2.1.2 Agreement Computing and Self-Evolution

Nardine Osman and Carles Sierra

Abstract  Self-organisation and self-evolution is evident in physics, chemistry, biology, and human societies. Despite the existing literature on the topic, we believe self-organisation and self-evolution is still missing from the IT tools (whether online or offline) we are building and using. In the last decade, human interactions have been moving more and more towards social media. The time we spend interacting with others in virtual communities and networks is tremendous. Yet, the tools supporting those interactions remain rigid. This article argues the need for self-evolving software-enabled communities, and proposes a roadmap for achieving this required self-evolution via agreement computing. The proposal is based on building agreement-driven normative-based communities, where community interactions are regulated by norms and community members are free to discuss and agree on their community’s norms. The evolution of communities is then dictated by the evolution of its norms, which is driven by members agreeing on those norms.

2.1.2.1 Motivation

The concept of social self-organisation has deep roots in political theory. In the Leviathan, Hobbes expresses the idea that the social structure, the State or res publica, is the result of a social contract or pact. The social contract is an agreement among free individuals in order to surrender power to a central authority in order to get a guarantee of peace. In essence, a social contract is the implicit acceptance of self-limiting individual freedom in order to guarantee self-preservation and a better life. The concept of social contract implies that individuals reflect on the community governance in order to decide on the means to reach commonwealth.

This capacity of individuals to reflect and decide on community governance is missing in the IT tools we are currently building. How individuals interact is decided by groups of engineers that design tools, determine the individuals’ privacy levels, hinder some from performing certain tasks, and all this without the individuals having a say. We argue that it is time for us to take control of the
evolution of our own virtual communities, which we refer to as software-enabled communities – examples of such communities are those based on social networks or any other community whose management relies on software, such as online learning environments or open collaboration (e.g. open source software). To address this issue, we propose normative-based communities driven by agreement computing, where community interactions are regulated by norms and community members are free to discuss and agree on their community’s norms (or social contracts), while respecting their free will. The roadmap for self-evolution is then dictated by the lifecycle of norms and driven by the members’ agreement on those norms. The research required for supporting this lifecycle is varied and multidisciplinary – from social sciences and legal studies, to learning mechanisms, agreement technologies, natural language processing, formal logics, norm enforcement and regimentation, human computer interaction, and other software engineering techniques.

It is true that the notion of self-organisation is not new. Self-organisation, in general, refers to how a system reorganises to adapt to changes to its goals or environment, where the reorganisation is driven from within the system as opposed to some external control. In many cases, self-organisation is associated with emergence, where a structure appears at a higher level without it being represented at a lower level. Self-organisation has been studied in a number of areas, such as biology, chemistry, geology, sociology, as well as information technology. In AI, self-organisation has mostly been inspired by naturally existing self-organisation models, although some new mechanisms have been designed specifically for software applications [Marzo Serugendo et al (2004)]. For example, taking inspiration from the natural immune system for intrusion detection in network security [Foukia (2005)], or using swarm intelligence for the coordination and control of data network traffic [Wedde and Farooq (2006)].

In this article, however, we do not propose to follow traditional self-organisation AI techniques, but to open a new direction in AI research with a novel method for self-evolving software-enabled communities that is based on normative systems and agreement computing. The proposed research is original in the sense that we do not simply talk about self-organising software that usually imitates
existing natural systems, but we study how the human users’ evolving needs consciously direct evolution. We essentially propose to provide the human users with learning mechanisms that help them learn the best evolution path, along with agreement technologies that help them discuss, argue, and agree on their preferred path of evolution. The system then takes care of the formalisation and operationalisation of the agreed upon norms and their impact on the user interface. This is a fundamentally different approach from existing self-organising systems.

2.1.2.2 Roadmap for Self-Evolving Virtual Communities

In this article, we argue that it is time for us to take control of the evolution of our own virtual communities. To address this issue, we propose normative-based communities driven by agreement computing, where community interactions are regulated by norms and community members are free to discuss and agree on their community’s norms.

Norms are the rules that govern behaviour in groups and societies. They motivate and influence individual actions by dictating what values, beliefs, attitudes, and behaviours are deemed appropriate or not. Social norms have been extensively studied by anthropologists, sociologists, philosophers, and economists in the hope of understanding how they motivate individual actions, influence market behaviour, and so on. In multiagent systems, the study of norms

![Figure 1](image.png)

**Figure 1.** The lifecycle of norms lays out the roadmap for self-evolving communities
gained tremendous attention due to the critical issue of coordinating agent behaviour and actions. Although, unlike the social sciences, the distinction between social and legal norms has not been concrete in the field of computer science.

We adopt the idea of using norms to control, or mediate, community behaviour. This is motivated by the fact that software is usually engineered based on some notion of norms in mind; furthermore, the use of norms allows human users to discuss their interactions without the need for any technical knowledge about the software actually mediating their interactions. We note that by specifying a community through its norms, the evolution of the community becomes dictated by the evolution of its norms, and the evolution of the norms

Figure 2. Example of a First Evolution Cycle. Say a group of friends, the Anthropology Class of 2012, are active on their social network. They regularly discuss their social lives and upload photos and videos of their whereabouts. At the beginning, they are excited to share and discuss their social lives online. It makes their communication easier and they start feeling closer to each other. Overtime, however, they start feeling that photo tagging is being misused and it is cluttering their photos. Anna first raises the issue, after feeling she is being tagged in too many unrelated photos that she does not want to appear on her page. As the number of such photos starts to increase, she decides to raise the issue with her friends. During their discussion, it turns out that she is not the only one bothered with the tagging issue. Several others agree with her. After a lengthy discussion, where different solutions were proposed, the group agrees through voting that only the person who uploads the photo may tag people, as they should know best whom to tag. The system automatically accommodates this new norm and starts to enforce it.
will be driven by members agreeing on those norms. The evolution of software-enabled communities may be depicted by the lifecycle of norms, as illustrated by Figure 1.

We divide the lifecycle of norms in a software-enabled community into four different stages. In the first stage, community members discuss and agree on their community’s norms. If community members do agree on a new set of norms, then the norms’ lifecycle enters its second stage, where the agreed upon norms are automatically translated into some formal specification. Given the formal specification of norms, the lifecycle then enters the third stage, where the software-enabled community interactions are modified to operationalise the new norms. Finally, as the software runs to mediate community interactions, norms are enforced by ensuring that community interactions follow the agreed upon norms. See an illustrative exam-
ple in Figure 2 and Figure 3. The details of these four stages are presented next.

2.1.2.2.1 Agreement on Norms

In this stage, community members propose norm modifications (including the abandoning and adoption of norms), discuss the different proposals amongst themselves, and collectively agree on their community’s new (or modified) norms. Learning mechanisms can help learn from a community’s past experience, or a similar community’s past experience, and suggest norm modifications accordingly. However, the main building block for the collective agreement on community norms will be based on agreement technologies.

Concerning learning mechanisms, learning from data has been one of the main subfields of artificial intelligence, yet the literature lacks the in-depth study of learning the best norms for multiagent interactions. In multiagent systems, some work has been carried out on norm synthesis [Shoham and Tennenholtz (1995)]. Most norm synthesis (online) approaches, such as that by Sen and Airiau (2007), are based on norm emergence techniques that require agents to collaborate to synthesise their own norms. Morales et al (2011) does not require the active participation of agents in norm synthesis as norms are synthesised from the observation of agents’ interactions.

Concerning agreement technologies, we note that this field is one of the most vibrant and active fields in multiagent systems. Agreement technologies aim at helping individuals collaboratively reach a decision, which is crucial when individuals are intelligent and autonomous, having their own goals and agendas to fulfil. The field is based on a number of models and mechanisms, such as argumentation and negotiation mechanisms, social choice theory and voting mechanisms, and trust and reputation models.

Argumentation theory uses logical reasoning to illustrate how conclusions may be drawn in cases of uncertainty and conflict. Argumentation frameworks in multiagent systems have mostly been built on Dung’s theory of abstract argumentation [Dung (1995)], where a directed graph is used to represent arguments (the nodes of
the graph) and the attack relations between them (the directed links of the graph). A calculus of opposition is then applied to help determine the winning argument. Dung’s framework has been extended in a number of directions, such as adding preferences [Amgoud and Cayrol (2002)], or allowing attacks on attacks [Baroni et al (2011)].

Social choice theory is concerned with the representation and aggregation of individual preferences. Voting mechanisms and ranking systems are considered to fall under the social choice category. Social choice theory studies a variety of aspects, such as the manipulation of a voting rule [Conitzer et al (2003)], and the efficiency and fairness of voting rules [Endriss et al (2006)].

A common issue that arises in building open distributed systems is the issue of trust and reputation. Trust has become an increasingly important concept in computer science. For example, there are mechanisms that determine which sources to trust when faced with multiple conflicting information sources [Yin et al (2007)], and mechanisms that identify which individuals to trust based on their past activity [Adler and de Alfaro (2007)]. Trust is an especially important issue for autonomous agents and multiagent systems. In multiagent interactions, agents will have to reason about how much they should trust other entities (in various contexts). For reviews on trust in multiagent systems, we refer the reader to Sabater and Sierra (2005).

Roadmap.

We present below what is our proposed work for this stage and how it may be achieved. Recall that the output of this stage is to have the members’ agreed upon norms.

What? In the context of self-evolving communities, research in the information and computer science field should focus on a number of issues. (1) Define a language for norms that allows them to be comprehended and specified by the layperson and yet understood and reasoned over by the system. (2) Investigate when should norms evolve, and how can this be triggered. (3) Allow and aid members of a community to easily suggest, discuss, and agree on norms in a social context. (4) Understand the consequences of norms and suggest adequate norms accordingly. (5) Understand the emergence and adoption of norms and how this may be used in shaping
the tasks above. Recall that the output of this stage is to have the community agree on its norms.

**How?** First, we suggest norms to be specified in a controlled natural language, such as an Attempto Controlled English, which allows members of a community to easily understand, suggest, and discuss norms in a social context. This helps eliminate ambiguity and unclarity, and helps translate norms into a formal logic that permits reasoning over them. For example, we suggest building on work similar to that of Wyner et al. (2010), where a controlled natural language constrains the domain of discourse in a sample discussion from on-line discussion forums for e-government policy-making.

Concerning the triggering of evolution, a number of existing mechanisms may be used, such as sentiment analysis that picks up the dis/satisfaction of the community [Kouloumpiset et al. (2011)], or machine learning and pattern recognition techniques [Weiss and Hirsh (1998)] that learn when things are wrong and the rules of interaction need to be revised (say when collaboration decreases).

Concerning the norm suggestion and discussion stage, and as we are focusing on social contexts, we suggest following an approach similar to that of Leite and Martins (2011), which incorporates social voting to argumentation in social networks. This work may then be extended to study the combination of trust and argumentation mechanisms [Bonatti et al. (2014)]. For instance, by modifying the reasoning process to consider reputation/trust based measures of the strengths of arguments. It would also be interesting to study voting algorithms mediated by trust, with votes potentially weighted by the trustworthiness or degree of involvement in the community.

Another interesting aspect to investigate is the automatic learning of the best norms for a community. Existing norm synthesis techniques are one approach for pursuing this [Morales et al. (2011)].

Additionally, we suggest learning the consequences of norms. For this, either simulations can be used to study the ramification of norms in a given community, or learning techniques can be used to learn from the history of past experiences (of either the same community or similar communities, where similarity measures may be used to help decide which history of which community is worthwhile learning from). When norm consequences are learned, they can be used either to (1) suggest norms by choosing those with the
least undesired consequences, or even (2) provide arguments for the discussion between human members, where the system may join the discussion by supporting or attacking arguments.

It is important to note that the currently existing challenges of the above technologies decrease when focusing on our proposed ‘evolving communities’ scenario. This is because (1) we rely on a simple language for norms, and the techniques may be simplified for this scenario; and (2) there is the human intelligence, which helps guide and rectify the technologies at work.

Last, but not least, we note that an interdisciplinary research is necessary at this stage to help understand norm emergence and adoption. For example, neuroeconomics [Kvaran and Sanfey (2010)] can help us understand what norms are easier to accept. Social science [Bicchieri and Muldoon (2014)] can also help understand norm emergence, the impact of norms on behaviour, etc. Legal studies [Michaels and Pauwelyn (2011)] can aid with resolving conflicting norms, enforcement of norms, etc. The necessary interdisciplinary research forms the main challenges of this stage, as (to our knowledge) such an interdisciplinary research has not been carried out and employed yet in norm evolution mechanisms.

### 2.1.2.2 Automated Formalisation of Norms

Norms agreed upon by community members need to be formalised in a language that can be understood by the system mediating community interactions. We expect community members to be non-technical people who discuss and specify norms in natural language. An automatic translator is then needed to translate the norms from their natural language specification to some formal specification that is comprehensible by the system. We suggest translating to formal logic as an intermediary step before translating to software code because formal logic allows us to reason about the coherence of norms and their implications (e.g., accepting a new norm implies rejecting another).

The research that should support this stage should focus on natural language processing, which will help perform the automatic
translation, and formal logics, which will define the formal language used to specify norms.

Concerning natural language processing, machine translation is a sub-field that translates one natural language into another. This sub-field has been well studied and it has been even used in commercial applications, such as Google Translate or Yahoo’s Babel Fish. The translation from a natural language to some logic, on the other hand, has attracted much less attention in the natural language processing field. Nevertheless, important research has been carried in that direction. Wyner et al (2010) adopt and apply a controlled natural language to constrain the domain of discourse in a sample discussion from on-line discussion forums for e-government policy-making. The controlled natural language helps eliminate ambiguity and unclarity, and allows a logical representation of statements. Each of the policy statements is then automatically translated into first-order logic. Wyner and Peters (2011) present a linguistically oriented, rule-based approach, for extracting conditional and deontic rules from regulations specified in natural language. Finally, Athan et al (2013) present approaches for the logical representation of regulations.

Concerning the formal specification of norms, existing approaches may be divided into two main categories: declarative approaches and procedural ones. We note that in this proposal, we refer to the declarative approaches as norms and the procedural ones as interaction models. Declarative formalisms are concerned with the expressiveness of norms, the formal semantics, and how to resolve conflicts arising from an inconsistent set of norms. Declarative approaches are usually based on deontic logic, which is the logic of duties. They deal with concepts like permissions, prohibitions, and obligations, which help specify who can do what, under what conditions, and so on.

Deontic-based policy languages have been used widely in hardware systems and networks for security reasons, trust negotiation, access control, authorisation, etc. Sloman (1994) defines policies to be “one aspect of information which influences the behaviour of objects within the system”. Damianou et al (2002) categorise policies into two types. The first type covers obligation policies for managing actions, which are usually event triggered condition-action rules.
The basic concept is that specific events trigger specific actions, and the actions may only be executed if a predefined set of conditions is satisfied. The second type covers authorisation policies, which are usually used for access control.

In multiagent systems, several deontic-based formal logics have been proposed for defining a normative specification of agents’ interaction, such as that of Pacheco and Carmo (2003) and Dignum et al (2002a). Pacheco and Carmo (2003) define the roles within the community, the relationships among them, which actions each role can do, and how the obligations are distributed among the roles. Each role has associated deontic notions that describe the role obligations, permissions and prohibitions. Dignum et al (2002a) extends the BDI model of agents to include goals, obligations, and norms; the proposal is based on providing a formal definition of norms by means of some variation of deontic logic that includes conditional and temporal aspects.

**Roadmap.**

We present below what is our proposed work for this stage and how it may be achieved. Recall that the output of this stage is to have the agreed upon norms translated into some formal logic that is ready to be operationalised by the system.

**What?** Research for this stage should focus on three main issues. (1) Define a formal logic for the specification of norms that can be used for reasoning over the norms. (2) Build an automatic translator that can translate norms specified in a (controlled) natural language into the designed formal logic. (3) Build a reasoner that reasons over the norms and their incoherences.

**How?** The formal logic, which could be a basic first-order logic or deontic logic, should be designed/chosen taking into consideration the research work of stages 1 and 3 of the norm evolution lifecycle. This is because (1) norms in a (controlled) natural language need to be translated into this formal logic, and as such, there needs to be a clear and straightforward mapping between the two; and (2) the system operationalising norms needs to enforce the norms specified in this formal logic (whether through regimentation, or other means such as sanctions), and as such, they need to be properly interpreted by the system.
Concerning the translator, we suggest to get inspiration from existing work on translating policy statements into first order logic [Wyner et al (2010)], extracting deontic rules from regulations specified in natural language [Wyner and Peters (2011)], or providing a logical representation of regulations [Athan et al (2013)]. In fact, the translator should in principle be straightforward, as it will not be translating from a natural language, but from a controlled natural language, where we require a clear straightforward mapping between the controlled natural language and the chosen formal logic for norms.

Concerning the reasoner, the literature is rich with systems implementing deontic logic that deal with reasoning about the rules and addressing incoherences that may arise, such as the deontic-based policy languages [Damianou et al (2002)], and deontic-based normative multiagent systems [Dignum et al (2002a)]. We again suggest to get inspiration from existing research in this area, taking into account the particularities of the norm evolution lifecycle scenario and the chosen formal language.

It is important to note that the currently existing challenges of the above technologies decrease when focusing on our proposed ‘evolving communities’ scenario. Again, this is because we rely on a simple language for norms, and the techniques may be simplified for this specific language.

2.1.2.2.3 Automated Operationalisation of Norms

Given a formal specification of norms, the ultimate goal is to enforce those norms. One approach to achieve this is through what is known as norm regimentation. In other words, the system mediating community interactions needs to be modified in order to operationalise those norms.

The literature provides a variety of solutions that deal with specifying and regulating interactions in multiagent systems based on the concept of following social norms [Shoham and Tennenholtz (1995)], such as having contracts and commitments [Dignum et al (2002b)], electronic institutions [d’Inverno et al (2012)], distributed dialogues [Robertson (2004)], and so on. These approaches usually
are based on the idea of agents playing different roles and interacting by means of speech acts. Each role is defined by specifying what actions agents can perform, when can they perform such actions, under what conditions these actions may be carried out, etc.

In the context of self-evolution, it is important to build a system whose interaction models can operationalise norms requested by its users. For example, it is important not to allow users to submit their work after the deadline, or not to allow the seller to ship an item if the buyer did not make a payment. We note, however, that not all norms are capable of being operationalised. For example, in an e-commerce context, the system can operationalise the norm that states that a buyer cannot rate the seller more than once, as the system can actually prevent the buyer from doing so. However, the system cannot operationalise a norm that states that only people with sufficient credit can bid, as credit is private information that cannot be accessed and assessed by the system. It is also argued that not all norms should be operationalised [Castelfranchi (2003)]. For example, it may be argued that user’s autonomy is maintained when s/he is allowed to violate a norm, say for instance, one who writes an opinion about an item that she has not purchased.

To deal with norms that cannot or should not be operationalised, alternative norm enforcement methods will need to be applied, such as applying sanctions (punishments and rewards). In this case, the system will need to ensure the enforcement of un-operationalised norms by checking the members’ abidance to those norms and enforcing the appropriate sanctions accordingly. To achieve this, it is important to design a lightweight computational norm enforcement model that allows for the detection of norm violations and the application of remedial actions. This work may build upon existing

---

1 The literature usually refers to the operationalisation of norms as ‘norm regimentation’, and alternative norm enforcement methods (such as the application of sanctions) as ‘norm enforcement’. In this proposal, however, we overload the word ‘enforce’ as we use it to describe both approaches.

2 It may be argued that punishment and reward is not always the right approach for motivating the abidance to norms. Furthermore, what may be considered a punishment for one may be viewed as a reward for another. In this proposal, although we talk about punishments and rewards, we concur that sanctions may be labelled more generally as the post-conditions of abiding with or breaking a norm.
work, as the literature is rich with norm engines that may be used for resolving conflicting norms and applying appropriate sanctions [Modgil et al (2009)].

**Roadmap.**

We present below *what* is our proposed work for this stage and *how* it may be achieved. Recall that the output of this stage is to have a system that is capable of enforcing the norms agreed upon by the community members.

**What?** Research for this stage should focus on two main issues. (1) Build a norm regimentation mechanism, which includes a) building a translator to represent the norms as software constructs (e.g. protocols, constraints or rules), and b) adapting the community mediation software to allow for the internalisation/embedding of these software constructs. (2) Build a norm enforcement engine that can check members’ abidance to un-operationised norms and enforce appropriate sanctions accordingly.

**How?** Building the norm regimentation mechanism will need to take care of translating the norms specified in formal logic into the selected software construct type. Problems that may arise here could be in the lack of alignment of the expressivity of both languages. Aligning the expressivity will be one of the challenges of this work, as we discuss shortly.

The requirements of adapting the software will depend on the type of software constructs. If they are imperative, such as interaction protocols [d’Inverno et al (2012), Robertson (2004)], then formal verification will be needed to ensure the correctness of the resulting protocols [Osman et al (2006)]. If the constructs are declarative, such as constraints or rules, then conflict resolution mechanisms [Chomicki et al (2003)] will be needed to deal with potential conflicts that may arise when adding or modifying these constructs.

Concerning the norm enforcement engine, we suggest to get inspiration from existing work on norm enforcement [Modgil et al (2009)], such as applying sanctions [Fornara and Colombetti (2009)] or providing incentives for norm compliance [Lopes Cardoso et al (2013)].

2.1.2.2.4 Automated User Interface
Finally, running the software that mediates community interactions will allow members of the community to interact, while ensuring the enforcement of the norms. The research that will support this stage will need to focus on human computer interaction techniques that are required to ensure an intuitive, user-friendly interaction for non-technical community members. The most relevant work in this area is that on visualising discussions (e.g. debategraph.org and Carneades [Gordon (2013)]).

Research will also need to focus on software engineering techniques to ensure efficient community interactions. However, the context of self-evolution imposes an additional challenge: how can the system be modified at runtime when different members may still be interacting together? Indeed, it is very unrealistic to assume that all members will need to complete their current interactions successfully and pause any future interactions for the system to get modified (as future interactions will need to follow the new norms). As such, research will need to address the issue of seamlessly adapting the system at runtime, and it may take inspiration from existing software compatibility models that allow different versions of an application to continue to communicate and collaborate.

Roadmap.
We present below what is our proposed work for this stage and how it may be achieved. Recall that the output of this stage is to have an efficient, user-friendly, and intuitive user interface.

What? Research for this stage should focus on two main issues. (1) Build an automated user interface that would adjust itself to the evolving software. Additionally, the user interface should allow community member to interact, following the norms, without any knowledge of their interactions and their complexity. When members need to discuss the norms of their community, the interface should visualise the norm agreement process in a user friendly and comprehensible manner, which would include visualising the attacking and supporting arguments, norms, the past violation of norms, the consequences of norms, and so on. (2) Engineer the system in such a way that permits evolution at runtime in a seamless manner that does not interrupt community interactions.

How? Having an automated user interface is a major challenge. Initial work in this direction has been presented by de Jonge et al
Though this is very initial work on an otherwise uncharted territory. In fact, the work by de Jonge et al (2015) illustrates that although an automated interface is possible, generic solutions are not suitable (yet) for intuitive and user-friendly solutions. Another challenge is the visualisation of the norm agreement process. Existing work on visualising argumentation and deliberation (e.g. debate-graph.org and Carneades [Gordon (2013)]) may be built upon for the visualisation of the agreement process in general. However, the visualisation of norms, their violations, their consequences, is again uncharted territory. As such, we believe building the user interface will constitute one of the major challenges of this work.

Concerning the evolution of the system at runtime, problems may arise when different community members are using different versions of the system, with each implementing a different set of norms. Existing software compatibility models allow different versions of an application to continue to communicate and collaborate. We hope we can make use of such models, adapting them as needed to the idea of self-evolving community interactions.

2.1.2.3 Challenges and Open Questions

We have noted in the previous sections that the currently existing challenges of the aforementioned technologies decrease when focusing on our proposed ‘evolving communities’ scenario. This is because we rely on a simple language for norms, and the techniques may be simplified for this specific language. In fact, this has been illustrated by the preliminary work of de Jonge and Sierra (2015), which proposes a very basic controlled natural language (SIMPLE) that is used for the specification and enforcement of norms. The simplicity and intuitiveness of the SIMPLE programming language seems to support the mapping of SIMPLE into a formal logic for reasoning, as well as support the comprehension and use of SIMPLE by the layperson.

The new main challenge that arises with this research, however, is specifying the languages of each stage and building their relevant mechanisms (e.g. the reasoner for the formal logic, the software’s norm enforcement mechanism) with respect to the work carried out
in every other stage. In other words, the complete norm evolution lifecycle needs to be considered when working on the different lines of research presented earlier. For instance, there will be a need to align between the user requirements, natural language concepts, and the expressivity of the chosen formal logic. In this research, there is also a need to make the norms operational. As a result, the semantics will need to be determined by the chosen software system. With respect to the software, the need will arise with respect to revisiting (by extending or modifying) data structures as well as control structures to accommodate for new requirements imposed by the chosen formal logic. The work on the software will also impact the choice of the controlled natural language. For instance, when data structures cannot be specified due to requirements imposed by the controlled natural language, then the controlled natural language should be modified to eliminate such requirements.

We note that another main challenge that has been discussed earlier in detail is the challenge of building an automated user interface that gets modified with respect to the evolving norms. Although initial work has been carried out in this area [de Jonge et al (2015)], this remains preliminary work that requires new lines of research in the field of human-computer interaction.

2.1.2.4 Conclusion

This article argues that just like human communities, e-communities (or virtual communities) also need to self-evolve. Instead of creating numerous rigid systems, which is very common in online social systems, what we should aim at instead is providing tools for creating self-evolving systems that adapt to the community’s needs. We believe different communities should be governed by different rules. These rules should be an ever evolving set resulting from the aspirations of its members, and not a rigid set defined by a corporation (or a system designer) that does not take the community members’ interests at heart. Furthermore, for the community’s rules to be effective, they need to be tailored to the specific community, such as considering the character traits of the community’s members.
We believe that the notion of self-evolving communities will revolutionise the way how software is build, as well as how we interact with software. We say that in the context of social interactions, software does not need to be rigid, nor customised for the individual, but adaptable to the group. Software engineers will need to design intelligent software that is capable of evolving according to the needs of the community as a whole, and users (or community members) should become more active in shaping their community by discussing and agreeing on their community rules.

In summary, the proposal of this article is based on the traditional notion of mediating social interactions via norms. The roadmap for self-evolution is then dictated by the lifecycle of norms, which we divide into four different stages (see Figure 1): (1) the agreement on norms, (2) the automated formalisation of norms, (3) the automated operationalisation of norms, and (4) the automated user interface. Varied, and even interdisciplinary, research will be required for supporting these four stages, from social sciences and legal studies, to learning mechanisms, agreement technologies, natural language processing, formal logics, norm enforcement and regimentation, human computer interaction, and other software engineering techniques.

Acknowledgments This work is supported by the CollectiveMind project (funded by the Spanish Ministry of Economy and Competitiveness, grant # TEC2013-49430-EXP), and the EVOLVE project (funded by the Spanish National Research Council—CSIC).

References


[Michaels and Pauwelyn (2011)] Michaels R, Pauwelyn J (2011) Conflict of norms or conflict of laws?: Different techniques in the fragmentation of international law. In: Multi-Sourced Equivalent Norms in International Law


[Sabater and Sierra (2005)] Sabater J, Sierra C (2005) Review on computational trust and reputation models. AI Review


