

CHARMS: A Charter Management System. Automating the Integration of Electronic Institutions and Humans

Ismel Brito, Nardine Osman, Jordi Sabater-Mir, and Carles Sierra

Artificial Intelligence Research Institute (IIIA)
Spanish National Research Council (CSIC)
Bellaterra, Barcelona, Spain

Abstract. The execution of process models is usually presented through a graphical user interface, especially when users' input is required. Existing mechanisms, such as Electronic Institutions (EIs), provide means to easily specify and automatically execute process models. However, every time the specification is modified, the graphical user interface (GUI) needed during the execution stage should be manually modified accordingly. This paper proposes a system that helps maintain such GUIs in an efficient and automated manner. We present and test Charms, a system built on top of EIs that allows the automatic creation and update of GUIs based on the provided process model specification.

1 Introduction

The issue addressed by this paper is that of *specifying*, *executing*, and *maintaining* processes. The term 'process model' is a very generic term that has been used in various contexts, from business processes, to concurrent communicating systems. In all of these cases, the goal of a process model is usually to understand and define how a given process works. Several languages have emerged for the specification of processes. StateMate allows the specification of processes in an easy graphical manner [3]. Others have focused on the analysis of processes; for instance, process calculi permit the formal reasoning over process models [2, 4]. This was followed by executable languages that would allow the execution of processes in an automated manner (e.g. BPEL [1]). This paper tries to extend this line of work by providing means for *maintaining* processes in a straightforward manner that appeals to the average user.

We note that several existing tools already provide users with means for specifying and automatically executing their processes (e.g. BPMN). This paper, however, states that in addition to the specification and execution of charters, a management system is required for allowing the automatic creation and update of the user interface needed for execution. For instance, in current conference management systems, such as ConfMaster¹ and EasyChair,² the user interface

¹ <http://www.confmaster.net/>

² <http://www.easychair.org/>

is hard-coded to suit the predefined processes. But what if the entire peer review process is changed into something more open and similar to the Interdisciplines conferences,³ where authors are invited to submit papers (skipping the review process) and papers are simply discussed online by other researchers in the field over a predefined period of time? Such a change in the conference’s charter would require a radical redesign of the entire web-based interface. Sometimes even minor changes could require tweaking the interface accordingly.

In summary, the concrete goal of this paper is to provide a mechanism that eliminates the need for manually modifying the user interface every time a process is modified by introducing the automated generation and update of interfaces based on the specification of the process. This, in essence, provides a user friendly approach for maintaining process models.

The rest of this paper is divided as follows. Section 2 introduces the concept of charters, the key for specifying and managing processes. Section 3 describes Electronic Institutions, the system we choose for the specification and execution of charters. Section 4 briefly analyses the differences between EIs and traditional process modelling languages explaining why we think that EIs are a better option to specify and implement charters. Section 5 presents our main proposal, the Charms system which specifies and executes charters with an automated user interface (UI). Section 6 provides an overview of our motivating example, the Interdisciplines conference, and Section 7 explains how Interdisciplines can be specified and implemented using Charms, before we conclude with Section 8.

2 Charters

According to the Oxford English dictionary, a charter is “a written evidence, instrument, or contract executed between man and man”, it is a document “granting privileges to, or recognizing rights of, the people, or of certain classes or individuals”.

We say a charter is essentially a convention, agreed upon by a set of participants, that sets the rules of the interaction between these participants by declaring the rights and privileges each participant has. Accordingly, participants are legally binded to obey and observe these rules, their contract.

In its most basic form, a charter may be composed of a set of rights \mathcal{R} , where a right is defined as the tuple $\langle o, e, C \rangle$, and o represents the object that the entity e has a right to, and C represents the set of conditions under which this right holds. For example, $\langle \text{vote}, \text{citizen}, \{is_adult(\text{citizen})\} \rangle$ states that a citizen has a right to vote if it is an adult.

However, we note that the rights of one participant usually imply the obligation or forbiddance of another participant to perform some action(s). For example, the right to education implies an obligation on the governments to provide education to their citizens. As such, deontic systems have been used to specify the “rights and duties” of individuals and organisations. In a deontic logic, a

³ <http://www.interdisciplines.org/>

system is specified by a set of permissions, obligations, and related concepts. However, it is interesting to note that one may pick only one of the deontic concepts to be the basic one, and then all the remaining concepts may be defined in terms of the basic one by making use of negation. For example, if we assume ‘obligations’ to represent the basic deontic concept, then the ‘permission’ to smoke is equivalent to the lack of obligation to not smoke.

In organisations and institutions, charters are used to specify who may perform what action, when to carry out that action, and under what conditions (where the conditions are usually context dependent: different states of the interaction require different conditions). However, as we move from simple human interactions to more complex ones, enforcing the rules and guaranteeing an entity’s right becomes a challenge. The challenge becomes even larger in distributed open systems that are composed of autonomous interacting agents. One common approach in such cases is to make use a more precise plan of action — as opposed to relying on a set of rules — to guide the agent’s actions, yet enforcing the norms in an efficient manner and guaranteeing the rights of participants. This plan of action is usually defined through a flow graph. A flow graph essentially specifies the order in which the steps must be executed which steps could be executed in parallel, etc. The entire flow graph should be consistent with the norms and their requirements, and it could aid penalising misbehaving participants.

Formally, a charter may then be defined as tuple: $\mathcal{C} = \langle S, R, A, T \rangle$, where S represents the set of charter’s states, R is the set of roles that may engage in the given charter, A is the set of actions that may be performed, and $T : S \times A \times R \rightarrow S$ is a transition function that states that the system specified by this charter may move from one state in S to another is a role in R performs an action in A . The rights specified at the beginning of this section by the set \mathcal{R} map with the transitions of T (the transitions follow the rights \mathcal{R}). We note that such a tuple is analogous to Kripke structures and finite state machines.

Finally, in a charter, potential participants (humans, organisations, or autonomous agents) are grouped into roles according to their responsibilities. A role may be played by several participants (such as the case of having several bidders in an auction system), and a participant may play more than one role (for example, in a stock exchange scenario, a participant may play the role of the buyer of one stock and the seller of another). As such, charters do not focus on participants *per se*, but on the roles played by participants. This makes Electronic Institutions, which are introduced by the following section, very appealing to use for specifying charters as they already incorporate these concepts.

3 Electronic Institutions

The concept of *electronic institution* (EI) is inspired from human institutions. In open multi-agent systems, where autonomous entities interact to achieve individual goals, one cannot guarantee what will be the final outcome of this interaction. Therefore, and similarly to what happens in human societies, you need mechanisms to guarantee the good functioning of the system despite the

local behaviours, which is achieved by enforcing norms and penalising those that misbehave. The use of an electronic institution that regulates the behaviour of agents the same way human institutions regulate the behaviour of people is one of these mechanisms.

There are few concepts that have to be defined to fully understand the electronic institution machinery [5].

- *Agents and Roles*: agents are the players in an electronic institution, interacting by means of the exchange of illocutions, whereas roles are defined as standard patterns of behaviour. Any agent within an electronic institution is required to adopt at least one role although usually it will adopt several.
- *Dialogical Framework*: EIs establish the acceptable speech acts an agent can utter by defining an ontology and a common language for communication and knowledge representation. Both are bundled in what is called a dialogical framework. In an EI, all the actions are performed by means of speech act utterances.
- *Scene*: Interactions between agents are articulated through agent group meetings (which are called scenes) with a well-defined protocol. Protocols in a scene are considered to be the specification of the possible dialogues the participating agents may have.
- *Performative Structure*: Scenes can be connected, composing a workflow, in a so-called performative structure. The specification of a performative structure contains a description of how agents can legally move from scene to scene by defining both the pre-conditions to join and leave scenes. Transitions, a special type of scenes, allow synchronization among agents. They can be used also as choosing points where agents can decide which path to follow in the performative structure, and parallelization points that allow agents to be sent to more than one scene. Figure 2 shows the performative structure we will use in our example.

The link between an agent and the EI is the *governor*. The governor serves the purpose of safe-guarding the EI, i.e. it checks whether a particular message is allowed to be said (and therefore an action to be performed) at the current stage depending on the role of the agent, the state of the scene, etc.

The Electronic Institution Development Environment (EIDE)⁴ is a set of tools aimed at supporting the engineering of multiagent systems through electronic institutions. EIDE is composed of the following tools:

- ISLANDER: A graphical tool that supports the specification of the rules and protocols in an electronic institution.
- AMELI: A software platform to run electronic institutions specified using ISLANDER.
- aBUILDER: An agent development tool.
- SIMDEI: A simulation tool to animate and analyze electronic institutions.

⁴ <http://e-institutions.iiia.csic.es/>

Although at first sight EIs and process modelling languages might seem closely similar, there are some fundamental differences that make EIs much suitable for the specification and implementation of charters. In the next section we explain these differences.

4 Comparing EIs to Process Modelling Languages

There are several similarities and differences between business process modelling languages (e.g. BPEL [1], or the modelling framework BPMN⁵) and EIs. Both approaches have a notion of choreography and orchestration, but the notions are significantly different. While in BPMN orchestration refers to the workflow processes that are *internal* to an agent,⁶ in EI the orchestration consists of a workflow of *external* activities enacted by groups of agent, that is, to a set of dialogical activities where different agents interact. This is a rather fundamental difference as EIs remain neutral to the internal organisation of agents, that are considered autonomous and externally programmed, while constraining their public activity by the definition of these external activities. The choreography in BPMN, modelled as messages between processes in the workflow of the agents, does not have a clear structure in the form of a dialogue following a protocol; it consists of messages connecting internal processes of the agents. In other words, its structure is less precise than in electronic institutions where the choreography happens within the boundaries of a scene⁷ and as a clearly defined dialogue among a number of agents. Choreography and orchestration (in this collective sense) are more clearly differentiated in electronic institutions.

A major difference between the two orchestration approaches is that in the case of BPMN the connectors between activities indicate decision points made by a single agent (as the workflow in fact corresponds to a single agent) while in EIs the transitions correspond to synchronisation and parallelization points for agents. The flow of data is perhaps more clearly represented in BPMN than in EIs as in the latter formalism the data updates are performed as a consequence of the orchestration, i.e. as a consequence of the exchange of messages among agents within a scene, and written in an action language without any graphical help to the designer. The notion of group in BPMN resembles a bit the notion of scene in EI but the flow of agents between scenes does not have a clear equivalent in the group concept as they cannot be interconnected. The concept of group is more an annotation element than a fundamental operational one as is the case in EIs. Both approaches are neutral with respect to methodology and admit different ways of generating a specification (e.g. LOVeMTM or IDEF for BPMN or PROMETHEUS+ [6] for EI). In summary, the emphasis of EIs is on the flow of agents between activities and the choreography of their activities, while the emphasis of BPMN is on the coordination of the internal activities of the agents and on the sharing of data among these internal processes.

⁵ <http://www.bpmn.org/>

⁶ in BPMN terminology, an organisation.

⁷ similar to the notion of ambient in ambient calculus [2]

The fact that the most important aspect of a charter is to represent *joint* activities of the participants in a process (business or scientific), makes us think that EIs are more suitable for our purposes, given the emphasis on regulated scenes (precisely the notion of joint activity) and the existing tools to design and verify the flow of agents between them. Although BPMN processes can be used between different agents to coordinate their work, this task is usually not straightforward. The notion of group in BPMN is too weak and there is no direct equivalent to a flow between groups which is key in the specification of charters.

5 Charms: a Charter Management System

The charter management system, Charms, is an independent-domain platform built on top of the EI technology, and aims at specifying and executing charters in a fully automated manner. Although Charms is domain independent, the rest of this paper will use the conference management systems domain as a prototypical domain.

Existing examples of conference management systems are the well known ConfMaster⁸ and EasyChair.⁹ These systems are hardcoded and therefore, their web interfaces require a manual update every time a change occurs in the specification. They assume that you will manage a conference that follows the standard workflow. But what happens if you want to run a different kind of conference? These systems become inappropriate as the changes to adapt to different workflows are too costly.

Charms wants to overcome this problem. The objectives of the proposed system are twofold:

- to provide charter organisers with straight forward means for the specification and automatic execution of charters, and
- to provide the automatic creation and update of a web-based user interface. The main goal is to eliminate the need for manual modifications in the UI every time the charter changes.

Charms enacts charters as electronic institutions. As we have seen in Section 3, the ISLANDER editor for EIs already provides an easy to use user interface for the formalisation of process models through a simple drag-and-drop mechanism. This formalisation includes the specification of interaction protocols, illocution schemas and a concrete ontology. Once the charter's flow is specified as an EI, Charms executes it using AMELI, also provided by EIDE. However, what is truly novel here is that Charms generates the web-based user interface that allows the users to interact with the specified charter in a fully automated manner.

As illustrated by Figure 5, Charms architecture is based on the standard client-server distributed model. The server side of the Charms architecture relies on two components: an electronic institution (EI) and a java servlet (or I-Agent).

⁸ <http://www.confmaster.net/>

⁹ <http://www.easychair.org/>

Whilst the EI (the kernel of the system) aids us in specifying and executing charters, the servlet is in charge of notifying the user of relevant events occurring inside the EI. The servlet (or web server) is the main link between the user and the EI. The servlet also acts as an I-Agent, from the point of view of the EI. In the EI, the I-Agent represents the human user.

By contrast, the client side of the Charms architecture (Figure 5) includes a web-based client application (AJAX) and a web browser (or the UI). Concerning the client application, we have adopted an approach based on two web technologies: Java Server Pages (JSP) and Asynchronous JavaScript And XML (AJAX). JSP technology provides a simplified fast way to create dynamic web content by merging HTML with JAVA. In contrast, AJAX provides a high level of interactivity, avoiding the undesirable reloading of the web pages after each user action. Hence, the client application only needs a relatively recent web browser and therefore it is not necessary to install any special software. We note that in practice, there is one client side per each user connected to the Charms platform.

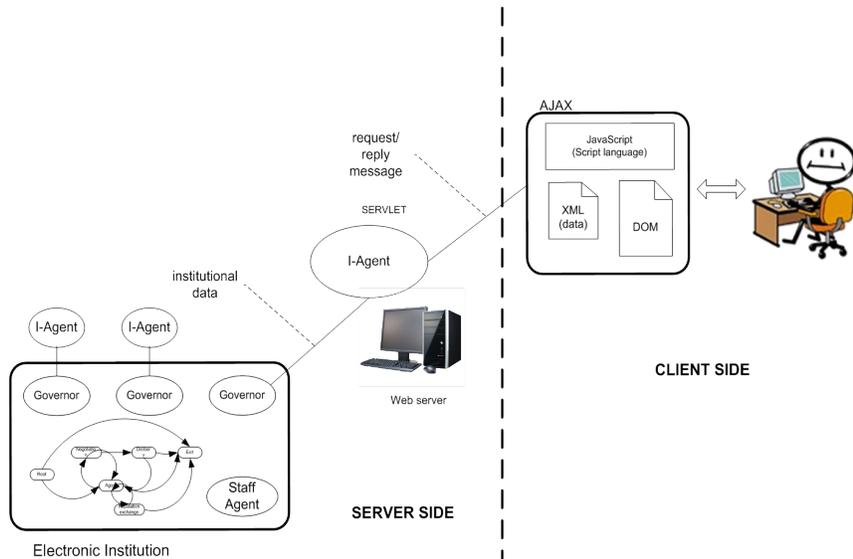


Fig. 1. Architecture of the Charter Management System, Charms

The central element in the link between the user and the EI is the servlet, which is activated once the user (client side), using a web browser, establishes the connection with the server side. Note that while the I-Agent is seen as a servlet from the point of view of the web server, it is seen as a normal agent from the point of view of the EI.

The client receives XML events from the servlet describing the changes produced in the EI that update the state of the charter's flow. These changes are shown to the user, and can be of four types:

- Events that inform a user about her position in the charter’s flow.
- Events that inform a user where she can move forward in the charter’s flow.
- Illocutions a user can say.
- Illocutions addressing the user in question.

Simultaneously, the I-Agent, acting as a servlet, receives events in XML format from the client application (which runs locally in the user’s computer as a web application). These XML events represent actions that the user wants to perform. These could be of two types:

- Events that inform the corresponding I-Agent about the illocution the user wants to say in the EI.
- Events that inform the corresponding I-Agent where the user wants to go from her current position in the general charter’s flow.

The I-Agent then acts accordingly, inside the EI, as soon as it receives an event of one of the above two types.

So far we have focused on the back-end issues of Charms. We have described how the different elements of the Charms architecture exchange events among them. But how are these events processed and displayed in the web-based interface that final users see? It should be first said that the current implementation of Charms does not allow to customize what users see. In other words, we use a fixed web template to build the web-based interface. This template is mainly divided into four tables, each one including one of the following information: where the user is in the charter’s flow (state), received messages, permitted actions and the messages the user can send (see Figure 3). Depending on the type of the event received, the client side displays it in the corresponding table. The AJAX technology allows the modification of only one section of the screen that is concerned with the new information.

The visualization process involves the I-Agents and the client application. As aforementioned, the link between the client side and the EI is the I-Agent (or servlet). The I-Agent stores the list of events coming from the EI that need to be notified to the user. The client application asks the I-Agent for new events and displays them properly in the web browser. To do that, the client application implements a cycle timer expressed in milliseconds. With each cycle, the client application requests new events from the I-Agent. When the I-Agent receives a request for events from the client, it pulls events from the list and builds a response for the client, including the new events in XML format.

6 Interdisciplines, a motivating example

*Interdisciplines*¹⁰ is a web-based platform aiming at organizing online interdisciplinary meetings such as workshops, seminars and conferences. It addresses interdisciplinary researches in philosophy, cognitive science and social science.

¹⁰ <http://www.interdisciplines.org/>

Since its creation in 2001, the Interdisciplines website has organised sixteen meetings with more than 150 papers. The site has more than 4000 registered users who participate playing the roles: *administrator*, *author*, *forum moderator*, *panelist* and *reader*.

Interdisciplines helps meeting organisers to create and run interdisciplinary meetings, which differ from standard academic meetings. In standard academic meetings researchers submit papers to be considered for presentation. Usually, every submitted paper is reviewed by members of a program committee, who evaluate the manuscript and determine if it is accepted for the meeting. By contrast, the organisers of an Interdisciplines meeting invite a set of authors to submit papers (with a maximum of 3000 words) to the meeting, and the papers are then debated within a discussion forum with the assistance of a scientific committee. We note that authors may also invite other researchers to coauthor papers with them. That request, however, needs to be approved by either an administrator or a moderator of the meeting. Furthermore, moderators and panelists assess papers according to several criteria, such as focus, originality, arrangement, methodology and writing. These reviews are not public and can only be accessed by the reviewers who made them, the authors of the manuscripts and the meeting administrators.

Meeting organisers must determine when and for how long a paper is debated. Generally, that period of time goes from 15 to 30 days. For every discussion thread, the meeting organisers also define who will play the roles of moderator and panelist. The debate of a contribution starts out when a panelist or reader makes a comment on the paper. The authors may respond to comments made by the panelists. Messages from panelists and authors are automatically shown in the forum threads that the website holds. However, comments made by users playing the role of reader must be validated first by the moderators.

7 Implementing Interdisciplines in Charms

In this section, we use Charms to implement Interdisciplines. In fact, the implemented version is a slightly lighter version of Interdisciplines, which, for example, does not take into consideration the different sections of a paper. This lighter version is used to keep the EI model clear and simple for the reader. As mentioned in Section 3, the modelling task requires the definition of roles, performative structure and interaction protocols. The roles, as introduced by the previous section, are: that agents can play in Interdisciplines are: *admin* (or the conference organiser), *moderator*, *panelist*, *author* (the main author), *coauthor*, and *reader* (or the guest that may read the website).

In addition to the above, EI requires one more role: *staff*. Except for staff, the roles above are played by human users. The staff role belongs to the EI and takes care of different aspects related to the good functioning of the system. Hence, the institution designers are in charge of defining and implementing the actions the staff role will do. As we mentioned in Section 3, this can be done by using aBUILDER, the agent development tool provided by EIDE.

The performative structure of an EI defines how agents are permitted to move among the different scenes and transitions. Figure 2 shows the performative structure of the proposed EI for Interdisciplines. In this figure, the blocks represent the different scenes, and the arcs linking these scenes represent the permitted transitions between scenes. Note that transitions are labelled with the roles that are allowed to perform this transition. For example, only an agent playing the role “admin” is allowed to move from the “Admission” scene to the “CreateConf” scene, where conferences are created. Also note that agents enter and leave the EI through the “begin” and “end” scenes, respectively. In summary, Figure 2 states that agents playing non-institutional roles must authenticate in the *Admission* scene before doing anything inside the EI. Afterwards, administrators can ask the staff agent for creating a conference in the *CreateConf* scene. This request includes the list of parameters we have mentioned in Section 6 needed to create an Interdisciplines conference. In the *Submission* scene authors can submit papers to the conference, invite other researchers to coauthor papers and propose text about their papers. Coauthors, however, can only submit papers and propose text about their papers but not inviting other researchers to participate in their papers. Except for admin, all the roles are involved in the *RunConf* scene, where they participate in paper discussions. Next we provide further details of the interaction protocols for the Admission, CreateConf, Submission and RunConf scenes:

Admission: Every agent must be authenticated by the staff agent in this scene. The staff agent determines who can enter the system by validating user authentication data (LoginData), which consists of an email and a password.

CreateConf: This interaction protocol allows conference organisers to create new conferences. Two roles may participate in this scene: staff and admin. The admin agent (meeting organiser) requests the staff to create a new meeting. The request includes the list of papers, moderators and panelists as well as relevant dates for the conference.

Submission: Basically this interaction protocol allows researchers to submit papers to the conference. Roles staff, admin, author, and coauthor are involved in this process. Authors can invite other researchers to co-write papers. However, these requests have to be validated by any administrator agent. The new invited researchers play the coauthor role. Coauthors can submit a paper to the conference but cannot invite new researchers to participate in the writing of the paper. Authors and coauthors of a given paper can share pieces of texts in order to improve the paper.

RunConf: This is the place where papers are discussed. Moderators, panelists, authors, coauthors and readers are the agents permitted to participate in this scene. They make comments on papers by exchanging messages. Panelists and readers react to papers, authors and coauthors reply and moderators validate messages from readers. Note that while comments coming from authors, coauthors and panelists are automatically published, those coming from readers need to be reviewed by the moderators.

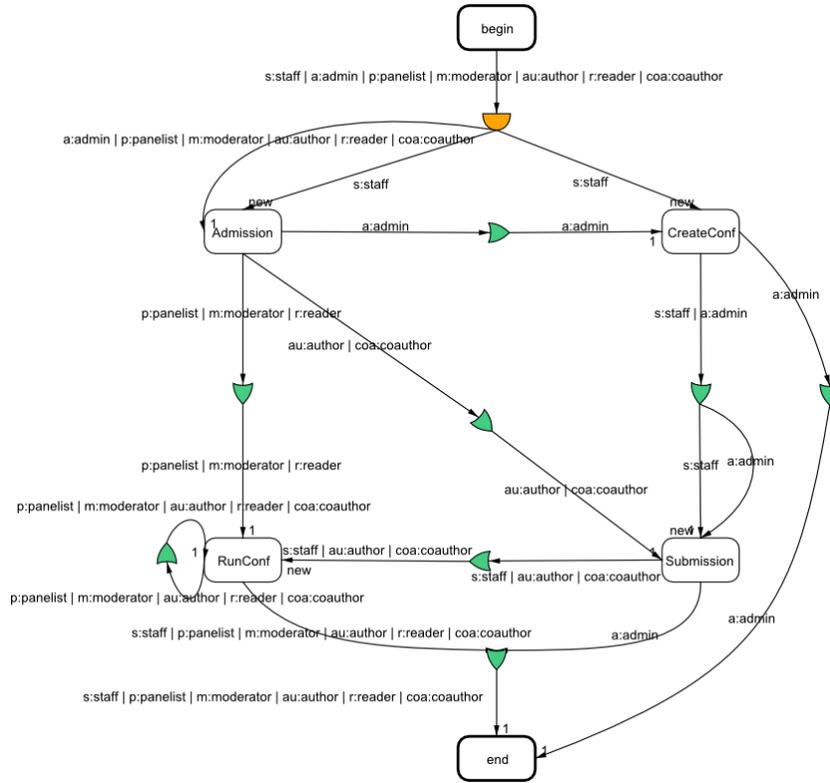


Fig. 2. Interdisciplines' performative structure

7.1 Web-based user interface for Interdisciplines

In the previous section, we illustrated how Interdisciplines may be specified through an EI. We now move on to discuss the automated UI of Charms.

The first step a human user has to do is to connect to the URL with the UI, in our case: <http://localhost:8080/interdisciplines-webapp/>. As we can see in Figure 3(a), the user is required to choose the name and the role she wants to play in the charter. In our running example, the user **Peter**, playing the role **admin**, is at the transition **start**. In general, when the user submits the name and role to start playing in a given charter, Charms forwards the user to a new web page (Figure 3(b)), which contains the following:

- Status:** This described the user's role and its location w.r.t. the step flow.
- Received messages:** The messages the user received since she entered the scene.
- Permitted actions:** The actions the user may perform according to its role and status in the charter's flow. These include:
 - Moves:** The moves the user is allowed to do, i.e. the list of transitions or scenes the user may go to.
 - Messages** The messages the user is allowed to say.

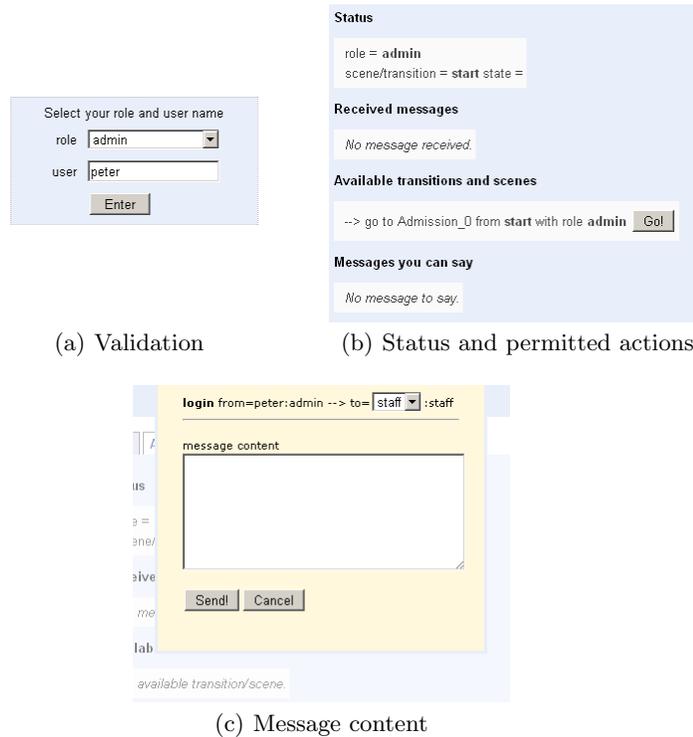


Fig. 3. Sample of the web-based user interface for Interdisciplines

The charter specifies that Peter can only move to the **Admission** scene after entering the system. Figure 3(b) illustrates that Peter neither has received any message from other users nor is he permitted to say any messages, and the only transition it may choose is to go to the **Admission** scene.

Afterwards, Peter goes into the Admission scene by pressing the **Go!** button of Figure 3(b). Once Peter is in the scene, he can only authenticate into the system by sending a message to the internal staff agent. By clicking on the **Message content** button, a pop-up appears as we see in Figure 3(c). It contains a description of the message, the possible recipients of it and a blank text area, where the user should fill out the content of the message, in this case an e-mail and a password as was defined in the specification of the EI.

7.2 Changing the Interdisciplines charter

In this section we show how easy it is to make a change in the above charter and how this change is automatically reflected in the web user interface. For the sake of simplicity, the change is quite simple with the aim to illustrate the potential of the approach.

As illustrated by Figure 4, we note that amongst the roles are the author role and its sub-role, the coauthor. This means that all the actions that are allowed to a coauthor, are allowed also to an author, but not vice versa. In what follows, we will focus on the actions that these roles has the right to perform.

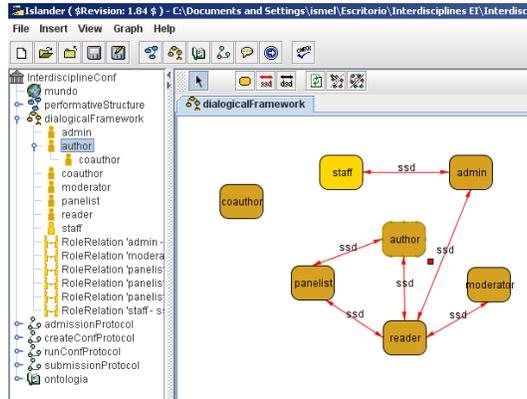


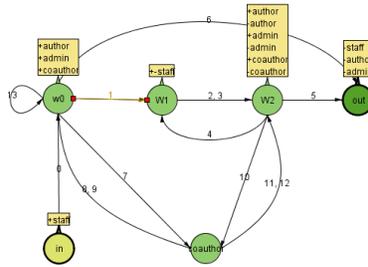
Fig. 4. Roles structure in Interdisciplines

In the original charter specification both, authors and coauthors, are allowed to submit a paper as shown by arrows #1 and #4 in the Submission scene protocol (see Figure 5(a)). According to the charter, authors could invite a coauthor, submit a full paper and propose content for a paper (i.e. text); whereas coauthors could only submit and propose text. This is automatically reflected in the web interface that the users playing the role of author and coauthor see when they are in the submission scene, as shown for the coauthor case in Figure 6(a).

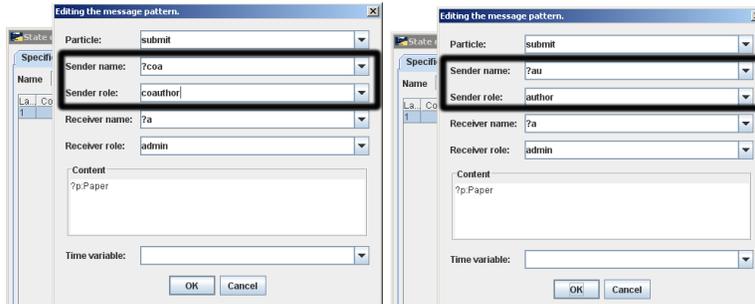
Now we want to change this basic functionality and modify the charter to limit the actions that a coauthor is allowed to do. In this new version, only authors are permitted to submit a paper whereas coauthors are limited to propose new texts. Table 1 summarizes the actions allowed to authors and coauthors in the original charter (column Before) and the functionality we want for the new version (column After).

In this case, updating the charter is quite easy. In the specification protocol of the original charter, the role associated with arrows #1 and #4 is “coauthor” (see Figure 5(b)). This means that only the agents playing the role of coauthor and author can submit a paper (remember that coauthor is a sub-role of author). If we want that only authors are allowed to submit a paper what we have to do is to change the role associated with arrows #1 and #4 so that only agents playing the role author can perform that action (see Figure 5(c)).

The key aspect of the approach is that once the charter specification has changed, no further modifications are necessary in the web interface. Now, when a coauthor arrives to the Submission scene she sees the screen in Figure 6(b) where the functionality of submitting a paper has automatically disappeared.



(a) Submission Protocol



(b) Before the change

(c) After the change

Fig. 5. Submission scene protocol specification



(a) Before the change

(b) After the change

Fig. 6. Submission scene web interface for a coauthor

Actions	Author		Coauthor	
	Before	After	Before	After
Invite coauthor	X	X	-	-
Submit	X	X	X	-
Propose text	X	X	X	X

Table 1. Permitted actions for authors and coauthors before and after the change

8 Conclusions and future work

This paper introduced using electronic institutions as a mechanism to specify and execute charters. We presented Charms, a running system that takes advantage of all the tools that are already available for the specification and execution of electronic institutions. The novelty of our work is that the system permits the maintenance of a web based user interface that is automatically generated from a charter specification. The changes in the specification are reflected in the web user interface in a fully automated manner, and without the need of any manual modification to the UI. We note that available technologies, such as Java Server Faces, may be used in aiding the design. However, Charms differs from these technologies in that it generates the mapping between the electronic institutions and the user interface in an automated manner. To our knowledge, there is not such a thing implemented yet in electronic institutions.

With this tool, managing charters becomes a simple task. Defining different policies, norms, or social behaviour for particular components in a complex structure is done in a modular and automatic way. The behaviour allowed to agents (or human users) is automatically and transparently derived from these charters. Thus, when managing pieces of knowledge, the user is offered the functionality that is appropriate to each type of knowledge in a dynamic way. That is, as knowledge evolves by the actions of others, the user's allowed functions are updated in front of his/her eyes. The feeling of being in a dynamic world is then perceived in a very vivid way.

Currently, the generated web user interface is very simple and, although it works as a proof of concept, it is not friendly enough for a final user. The visual aspect and the functionality of the interface has to improve. The current interface refers to low level electronic institution structures, using terms like scenes, transitions, roles, etc. These low-level details should be hidden. To solve that, a mapping between low level electronic institution elements and domain dependent elements is necessary. This mapping should be part of the specification.

References

1. J. Boile, M. Cardella, S. Blanyalet, M. Juric, S. Carey, Chandran P., Y. Coene, K. Geminiuc, and M. Zirn. *BPEL Cookbook*. Packt Publishing, 2006.
2. L. Cardelli and A.D. Gordon. Mobile ambients. In *Proceedings of the First international Conference on Foundations of Software Science and Computation Structure*, volume 1378 of *LNCS*, pages 140–155. Springer-Verlag, 1998.
3. David Harel and Michal Politi. *Modeling Reactive Systems with Statecharts: The Statechart Approach*. McGraw-Hill, Inc., New York, NY, USA, 1st edition, 1998.
4. R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.
5. Carles Sierra, Juan A. Rodriguez-Aguilar, Pablo Noriega, Marc Esteva, and Josep Lluís Arcos. Engineering multi-agent systems as electronic institutions. *UPGRADE The European Journal for the Informatics Professional*, V(4):33–39, 2004.
6. Carles Sierra, John Thangarajah, Lin Padgham, and Michael Winikoff. Designing institutional multi-agent systems. *LNCS*, 4405:84–103, 2007.