.), 2002.
35. Ambra Molesini, Andrea Omicini, Enrico Denti, and Alessandro Ricci. SODA: A roadmap to artefacts. In *Engineering Societies in the Agents World VI*, volume 3963 of *LNAI*, pages 49–62. Springer, 2006. 6th Int. Workshop (ESAW 2005).
36. Marco Montali, Diego Calvanese, and Giuseppe De Giacomo. Verification of data-aware commitment-based multiagent system. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 157–164. IFAAMAS/ACM, 2014.
37. Rumyana Neykova and Nobuko Yoshida. Multiparty Session Actors. In eva Kühn and Rosario Pugliese, editors, *Coordination Models and Languages - 16th IFIP WG 6.1 International Conference, COORDINATION 2014, Held as Part of the 9th International Federated Conferences on Distributed Computing Techniques, DisCoTec 2014, Berlin, Germany, June 3-5, 2014, Proceedings*, volume 8459 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2014.

38. Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, December 2008. Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.
39. Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, 2011.
40. Alessandro Ricci and Andrea Santi. From Actors and Concurrent Objects to Agent-Oriented Programming in simpAL. In Gul A. Agha, Atsushi Igarashi, Naoki Kobayashi, Hidehiko Masuhara, Satoshi Matsuoka, Etsuya Shibayama, and Kenjiro Taura, editors, *Concurrent Objects and Beyond - Papers dedicated to Akinori Yonezawa on the Occasion of His 65th Birthday*, volume 8665 of *Lecture Notes in Computer Science*, pages 408–445. Springer, 2014.
41. Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.
42. Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, March 1993.
43. Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
44. Munindar P. Singh. Norms as a basis for governing sociotechnical systems. *ACM TIST*, 5(1):21, 2013.
45. Munindar P. Singh and Michael N. Huhns. *Service-oriented computing - semantics, processes, agents*. Wiley, 2005.
46. Samira Tasharofi, Peter Dinges, and Ralph E. Johnson. Why Do Scala Developers Mix the Actor Model with Other Concurrency Models? In *Proceedings of the 27th European Conference on Object-Oriented Programming*, ECOOP'13, pages 302–326, Berlin, Heidelberg, 2013. Springer-Verlag.
47. Pankaj R. Telang, Munindar P. Singh, and Neil Yorke-Smith. Relating Goal and Commitment Semantics. In *Post-proc. of ProMAS*, volume 7217 of *LNCS*. Springer, 2011.
48. Göran Therborn. Back to norms! on the scope and dynamics of norms and normative action. *Current Sociology*, 50:863–880, 2002.
49. Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.
50. Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. *JAAMAS*, 14(1):5–30, 2007.
51. Michael J. Wooldridge. *Introduction to multiagent systems, 2nd edition*. Wiley, 2009.
52. Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.