Governing Agents on the Web (Blue Sky Ideas)

Victor Charpenay[1][0000−0002−9210−1583], Matteo Baldoni[2][0000−0002−9294−0408], Andrei Ciortea[3][0000−0003−0721−4135], Stephen Cranefield[4][0000−0001−5638−1648], Julian Padget[7][0000−0003−1314−2094], and Munindar P. Singh[8][0000−0003−3599−3893]

[1] Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, France, victor.charpenay@emse.fr
[2] University of Torino, Italy, matteo.baldoni@unito.it
[3] School of Computer Science, University of St.Gallen, Switzerland, andrei.ciortea@unisg.ch
[4] University of Otago, NZ, stephen.cranefield@otago.ac.nz
[5] University of Bath, UK, j.a.padget@bath.ac.uk
[6] North Carolina State University, USA, mpsingh@ncsu.edu

Abstract. This paper summarizes the findings of the working group "Governance" at the Dagstuhl seminar on *Agents on the Web*, which was held in February 2023. This paper seeks to identify research challenges relating to governance and web architecture.

1 Introduction

There are major similarities in the motivations behind multi-agent systems (MAS) and the Web. Both disciplines and practices seek to advance decentralization and openness in that ideally there isn't a single locus of control and participants can behave and interact broadly autonomously under local control.

This paper addresses the interplay of multi-agent systems with the Web. Specifically, it concerns how constructs and techniques identified in the study of the governance of MAS be realized over the Web architecture and how the governance of the Web be beneficially structured based on constructs and techniques developed for the governance of MAS. In simple terms, it seeks to identify synergies in both these directions:

- What does the Web offer to support the governance of MAS?
  We anticipate ways to use the interoperability and scalability of the Web to build easy-to-use, widely deployed MAS. Scalability, evolvability, and visibility are important non-functional requirements that are "guaranteed" if a system's architecture relies on the Web.
- What does the governance of MAS offer the Web?
  We anticipate approaches for governance that provide flexibility and local control with formal models that support correctness and generality going

beyond the typically procedural kinds of governance seen on the Web today. A challenge here is to map abstract models for governance in MAS to Web components, in a way that preserves Web architectural constraints and thereby guarantees the associated non-functional requirements of scalability and decoupling.

The main contribution of this paper is an identification of crucial challenges pertaining to the interplay of MAS and the Web, and the formulation of some initial research questions that might guide future research on this topic.

2 Key Concepts and Considerations

We understand the Web as a combination of uniform resource identifiers (URI) supporting hyperlinked representations, along with a computational architecture that supports locating and accessing the identified resources. The computational architecture is based on standard protocols for manipulating resources (e.g., HTTP, CoAP). We think here of architectural constraints (such as for caching, layering, and uniform interaction) as captured by the original design rationale behind the Web Architecture [18]. The W3C Recommendation for the Web Architecture [22] provides additional information on identification, interaction, and representation of resources on the Web. Linked Data principles [21] characterize the relevant constraints, which can be supplemented by ontology specification [37] in general. Some extensions to the above computational and information architectures, such as through the observer pattern (implemented in CoAP [30]) and local state transfer [33], aim at going beyond the "Web of Documents" [19].

There exists numerous studies that relate the Web with MAS, following two general trends. Some studies focus on applying RDF and Linked Data to expose agents to hypermedia-driven environments [15, 9, 27, 4]. Other studies combine a formal declarative model of norms [11, 34, 6] to specify social protocols [7]. Such social protocols provide a more thoroughly decentralized conception of Berners-Lee's [1] notion of social machines. Recent W3C Recommendations for the Social Web, including Linked Data Notifications, ActivityStream, ActivityPub and WebSub [20], offer ways to join these two trends, by implementing social protocols over Linked Data.

Dimensions that are common to all MAS architecture, such as the environment, organization and interaction dimensions, are defined at a higher level of abstraction than Web resources and protocols. Constraints such as caching, layering and uniform interaction apply to components, exposing a certain functionality through ports. Web components may only have client or server ports, exchanging messages in a standard protocol. Components with client ports only are called origin clients, those with server ports only are origin servers and a third kind of components, proxies, have an equal number of client ports and server ports, forwarding requests from clients to servers or vice-versa [18].

To be able to analyse the interplay between MAS and the Web, a mapping from MAS abstractions to (more concrete) Web components is necessary. Given the complexity of both fields, there is no trivial mapping and most likely not a

unique mapping across the two levels of abstraction. In the following, we perform a case study to help identify mappings that would preserve effective governance mechanisms developed in MAS research.

3 Case Study: Organ Allocation

As a case study, we consider a simplified version of the process for allocating donated bodily organs to potential transplant recipients [36]. Due to the scarcity of organ donors, the limited period of organ viability after removal from the donor, and the desire to maximise the chances of a successful outcome, national bodies such as the United Network for Organ Sharing (UNOS)[7], in the United States, or the Organización Nacional de Trasplantes (ONT)[8], in Spain, have been formed to manage organ waiting lists, match donors with recipients, and develop and monitor policies governing this process.

3.1 Carrel Revisited

Vázquez-Salceda et al. [36] present the design of Carrel, using an agent-mediated electronic institution (e-institution) [25] for the organ allocation process. This design specifies in a formal manner [16] the structure of the interactions between hospitals, tissue banks, and institutional agents managing the process, as well as the norms that govern these interactions and the match between a donor and recipient. The distribution of organs and tissues in Carrel, given its Spanish congtext, would be overseen by the Spanish ONT, together with the Catalan Organitzaciò CATalana de Trasplantament (OCATT)[9].

The Carrel platform is conceived to model the servicing of hospitals by connecting them with sources of tissues and organs, where the goal is to provide the *best* matched organ or tissue across all the sources registered with the platform. The interpretation of *best* is complicated and depends on a variety of factors that change over time, particularly in the case of organs. A strong requirement for a capacity for evolution motivated the development using an e-institution, made concrete through explicit computable norms. Because capacity for evolution is one of the main properties of the Web, Carrel offers an interesting case study through which to start motivating our research questions.

Tissue distribution is essentially demand-driven because tissues can be preserved and stored over extended periods with no significant degradation. Organ distribution is essentially supply-driven because the need is known before a suitable part becomes available. For the purposes of this paper, we only consider organ distribution where the need is known before availability.

Each hospital interfaces with Carrel through its Transplant Coordination Unit (TCU). A surgeon can request an organ or tissue via this unit which leads to the creation of an agent whose task it is to join the Carrel institution to

[7] https://unos.org/
[8] https://www.ont.es
[9] https://trasplantaments.gencat.cat

obtain an organ or tissue that satisfies the surgeon's requirements. The requirements include the urgency of the request, hospital authentication information, organ/tissue data, recipient data, and a set of constraints on the organ or tissue.

To achieve its goal, the requesting agent must negotiate with other agents representing hospitals with potential donors[10] All agents are subject to behavioral norms that, if violated, are sanctioned. In the original version of Carrel, which was based on the ISLANDER framework [17], participating agents are in effect regimented by so-called governor agents, to prevent non-compliance. Here, we make the more general assumption that agents are regulated by norms and may choose to take non-compliant actions. Thus, we assume that agent actions in this contemporary Carrel are all visible to the regulatory bodies, as is the case in the physical world: hospital TCUs communicate their requests to members of ONT/OCATT, who then contact other hospitals to find a donor and select the best match.

In the course of the negotiation, hospitals may be tempted to behave unfairly towards others. They may want to provide incorrect information to ONT/OCATT to maximize their chances of finding a matching donor. The sanctioning power of ONT/OCATT is therefore essential. If the Carrel institution is to be cast as an (institutional) environment deployed on the Web, this sanctioning power should be retained by the environment. For the purposes of this paper, we consider a simple norm guaranteeing fair allocation of donors: *a requesting agent must not accept a organ/tissue if it was previously offered a better match* (with respect to criteria defined by the institution). The associated sanction is that the institution may reject any future request made by the violating agent.

3.2 Norms and Roles

Following one approach from the MAS literature [5], we can capture the sociotechnical system requirements in terms of accountability [8] and only then proceed to identify the information exchanges between the agents and from there their individual actions.

For concreteness, in the example below, we adopt the Custard notation [6]. The language is based on a conventional relational model. We can think of each predicate, such as `Registered` and `Certify`, as mapping to a relation and containing event instances. Additional relations are computed from the norm semantics: these include `violated(AcceptBestMatch)`, which refers to the event of the `AcceptBestMatch` norm being violated. In Listing 1, the concerned roles are institution and hospital. A computational entity representing the stakeholder playing the institution role is empowered to revoke the certification of an agent representing the stakeholder playing a hospital role. The power is instantiated when a hospital is registered and applies when it violates the `AcceptBestMatch` norm.

[10] in its original form, Carrel includes a waiting list of donors, which other TCU agents can consult once a donor is available. Here, we only consider agent-to-agent interactions and assume that the agent holding the request asks other agents for potential donors.

Listing 1. OCATT's power to revoke certification in Custard. Here, "power" is the norm type, "DecertifyPower" is its schema name, the IDs refer to the parties concerned with the "by" indicating who has power over whom; "create" is the event with which the schema is instantiated, "detach" is the event under which it goes into effect, and "discharge" is the event that describes what the power brings about. As in SQL, the attribute (column) names are elided but the matches (joins) take place based on their values.

```
power DecertifyPower hospitalID by institutionID
 create Registered
 detach violated(AcceptBestMatch)
 discharge not Certify
```

3.3 Allocation Protocol

We provide an overview of the (simplified) negotiation protocol followed by TCU agents. The agent holding a surgeon's request for transplantation sends its request to another agent, which answers with a donation offer. The requesting agent has then the choice of confirming or rejecting the offer, presumably depending on the strength of this match vis à vis any other offers it may have received from other prospective donor hospitals.

Listing 2 shows a simple protocol in the Blindly Simple Protocol Language (BSPL) language [32]. Some message parameters are adorned *out*, meaning the sender can set them freely (constrained only by key integrity); the sending of a message generates a binding for each such parameter. Other parameters are adorned *in*, meaning the sender must know the binding prior to sending the message, which means it must have obtained the binding from a previous message it sent or from a message it received from another agent. In particular, the requesting agent must obtain a certificate to confirm or reject an offer. This certificate must be obtained by some system component that embodies ONT/O-CATT, which may or may not deliver it, depending on prior violations made by the agent. The protocol of Listing 2 assumes the certificate is known prior to this protocol being enacted, which fact is indicated by the *in* adornment on the parameter `recipientCertificate`.

Listing 2. A possible *Organ Donation* protocol.

```
Donation {
  roles RecipientH, DonorH
  parameters out organRequestID key, out offerID key, out recipientInfo, out
      donorInfo, in recipientCertificate
  private decision, transferInfo

  RecipientH -> DonorH: request[out organRequestID, out recipientInfo]
  DonorH -> RecipientH: response[in organRequestID, out offerID, out
      donorInfo]

  RecipientH -> DonorH: confirmation[in organRequestID, in offerID, in
      recipientCertificate, out transferInfo, out decision]
  RecipientH -> DonorH: rejection[in organRequestID, in offerID, in
      recipientCertificate, out decision]
}
```

The norm of `AcceptBestMatch` would apply to how this protocol is enacted and the norm `DecertifyPower` builds on top of `AcceptBestMatch`. To

give `DecertifyPower` teeth, the information architecture must be such that the requisite information is available to the concerned party (i.e., the institution).

3.4 Norm Encoding and Monitoring

Over the last 30 years, researchers in MAS have proposed many approaches to reasoning about norms as well as computational approaches to monitoring a MAS for norm violations [31, 12, 2]. A crucial aspect of this work is to provide a formal representation of norms (see, e.g., [14] for an overview of approaches). Here we choose one possible representation of the `AcceptBestMatch` norm for illustrative purposes (Listing 3). Using the Expectation Event Calculus (EEC) [13], we can express the norm as a conditional rule of expectation, i.e. one that introduces a constraint on the future, that should be monitored for fulfilment or violation, once a condition is satisfied.

Listing 3. `AcceptBestMatch` orm expressed in the Expectation Event Calculus (EEC).

```
exp_rule(
  and([request(OrganRequestID, RecipInfo),
       response(OrganRequestID, OfferID1, DonorInfo1),
       quality_of_match(DonorInfo1, RecipInfo, Q1)]),
  never(
    and([response(OrganRequestID, OfferID2, DonorInfo2),
         quality_of_match(DonorInfo2, RecipInfo, Q2),
         Q2 < Q1,
         eventually(confirmation(OrganRequestID, OfferID21, _, _, _))]))))
```

The first argument of this rule checks the record of messages, assumed to be recorded as event calculus 'fluents', for the existence of an organ request and offer that match with a quality `Q1`. If that condition is satisfied, an instantiation of the second argument is created within an expectation fluent, stating that it should never be the case that another offer with a lower match quality has been made then (eventually) confirmed by the recipient.

The EEC engine will track this expectation over time as new information arrives, and will create a violation event if such a confirmation occurs. Further reasoning with related norms may conclude that an associated sanction should be applied or that a compensating action should be performed. Here, the EEC engine is an example of an active component of an MAS governance system that should be accommodated within any architecture for governance of agents on the Web.

4 Candidate Architectures

On the Web, "effective" power is held on the server side: servers may hide information, redirect requests or reject them, thereby reducing the action space of agents. It is natural to think of institutions as components with a server port, receiving requests from the client port of agents, such as described in [28]. Several candidate architectures offer various level of control, however. We have identified four classes of components for institutions (see Table 4).

Table 1. Mappings of an institution to Web components and associated characteristics with respect to governance; checkmark (✓): the characteristic of the institution is guaranteed by constraints on the Web component, question mark (?): the characteristic of the institution depends on the MAS architecture, not on the Web component

Institutional component	Norm update	Statefulness	Sanctioning
Read-only server	✓		
Read-write server	✓	✓	
Proxy	✓	✓	✓
Servient	✓	?	?

Autonomous agents differ from classical user agents in the Web architecture (i.e., browsers). In particular, an autonomous agent might require both client and server roles simultaneously and can have one-sided elementary interactions, where they are not awaiting a response. The requirements of autonomous agents are closer to the ones of servients in the W3C Web of Things Architecture [24] lingo, which are components with both client and server roles that can interact in a peer-to-peer manner. An agent could be implemented as such a component, or it could be a process that runs in a runtime environment provided by such a component. If the agent is visible to other agents, i.e. if its representation is dereferenceable, the representation can point, for instance, to a Linked Data Notification inbox that receives messages from other agents [3].

In a typical MAS, an institution may not materialize as a Web component at all. If norms are defined at design time, agents are guaranteed to behave (in general) as per these norms. Certification may also occur at design time, such that an agent either uses a single certificate throughout the system's lifetime or periodically renews its certificate as long as its behavior specification does not change. In the above configuration, however, the institution has no sanctioning power at run time. It cannot easily redesign the normative framework in which agents interact either. Updating a norm would potentially require modifying the behavior of all agents at the same time.

4.1 Institution as Read-Only Server

Figure 1 illustrates one possible architecture. The behavior of an agent may easily be decoupled from the normative framework that regulates it, though: if norms are exposed by a read-only (origin) server, an agent may dereference the norms from time to time and internalize whatever formal specifications the server returns.

In this alternative configuration, the institutional component gains the power of dictating norms and changing them at run time. The ability of agents to get a certificate may depend on the fact they are aware of the latest version of the normative framework.

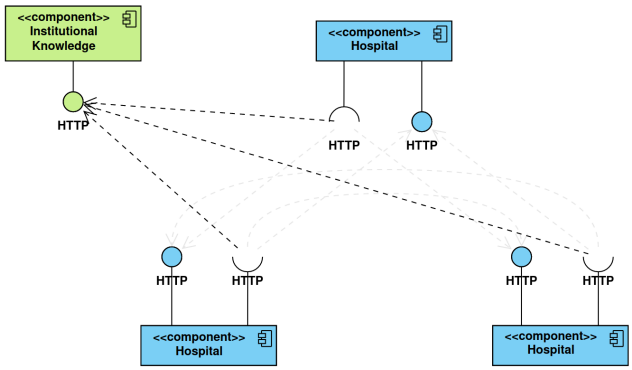

Fig. 1. Architecture for an institution governing Web agents: Institution as Read-Only Server

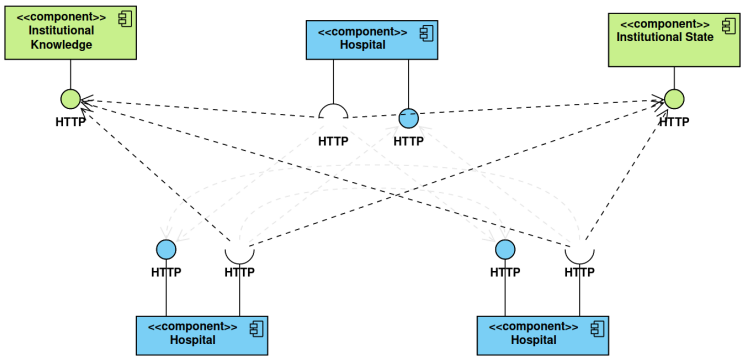

Fig. 2. Architecture for an institution governing Web agents: Institution as Read-Write Server

4.2 Institution as Read-Write Server

In order for the institution to gain sanctioning power, another component may manage its real-time state.

The state of the institution includes the level of obedience of each participating agent, which is directly derived from the confirmations/rejections they generate. To be able to maintain its state, the new institutional component must be able to observe each agent-to-agent interaction. For instance, the certificate may be signed not for an agent but for a pair (agent, donation offer ID), forcing agents to request a new certificate every time they make a decision. If the certification server stores a history of confirmations/rejections, it effectively becomes an institutional component that is capable of deciding in real time whether agents violate norms and, if they do, to sanction them by rejecting their certification request.

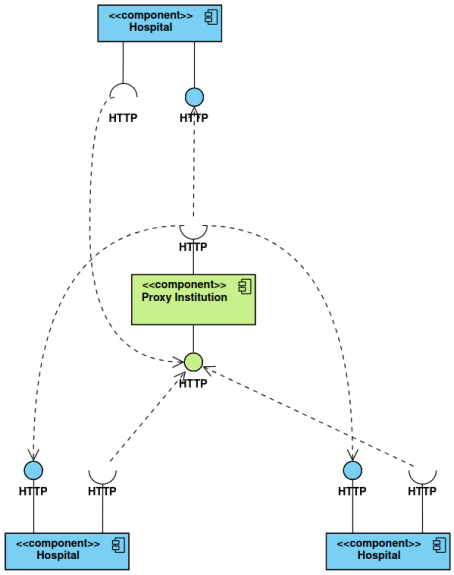

Fig. 3. Architecture for an institution governing Web agents: Institution as Proxy

The institutional server would become a stateful read-write component, as agents, through their certification requests, change the state of the overall institution. Yet, it remains a purely reactive component, with a single server port.

4.3 Institution as Proxy

In the above configuration, the institutional component has no knowledge of how agents negotiate. If the institution is to be omniscient, another kind of component should be used. On the Web, it is common to use proxy servers to monitor activity.

The main architectural constraint over proxies is that they have a client port and a server port, such that incoming requests (on the server port) are either immediately responded to or forwarded to another server, possibly after a rewriting step.

In our working example, the institutional server may be replaced by a proxy without modifying in any way the behavior of agents. Agents send requests to the proxy, which can keep track of negotiations and add a certificate on-the-fly if the requesting agent behaves properly. The proxy may also turn a confirmation into a rejection, to sanction any misbehaving agent. The requesting agent receives feedback on the sanction through the other agent (which acknowledges the rejection, instead of the initial confirmation).

4.4 Institution as Servient

An alternative is to capture the institution as an explicit stakeholder supported by an agent on par with the other parties in the system. The institution becoming both reactive and proactive, it must include independent client and server ports and becomes *a minima* a servient.

In ordinary operations, this agent may have little to say beyond conveying institutional norms and facts as in the previous approach. (In some real-life cases, the institutional agent is the same as a member of the system taking on that additional role.) However, by identifying this institutional entity, we make it subject to accountability. As a result, we can more perspicuously model the accountability relationships between the concerned parties than otherwise. A party can also question the institution, for example, if they fail to receive an organ in a timely fashion. This process may result in the institutional facts being disputed and adjudicated [35] and the norms potentially revised.

If the institution is embodied by a agent, its monitoring and sanctioning power doesn't depend on architectural constraints (at the level of Web components) but on the behavior of other agents (at the level of MAS abstractions).

5 Hypermedia-driven Interaction

In the previous section, we discussed several alternatives for implementing institutions as Web components. To promote interoperability on the open Web, agents should be agnostic to such implementation details. This can be achieved by hiding the specifics of a configuration behind a (semantic) hypermedia layer.

To this end, hypermedia-driven interaction can support autonomous agents to interact with Web resources in a uniform way while being decoupled from the underlying components. Most prominently, this approach is used on the Web of Things to decouple clients from Web-enabled devices by hiding the interfaces used to access the devices behind abstract interaction possibilities and hypermedia controls. To illustrate how this works, an HTML page typically provides the user with a number of action possibilities, such as navigating to a different page by clicking a hyperlink or sending an order by filling out and submitting an HTML form. Performing any such action transitions the user to a new page and exposes a new set of possible actions. In each step, the user's browser retrieves not only an HTML representation of the current page from a server but also the hypermedia controls required to transition to new pages. Retrieving all this information through hypermedia allows websites to evolve without impacting the browser, and allows the browser to transition seamlessly across components. Hypermedia-driven interaction reduces coupling between components (e.g., browsers, proxies, origin servers) and allows them to be deployed and to evolve independently from one another—a central feature that allowed the Web to scale up to the size of the Internet.

In the context of an institution on the open Web, the various action possibilities—such as retrieving formal specifications of norms, requesting a certificate, or sending messages to other agents—can be made available to the agents

through hypermedia controls. Such hypermedia controls would encapsulate all the information required by an agent to interact with the component(s) that implement the institution, but to use the hypermedia controls in a reliable manner the agent would have to operate on an abstract model of the institution. For example, if an agent is required to obtain a certificate at run time to enact the *Organ Donation* protocol in Section 3.3, the agent could discover such an action possibility through hypermedia—but the agent would have to be aware of the notion of a certificate and should have the ability to request a certificate at run time.

In long-lived Web-based MAS, the institutional model could also evolve throughout the agents' lifetime—for example, from using norms and certificates defined at design time to a model based on an evolving set of norms [29] and certificates that have to be obtained at run time. Such an evolution would be reflected in the hypermedia environment through the set of action possibilities provided to agents at run time—and thus, to cope with this evolution, the agents would have to adapt by synthesizing a new course of action that meets their design objectives. While engineering such agents is still an open challenge, some related work investigates the design of agents able to plan and adapt to dynamic hypermedia environments (e.g., see [23, 10]).

6 Discussion: Research Questions and Challenges

The main contribution of this paper is in identifying some interesting research questions that can motivate research on the interface of MAS and Web architectures. Specifically, we propose the following research questions:

How should we model the presence of a governance layer in a MAS? In most common approaches, the governance layer is either implicit (agents are regimented) or embodied by norm-enforcement agents. In the Custard language for instance (Listing 1), institutions and agents are at the same level of abstraction. An alternative approach would be to model an *institutional* environment, as a generalised form of agent environment (see [38, 26]), such that reusable server or proxy components may be applied on several MAS architectures.

What aspects of a web-based deployment of a MAS may be subject to governance policies and included in an institutional environment? Examples would include constraints on ownership and physical location of the component hosting an agent. A MAS that operates on a physical (or simulated) environment, e.g. on the Web of Things, must necessarily be regulated by stateful institutional components. Statefulness alone is however not enough, as institutional facts may be derived from brute facts generated in the physical environment. Only proxies and servients—components with a client port—could access the physical environment—exposed by environmental servers. If servers implement the observer pattern, as in CoAP, another class of institutional components may be added for "observer" servients. Observer servients would not be proactive (i.e. not agents) but still have the possibility to subscribe to environmental events.

How do we map the required properties of an institutional environment (monitoring, reasoning, sanctioning power) to constraints and mechanisms of a web-based deployment and more specifically to hypermedia control? Governance requires a certain point where governance decisions are made. Even in a decentralized architecture, that point reflects a form of centrality, albeit a weak one. Institutional components, such as the ones we have identified in the paper, have decreasing levels of centrality:

- proxy (to enforce policies), which supports real-time sanctions and requires no effort to continually re-engineer agents.
- institutional server (to maintain a shared institutional state), which supports violation detection and sanctioning but lets agents autonomously interact for the most part.
- federation of institutional servers, which relaxes the single institutional server case thereby reducing centrality but supporting potentially delayed sanctioning.

As suggested in Sec. 5, hypermedia controls, such as links and forms, should help agents follow high-level social protocols while still deviating locally, depending on the controls that servers expose. Hypermedia controls make it easy to decentralize interactions (a proxy may e.g. redirect to a read-write server at run time) but hard to recentralize interactions (if agents interact in a peer-to-peer fashion, moving to proxy-mediated interactions would require to re-engineer all agents).

7 Conclusion

Thinking about MAS and the Web together opens up new opportunities in building large-scale sociotechnical systems. Such systems would take advantage of the flexibility derived from MAS and the scalability and familiarity (to most developers) derived from the Web. The possibilities are promising and we invite the research community to join us in investigating them.

References

1. Berners-Lee, T.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. Harper Business, New York (1999)
2. Boella, G., van der Torre, L.W.N., Verhagen, H.: Introduction to normative multiagent systems. Comput. Math. Organ. Theory **12**(2-3), 71–79 (2006). https://doi.org/10.1007/s10588-006-9537-7
3. Capadisli, S., Guy, A.: Linked data notifications (2017), https://www.w3.org/TR/ldn/
4. Charpenay, V., Käfer, T., Harth, A.: A unifying framework for agency in hypermedia environments. In: Alechina, N., Baldoni, M., Logan, B. (eds.) Engineering Multi-Agent Systems. pp. 42–61. Springer International Publishing (2022)

5. Chopra, A.K., Dalpiaz, F., Aydemir, F.B., Giorgini, P., Mylopoulos, J., Singh, M.P.: Protos: Foundations for engineering innovative sociotechnical systems. In: Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE). pp. 53–62. IEEE Computer Society, Karlskrona, Sweden (Aug 2014). https://doi.org/10.1109/RE.2014.6912247
6. Chopra, A.K., Singh, M.P.: Custard: Computing norm states over information stores. In: Proceedings of the 15th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS). pp. 1096–1105. IFAAMAS, Singapore (May 2016). https://doi.org/10.5555/2936924.2937085
7. Chopra, A.K., Singh, M.P.: From social machines to social protocols: Software engineering foundations for sociotechnical systems. In: Proceedings of the 25th International World Wide Web Conference. pp. 903–914. ACM, Montréal (Apr 2016). https://doi.org/10.1145/2872427.2883018
8. Chopra, A.K., Singh, M.P.: Accountability as a foundation for requirements in sociotechnical systems. IEEE Internet Computing (IC) **25**(6), 33–41 (Sep 2021). https://doi.org/10.1109/MIC.2021.3106835
9. Ciortea, A., Boissier, O., Ricci, A.: Engineering world-wide multi-agent systems with hypermedia. In: Weyns, D., Mascardi, V., Ricci, A. (eds.) Engineering Multi-Agent Systems. pp. 285–301. Springer International Publishing, Cham (2019)
10. Ciortea, A., Mayer, S., Michahelles, F.: Repurposing manufacturing lines on the fly with multi-agent systems for the web of things. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. p. 813–822. AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2018)
11. Cliffe, O., De Vos, M., Padget, J.: Specifying and reasoning about multiple institutions. In: COIN 2006. pp. 63–81 (2007). https://doi.org/10.1007/978-3-540-74459-7˙5
12. Conte, R., Falcone, R., Sartor, G.: Introduction: Agents and norms: How to fill the gap? Artif. Intell. Law **7**(1), 1–15 (1999). https://doi.org/10.1023/A:1008397328506
13. Cranefield, S.: Agents and expectations. In: International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems. Springer (2013). https://doi.org/10.1007/978-3-319-07314-9˙13
14. Cranefield, S., Winikoff, M., Vasconcelos, W.: Modelling and monitoring interdependent expectations. In: Cranefield, S., van Riemsdijk, M.B., Vázquez-Salceda, J., Noriega, P. (eds.) Coordination, Organizations, Institutions, and Norms in Agent System VII. pp. 149–166. Springer Berlin Heidelberg (2012)
15. Dikenelli, O., Alatlı, O., Erdur, R.C.: Where are all the semantic web agents: Establishing links between agent and linked data web through environment abstraction. In: Weyns, D., Michel, F. (eds.) Agent Environments for Multi-Agent Systems IV, pp. 41–51. Springer International Publishing, Cham (2015)
16. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Meyer, J.J., Tambe, M. (eds.) Intelligent Agents VIII. Lecture Notes in Artificial Intelligence, vol. 2333, pp. 348–366. Springer Verlag (2001), iSBN 3-540-43858-0
17. Esteva, M., de la Cruz, D., Sierra, C.: ISLANDER: an electronic institutions editor. In: The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings. pp. 1045–1052. ACM (2002). https://doi.org/10.1145/545056.545069

18. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. **2**(2), 115–150 (May 2002). https://doi.org/10.1145/514183.514185
19. Fielding, R.T., Taylor, R.N., Erenkrantz, J.R., Gorlick, M.M., Whitehead, J., Khare, R., Oreizy, P.: Reflections on the REST architectural style and "principled design of the modern web architecture" (impact paper award). In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. p. 4–14. ESEC/FSE 2017, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3106237.3121282
20. Guy, A.: Social web protocols (2017), https://www.w3.org/TR/social-web-protocols/
21. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology **1**(1), 1–136 (Feb 2011). https://doi.org/10.2200/S00334ED1V01Y201102WBE001
22. Jacobs, I., Walsh, N.: Architecture of the world wide web, volume one. Tech. rep., World Wide Web Consortium (2004), https://www.w3.org/TR/webarch/
23. Kovatsch, M., Hassan, Y.N., Mayer, S.: Practical semantics for the internet of things: Physical states, device mashups, and open questions. In: 2015 5th International Conference on the Internet of Things (IOT). pp. 54–61 (2015). https://doi.org/10.1109/IOT.2015.7356548
24. Lagally, M., Matsukura, R., McCool, M., Toumura, K.: Web of things (wot) architecture 1.1. Tech. rep., World Wide Web Consortium (2023), https://www.w3.org/TR/wot-architecture/
25. Noriega, P.: Agent-Mediated Auctions: The Fishmarket Metaphor. No. 8 in IIIA Monograph Series, Institut d'Investigació en Intel.ligència Artificial (IIIA), Bellaterra, Spain (1997), phD Thesis
26. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems **17**(3), 432–456 (dec 2008). https://doi.org/10.1007/s10458-008-9053-x, https://doi.org/10.1007/s10458-008-9053-x
27. O'Neill, E., Lillis, D., O'Hare, G.M.P., Collier, R.W.: Delivering multi-agent microservices using cartago. In: Baroglio, C., Hubner, J.F., Winikoff, M. (eds.) Engineering Multi-Agent Systems. pp. 1–20. Springer International Publishing, Cham (2020)
28. Padget, J., De Vos, M., Page, C.A.: Deontic sensors. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 475–481. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/66, https://doi.org/10.24963/ijcai.2018/66
29. Savarimuthu, B.T.R., Cranefield, S.: Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. Multiagent and Grid Systems **7**(1), 21–54 (2011). https://doi.org/10.3233/MGS-2011-0167
30. Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). Tech. Rep. RFC 7252, Internet Engineering Task Force (IETF), Fremont, California (Jun 2014), proposed standard; https://tools.ietf.org/html/rfc7252
31. Shoham, Y., Tennenholtz, M.: On the synthesis of useful social laws for artificial agent societies (preliminary report). In: Swartout, W.R. (ed.) Proceedings of the 10th National Conference on Artificial Intelligence. pp. 276–281. AAAI Press / The MIT Press (1992)

32. Singh, M.P.: Information-driven interaction-oriented programming: Bspl, the blindingly simple protocol language. In: Sonenberg, L., Stone, P., Tumer, K., Yolum, P. (eds.) 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3. pp. 491–498. IFAAMAS (2011), http://portal.acm.org/citation.cfm?id=2031687\&CFID=54178199\&CFTOKEN=61392764
33. Singh, M.P.: LoST: Local State Transfer—An architectural style for the distributed enactment of business protocols. In: Proceedings of the 9th IEEE International Conference on Web Services (ICWS). pp. 57–64. IEEE Computer Society, Washington, DC (Jul 2011). https://doi.org/10.1109/ICWS.2011.48
34. Singh, M.P.: Norms as a basis for governing sociotechnical systems. ACM Transactions on Intelligent Systems and Technology (TIST) **5**(1), 21:1–21:23 (Dec 2013). https://doi.org/10.1145/2542182.2542203
35. Telang, P.R., Kalia, A.K., Madden, J.F., Singh, M.P.: Combining practical and dialectical commitments for service engagements. In: Proceedings of the 13th International Conference on Service-Oriented Computing (ICSOC). pp. 3–18. No. 9435 in Lecture Notes in Computer Science, Springer, Goa, India (Nov 2015). https://doi.org/10.1007/978-3-662-48616-0˙1
36. Vázquez-Salceda, J., Cortés, U., Padget, J., López-Navidad, A., Caballero, F.: The organ allocation process: A natural extension of the carrel agent-mediated electronic institution. AI Communications **16**(3), 153–165 (2003)
37. W3C: OWL 2 Web Ontology Language: Document overview (Dec 2012), https://www.w3.org/TR/owl2-overview/, w3C Recommendation. Accessed 2023-03-03
38. Weyns, D., Omicini, A., Odell, J.J.: Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems **14**, 5–30 (2007)