Human Interactions in Electronic Institutions

Dave de Jonge, Bruno Rosell, Carles Sierra

IIIA-CSIC, Bellaterra, Catalonia, Spain
{davedejonge,rosell,sierra}@iiia.csic.es

Abstract. Every social network has its own fixed, but different, set of rules that apply to all users. This reflects the fact that in real life every community has different norms depending on the relationships between its members. Unfortunately this has required people to create many different social networks that exist next to each other even though they have largely overlapping sets of members. In this paper we argue that Electronic Institutions (EI) solve this problem by allowing to create a generic social network in which users can set up their own sub-communities with their own particular norms and protocols. Electronic Institutions make it easy for users to specify these protocols and norms in a visual way, and adapt them when necessary. Furthermore we present a new framework on top of the existing EI architecture that allows humans to interact in any EI. It can generate a graphic user interface from the institution-specification without the requirement of any extra programming or design. However, it still allows designers to design a more sophisticated, domain specific GUI.

1 Introduction

Electronic Institutions (EI) have introduced a new paradigm for the development of software applications, in which the tasks are executed by independent agents, that are not necessarily designed specifically for that application and that cannot be blindly trusted. Just like any human institution, an EI is a place where participants come together and interact according to some pre-defined protocol. It makes sure that the norms of the institution are enforced upon its participants and thus prevents them from misbehaving. An EI therefore provides the infrastructure in which agents can interact in an autonomous way within the norms of the institution. With the rise of the Internet more and more activities are taking place in an open environment where one does not have direct control over who is participating in it. When interacting with an online tool it is often difficult to determine who programmed it and whether it can be trusted. Electronic Institutions provide a framework where agents can safely interact with other agents, regardless of their origins.

Electronic Institutions have been under development for more than 15 years [1],[4],[5],[8] which has resulted in a large framework consisting of tools for implementing, testing, running and visualizing them. This framework was originally developed with the aim of creating institutions consisting purely of software agents. For many purposes, however, it is desirable to have both humans and agents participating. Therefore, in this paper we present a new framework on top of the existing EI framework that provides a Graphic User Interface (GUI) that allows humans to enter into an EI and interact with other users or software agents.

We think that this framework could be especially useful for the development of a new type of social network, where users can set up sub-communities, each with its own rules and protocols. We argue that the fact that many social networks are nowadays existing next to each other is inefficient and is due to the fact that users are not able to adapt norms and protocols to their own needs. Electronic Institutions would provide a solution to this problem. Another advantage of human users interacting in an EI is that it allows developers to test an institution during its development, without having to program its participating agents. While the EI is under development human users can take the place of the software agents that would later participate in it, for testing purposes. This will allow for faster development.

Our framework generates a default user interface automatically from the EI specification, without the need for extra programming. But, on the other hand, if one does require a more case-specific user interface, it still provides an API that enables any web designer to easily design a custom GUI without the need for much knowledge of Electronic Institutions, or Java programming. Our approach is completely web-based, meaning that the GUI is in fact a website, implemented using standard web-technologies such as HTML5, Javascript and Ajax. In short, we have developed our framework with the following goals:

- To allow people to interact in an EI through a web browser.
- To have a generic GUI that is generated automatically.
- To allow any web designer to easily design a new GUI, if wanted.
- To allow testing of an EI under development, before having implemented its agents.

Visual user interfaces for Electronic Institutions have been created before, for example in [9]. In their work the user controls an avatar that walks around in a 3-dimensional virtual world that represents the EI. Although a 3D-virtual world may be more impressive visually, we think that a simple website that runs in a web browser has several advantages:

- Websites are cross-platform.
- No need for heavy hardware: it works on mobile phones and tablets.
- More practical as one does not have to walk around an environment, but simply has all options directly available in the form of buttons or menu-items.
- We think a 2-dimensional environment is more common in the context of social media[1] and therefore users will feel more familiar with it.

[1] Although the 3D social game Second Life has been very popular for a while, it has never been nearly as popular as 2D media such as Facebook and Twitter.

This paper is organized as follows: first, in Section 2 we give an overview of Electronic Institutions, in which we explain the basic concepts and terms necessary for the rest of this paper. Next, in Section 3 we explain how we have realized the automatic generation of a Graphic User Interface for Electronic Institutions and give the technical details. In Section 4 we argue that Social Networks are in fact examples of institutions where human behavior is regulated by norms and protocols and explain why we think that these norms should become more flexible and how Electronic Institutions could help realizing that. In Section 5, we introduce two use cases that we are currently using to further develop our framework. Finally, in Section 6 we summarize our conclusions.

2 Electronic Institutions

An institution is an organizational structure for coordinating the activities of interacting agents; it typically embodies some rules that govern these interactions. A commonly cited example is that of a fish market, with buyers and sellers engaging in interactions aimed at buying and selling fish. They have strict conventions by which fish is traded under strict negotiation protocols. More specifically, the fish market is an auction house that enforces certain conditions on the eligibility of traders, the availability and delivery of goods and the behavior of participants. While the actual trading makes up the critical part of the fish market, there are other interactions that are also governed by rules. For example, before any trading can be undertaken, sellers must deliver fish to the market, and buyers must register. Furthermore, once a deal has been agreed, the sellers must pay for and collect the fish, and the buyers must collect payment. Beyond this example, many other institutions have similar sets of distinct activities that can be identified, like hotels and universities.

2.1 Scenes

Just as there are meetings in human institutions in which different people interact, Electronic Institutions have similar structures, known as *scenes*, to facilitate interactions between agents. Scenes are essentially group meetings, with well-defined communication protocols that specify the possible dialogs between agents within these scenes. For example, an electronic fish market may include an auction scene in which buyers compete to purchase fish, with a protocol that involves making bids. There may be many simultaneous instances of such auctions within a fish market, each referred to as a *Scene instance.*

Scenes within an institution are connected in a network that determines how agents can legally move from one scene to another. In the fish market example, a buyer can only enter the auction scene after passing the registration scene.

2.2 Actions

Activities are dialogical as they are achieved via agent interactions composed of non-divisible utterances, that occur at discrete instants of time. These utterances can be modeled as messages that conform to a certain pattern, and physical actions are represented by appropriate messages of this form.

In an auction, for example, a buyer commits to buy a box of fish at a certain price by making a bid, while the actual physical action of transferring money from the buyer to the auction house is triggered when the auctioneer declares that the box is sold. In the rest of the paper we will therefore use the words 'action' and 'message' interchangeably.

For each message that can be said, a number of parameters may be specified by the protocol. When making a bid in an auction for example, the maker of the bid should include the amount of money he bids in the message. Electronic Institution support several basic parameter types, such as 'Integer', 'String' and 'Boolean'. Apart from these basic types the designer of an institution can define custom types, which are composed of one or more parameters of a basic type.

2.3 Scene Protocols

The agents in a scene in an EI have to interact according to some protocol. The protocol defines which agent can say what and when within the scene. At each moment during the execution of a protocol, the protocol is in a certain *state*, depending on the messages that have been said so far. The current state of the protocol determines what kinds of messages each agent can send.

In an auction for example, the protocol may start in a state in which the auctioneer introduces the next item under auction. Participants are not allowed to make any bid yet in this state. Once the auctioneer announces the start of the auction, the state changes to a bidding state, in which the participants are allowed to make their bids.

A protocol is therefore represented as a directed graph in which the nodes are the states of the protocol. Each edge of the graph is labeled with one or more message patterns. A message can only be sent if it satisfies one of the patterns labeling one of the outgoing arcs from the current state.

2.4 Roles

Scene protocols are not specified in terms of agents, but rather in terms of *roles*. Every agent plays a specific role that determines which actions it can take at which moment. Roles can be understood as standardized patterns of behavior that agents, when instantiating a role, must respect.

2.5 Constraints

As explained above, the state of the protocol restricts the actions that can be taken by the agents. However, the actions can be restricted even further by including constraints in the protocol. Constraints are given as sentences in a first-order logic attached to a message pattern. A message can only be sent if its corresponding constraints are satisfied.

2.6 Governors

Each agent participating in the EI has a special agent assigned to it, called its *Governor*. The Governor of an agent α has control over each message that is being sent by α. Whenever α tries to sent a message, this message first passes α's Governor, which checks whether the protocol is in the correct state and whether the constraints are satisfied. If so, the Governor forwards the message to its recipient. If not (for example, because the agent made a bid that is higher than what he can afford), the Governor blocks the message.

2.7 Ontology

As explained in Section 2.2 messages can have parameters, which can be of a basic type or of a user-defined type. Each EI has an Ontology associated to it which stores the definitions of these user-defined types. Also it stores for each message how many parameters it has and which types those parameters have.

2.8 Developing an Institution

The design and execution of an EI is done through a framework called *EIDE*, which is implemented in Java. The two main components of EIDE are called *Islander* and *AMELI*.

Islander is a visual tool to design institutions. It allows you to visually define the scenes, roles, protocols, message patterns, constraints, ontology and other components of the institution. It then converts the visual representation into xml format (the *EI-specification*) that can be read by AMELI. AMELI is the

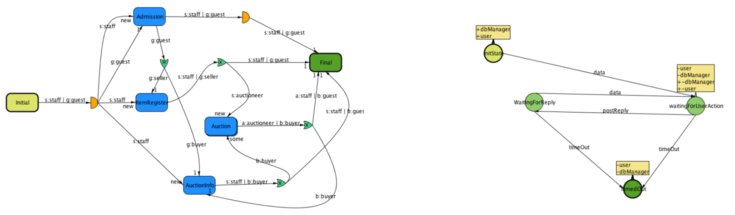


Fig. 1. Left: the institution-specification of an auction house with six scenes. Right: the specification of a Scene-protocol. Both were created with Islander.

component that executes the institution. It comprises of a set of agents that control the execution of scene instances, and is responsible for assigning a Governor to each participating agent. An EI can be executed by starting AMELI with an EI-specification. When it is running, agents can join it by requesting entrance to the institution, and, once entered, they can start communicating according to the protocols of the institution.

Note that apart from designing the institution, one also needs to implement the agents that are going to participate in the institution. These agents could be implemented by third parties, interested in participating in the EI for their own reasons. For example, one may design an online fish auction as an EI, so that any potential buyer can develop its own agent to make bids according to its own bidding strategy.

3 Human Interaction

The goal of our work is to enable human users to participate in Electronic Institutions, interacting with one another as well as with software agents. One can for example imagine an auction house in which bids are made by humans, but in which the tasks of the auction house, such as registration of participants and leading the auction are taken care of by automated agents.

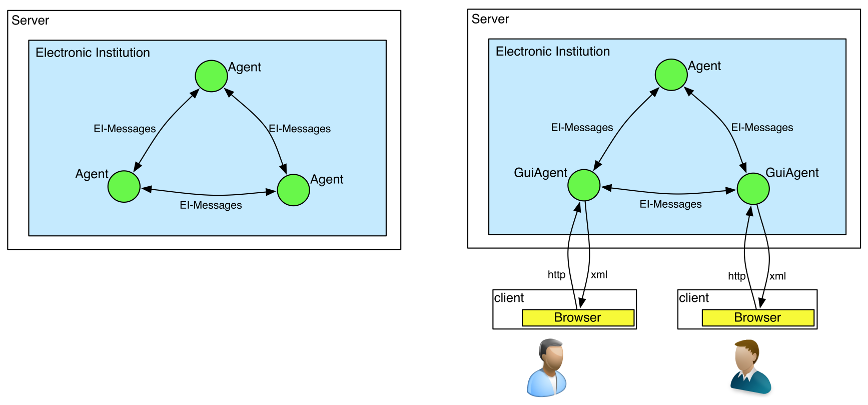


Fig. 2. Left: a 'classic' EI with only software agents. Right: an EI with one software agent and two users.

3.1 Our Framework

A human user would interact in an EI by clicking buttons in a browser window. To allow these actions to have effect in the EI, we have implemented a software agent that represents the user inside the EI and that executes the actions requested by the user. This agent is called the *GuiAgent*. Its current implementation does not do anything autonomously, but, if necessary, it can be extended with more sophisticated capabilities, such as giving intelligent strategic advice to the user.

When developing the framework we took into account that, on one hand, one may want to have a good-looking GUI that is specifically designed for a given institution. But, on the other hand, one may not want to develop an entirely new GUI for every new institution, or one may want to have a generic GUI available to test a new EI during its development, so that one can postpone the design of its final GUI until the EI is finished. Therefore, our framework allows for both. It generates a GUI automatically from the EI-specification, but at the same time provides an API that enables web designers to easily create a custom GUI for every new EI. The framework is used on top of the existing EI-framework and consists of the following components:

- A Java agent called *GuiAgent* that represents the user in the EI.
- A Java component that encodes all relevant information the agent has about the current state of the institution into an xml file.
- A Javascript library called *EiGuiInterface* that translates the xml file into a Javascript object called *EiStateInfo*.
- A Javascript library called *DefaultGuiGenerator* that generates a default Graphic User Interface (as html) based on the EiStateInfo object.

3.2 How it Works

In order for a user to participate in an institution, there must be an instance of that EI running on some server. To join the institution, the user then needs to open a web browser and navigate to institution's url. The process then continues as follows:

1. A web page including the two Javascript libraries is loaded into the browser.
2. The page sends a login request to the server.
3. Upon receiving this request the server starts a GuiAgent for the user and, depending on the specific institution, other agents necessary to run the institution.
4. When the GuiAgent is instantiated it analyzes the EI-specification to retrieve all static information about the institution.
5. The page starts a polling service that periodically (typically several times per second) requests a status update from the GuiAgent.
6. When the GuiAgent receives a status update request it asks its Governor for the dynamic information about the current status of the institution.
7. The GuiAgent converts both the static and the dynamic information into xml which is sent back to the browser.
8. The DefaultGuiGenerator Javascript library then uses this information to update the user interface (more information about this below).
9. The user can now execute actions in the institution or move between its scenes by clicking buttons on the web page.
10. For each action the user makes, a http-request is sent to the GuiAgent.
11. The GuiAgent uses the information from the http-request to create an EI-message which is sent like any other message in a standard EI.

As explained, the GuiAgent uses two sources of information: static information from the EI-specification stored on the hard disk of the server and dynamic information from the Governor. The static information consists of:

- The names and protocols of the scenes defined in the institution.
- The roles defined in the institution.
- The ontology of the institution.

While the dynamic information consists of:

- The current scene and its current state.
- The actions the user can take in the current state of the scene.
- For each of these actions: the parameters to be filled out by the user.
- Which agents are present in the current scene
- Whether it is allowed to leave the scene and, if yes, to which other scenes the user can move.

Http-Request	**Description**
/login?name=alice&role=guest	Enter the EI with given name and role.
/sendMessage?name=alice&receiver=bob &msg=bid&amount=1000	Send a message with given parameters.
/gotoScene?name=alice&role=guest &sceneName=Admission	Enter the given scene with the given role.
/exitScene?name=alice	Exit the given scene.
/gotoTransition?name=alice &transitionName=transition1	Go to the given transition.
/request_update?name=alice	Request an update of the status of the EI.

Fig. 3. The http-requests sent from the browser to the GuiAgent

3.3 Generating the GUI

Every time the browser receives information from the GuiAgent, it updates the GUI. This takes place in two steps, respectively handled by the two Javascript libraries. In the first step the EIGuiInterface converts the received xml into a Javascript object called EiStateInfo, which is composed of smaller objects that represent the static and dynamic information as explained above.

In the second step the EiStateInfo-object is used by the DefaultGuiGenerator library to draw the GUI. This GUI is completely generic, so it looks the same for every institution. If one requires a more fancy user interface tailored to one specific EI, one can write a new library that replaces the DefaultGuiGenerator.

The fact that these two steps are handled by two different libraries enables you to reuse the EIGuiInterface when designing a new GUI, so you doe not have to worry about how to retrieve the relevant information from the EI. All information will be readily available in the EiStateInfo-object, so you only need to determine how to display it on the screen.

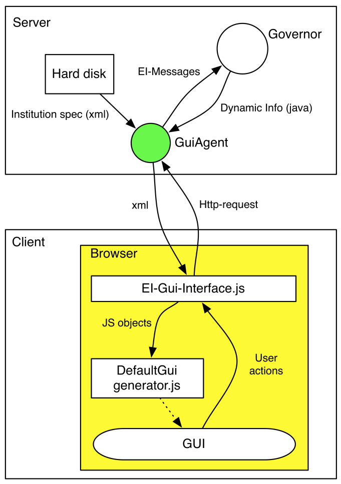


Fig. 4. The components necessary to generate the GUI. Solid arrows indicate exchange of information. The dashed arrow indicates that the GUI is created by the Default-GuiGenerator

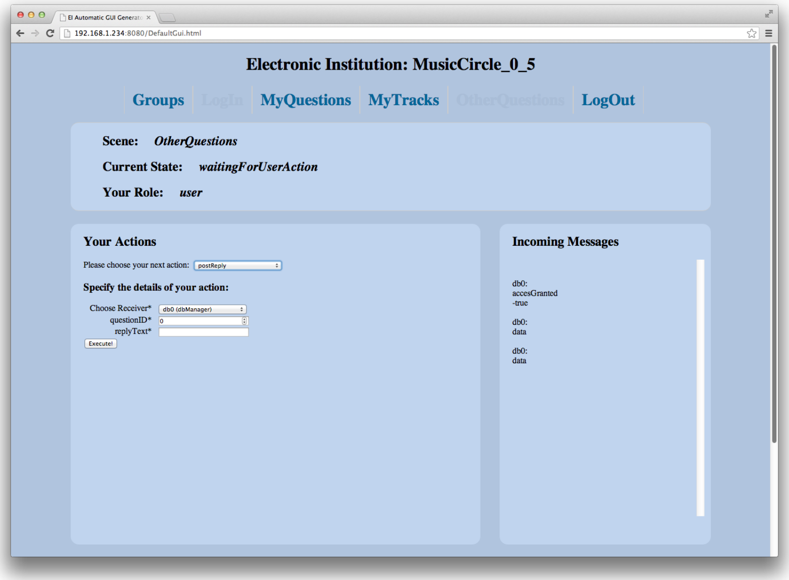
3.4 The Default User Interface

The default user interface is displayed in Figure 5. It is divided in four sections:

– A menu bar in the top that allows for navigating from scene to scene.
– A panel showing general information about the status of the user: the scene in which it currently is, the state of the scene, and the role the user is playing.
– A panel to display the messages coming in from other users and agents.
– A panel where the user can choose which action to take (i.e. which message to send), and fill out the parameters.

The figure shows the GUI for an agent participating in the MusicCircle institution (see Section 5). This institution has six scenes, hence the menu bar shows six menu options. Two of those menu items are grayed out meaning that the user currently cannot move to those scenes. If a user is present in more than one scene at the same time, the browser will have a separate tab opened for each scene. We have chosen to make navigation between scenes resemble as much as possible the way a user navigates between menu-items on a regular website.

The panel on the left is where the user interacts with the other users and agents in the EI. The user can choose which action to take from a drop down list. This list only shows those actions that the user currently can do, hence preventing the user from sending illegal messages (note however that even if the user would be able to send illegal messages, they would still be blocked by the Governor. But for the sake of user-friendliness we only want to display messages that the user can indeed send).



Electronic Institution: MusicCircle_0_5

Scene: *OtherQuestions*

Current State: *waitingForUserAction*

Your Role: *user*

Your Actions

Please choose your next action: postReply

Specify the details of your action:

Choose Receiver*	db0 (dbManager)
questionID*	0
replyText*	

Incoming Messages

db0:
accesGranted
-true

db0:
data

db0:
data

Fig. 5. The Default Gui, applied to the Music Circle institution

Once the user has selected an action to take, the browser displays a form where the user can fill out the necessary parameters. Since each action in the institution is modeled as a message, the user needs to choose which agent is going to receive the message. The EI-specification however may restricts which agents can receive which kind of message. A bid in an auction house for example should always be sent to the agent playing the role of auctioneer.

The form shows one input control for each parameter of the message. The type of control depends on the type of the parameter. For example, if the the parameter is of type integer, a numeric input control appears, while if the parameter is of type string, a text box appears. In case the parameter is of a user-defined type, a sub-form appears with several controls, one for each of the variables of the user-defined type.

3.5 Customizing the GUI

A customized GUI-generator can retrieve all necessary information from the EiStateInfo-object. For example: if the user chooses to make a bid in an auction, the GUI-generator would read from the EiStateInfo-object that an Integer parameter must be set to represent the price the user wants to bid. The programmer of the GUI-generator should make sure that whenever a parameter of type Integer is required, the GUI displays an input-control that allows the user to introduce an integer value.

The fact that one can also define user-defined types in an EI adds a lot of flexibility. Suppose for example that one would like a user to record an audio file and send this in a message to an other agent. Electronic Institutions do not support audio files by default. However, the institution designer may define a new type with the name ‘Audio’. Once the user chooses to send a message that includes audio, the EiStateInfo-object will indicate that a parameter of type Audio is needed. A customized GUI-generator could then be programmed such that a microphone is activated whenever this type of parameter is required.

4 Social Networks as Electronic Institutions

In the past few years there has been an enormous increase in the popularity of social networks and many different ones nowadays exist next to each other, such as Facebook, Twitter, LinkedIn and Couch Surfing. Although they are all based on the same idea: making contacts and sharing information with them, each of these networks applies different protocols and has different interpretations of what it means to have a connection. While on Facebook a friend is someone you share your pictures with, a connection on LinkedIn is someone you share your CV with. In this section we make two important observations:

– Social networks are institutions, each with their own norms and protocols.
– The flexibility of Electronic Institutions allows for a more generic type of social network, in which the users can determine their own norms.

Clearly, different kinds of relationships require different rules of behavior, and we claim that this is an important reason why so many social networks exist simultaneously. To illustrate this we will next compare two popular social networks: Facebook and Couch Surfing, regarding to their respective norms and protocols.

4.1 Norms and Protocols of Facebook and Couch Surfing

Facebook is mainly designed for friends to share social activities with each other. Due to the informal nature of these activities, like sharing pictures and playing games together, it does not require very strict norms.

– **Meaning of friendship:** Friends can see each others’ pictures.
– **Protocols:** Becoming friends requires only two actions: one person requests the friendship, the other accepts it.
– **Norms:** Users have full control over their profiles: you can remove anything that anyone else writes on your profile.
– **What could go wrong:** You may by accident share pictures with someone you don’t like.

Couch Surfing is a social network for travelers [2]. The main idea of this network is that when you go traveling, instead of booking a hotel, you find somebody at your destination who would be willing to host you for free in his or her house.

When planning a trip you can search for profiles of people at your destination and if you like somebody's profile you can request him or her to host you.

While Facebook focuses on *online* shared experiences with friends *you already know*, Couch Surfing focuses on meeting *new people*, in *real* life. This means that Couch Surfing requires much stricter policies than Facebook. After all, hosting a complete stranger in your house, or being hosted by a stranger, can be dangerous (cases are known of women getting raped using Couch Surfing [3]). Becoming friends on Couch Surfing therefore requires more effort, and there are several mechanisms to verify the trustworthiness of members that Facebook lacks, discussed for example in [7].

- **Meaning of friendship:** I trust this person, so you can safely host him.
- **Protocols:** To become friends, you need to indicate how well you know the other person, how much you trust him or her and specify details on how and where you met.
- **Norms:** If somebody posts a negative comment about you on your profile, you cannot remove it.
- **What could go wrong:** Hosts may get robbed by their guests, or worse.

4.2 Designing EI-based Social Networks

The fact that different kinds of social contacts require different protocols and norms, has lead to the creation many different social networks, even though they often have overlapping communities of users. We now discuss how the application of Electronic Institutions could make an end to this inefficiency by allowing users to set up new sub-communities within a given social network, and invent their own set of norms and protocols for these sub-communities, without having to create an entirely new website.

One problem we have to overcome when implementing a social network as an EI is the fact that Electronic Institutions are based on the assumption that all users that communicate with each other are together in one scene instance. Social networks on the other hand have a much more asynchronous design, in which it is not assumed that users are not online at the same time: when you share an image with a friend, you are in fact uploading it to a database. Your friend will not see it until he or she also appears online. At that moment the image is automatically downloaded from the database to your friends browser, so that he or she can see it.

To overcome this discrepancy we have come up with the following design: each activity a user can do in an EI-based social network takes place in a scene instance where the user and the database are both represented by an agent. So in each scene instance there are exactly two agents: the GuiAgent and a DataBaseAgent. When the user uploads a picture, for example, this is modeled as a message which is being sent from the GuiAgent to the DataBaseAgent. When the other user appears online his or her GuiAgent will also enter a scene together with a DataBaseAgent, and the image will be sent from the DataBaseAgent to the GuiAgent. A second problem we needed to tackle is that the current

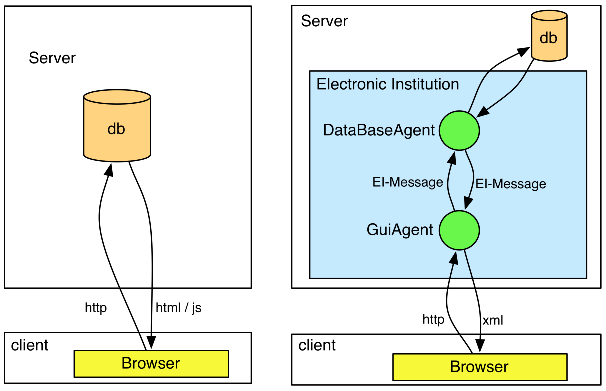


Fig. 6. Left: a standard website. Right: an EI-based website.

implementation of Electronic Institutions does not allow for any bulk data (i.e. images, video or audio) to be sent in a message. We therefore have to send the data itself outside the institution. The action of sending this data however, is still represented as a message inside the EI, so that the EI can still verify whether the user is actually allowed to undertake that action. We just need to make sure that when such a message is blocked, this is also prevents the user from sending the actual data. This can be achieved for example by disabling an upload button.

Finally, one more problem to tackle is the fact that moving from one scene to another scene in an EI is made in three steps: first you exit the current scene, then you move to a so called *transition* and finally you move to the new scene. We think it is very user-unfriendly, since going from one web page to another is usually done with one single menu click. The reason for this 3-step process is that it allows agents to choose to move into more than one scene instance at a time, or to synchronize with other agents before moving into the new scene. Although this is fine for software agents, we think this is overly complex for human users, and not necessary for the application to social networks.

We have solved this by making sure that these three steps are all triggered automatically, one by one, by a single click on a menu-item. The downside of this, however, is that it removes the possibility for the user to make any choices at the transition. Also it could happen that a user chooses to move to a scene instance that is not available. A scene can be unavailable because one needs to wait for other agents to participate in the same scene. However, in the model described above the GuiAgent and its associated DataBaseAgent are always together and never need to participate in any scene instance with any other agents. Therefore, as long as we stick to this model no scene can ever be unavailable.

5 Case Studies: MusicCircle and WeBrowse

As a test case we are applying our EI-technology to a social network for online music learning, called *MusicCircle*, which is currently under development. On this website students can learn to play an instrument, with and from their friends, in a community-driven way. Imagine for example a student who is learning to play the piano. He or she can play a piece and record it, and then upload it to the social network. The student can then ask the other community-members for feedback. These community members may be friends of the student, professional music teachers or even automated music analyzing agents.

MusicCircle will allow users to create their own sub-communities. For example, one can set up a community for guitar players, or for jazz-musicians, or a community consisting only of your colleagues from work. We are providing the EI framework underneath this social network. This will enables set or change the norms of their communities as they wish.

Some communities may for example have serious users and strict norms, because their members want to study seriously, and want to discuss their play with other serious students. Other communities may be much more non-committal, consisting of hobbyist who just want to spend some free time playing music without taking it too seriously. A few examples of norms that a community may set, could be the following:

- This group is only for *advanced* guitar-players: to join, you need to be at least at stage 10 of the guitar course.
- Only active users can receive feedback: to be able to receive feedback you must give feedback to others at least 5 times a week
- Experts will only help serious students: if want to get help from an expert player, you need to practice at least 3 times a week to request his help.

Another social application where we are applying our technology is the *WeBrowse* application [6], [10] which enables friends to simultaneously visit a museum online, each from his or her own mobile device. It allows friends, even though they are in different locations, to have a *shared experience* when they visit the museum. The users see the same artifacts on their respective computer screens, and they can see each others' actions, such as zooming in on an object, or adding tags and 'likes'.

Furthermore, the users need to make joint decisions on what to see or do next in the museum. Since people may have different opinions on this matter, protocols are needed to determine how individual decisions and opinions are aggregated into social group decisions that are acceptable to all group members. The underlying EI regulates these protocols.

6 Conclusions

We have managed to add a new tool to the existing EIDE framework that allows humans to interact in an with each other and with software agents in an

Electronic Institution. The tool generates a user interface automatically, so the creator of an institution can directly use it without having to design anything. However, it still allows designers to design a custom GUI, tailored to a specific institution, without having to worry about getting the necessary information from the EI. Furthermore, we argue that social networks could highly benefit form this technology. We are currently applying it to two projects related to social networks that will learn how useful it really is in practice.

7 Acknowledgments

This work is supported by the Agreement Technologies CONSOLIDER project, and CHIST-ERA project ACE, EU project 318770 PRAISE, and the CBIT project (TIN2010-16306).

References

1. Arcos, J.L., Esteva, M., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Engineering open environments with electronic institutions. Engineering Applications of Artificial Intelligence 18(2), 191–204 (2005)
2. CouchSurfing: http://www.couchsurfing.org (2012)
3. DailyMail: http://www.dailymail.co.uk/news/article-1205794/rape-horror-tourist-used-couchsurfing-website-aimed-travellers.html#ixzz29y3wxuck (2012)
4. d'Inverno, M., Luck, M., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Communicating open systems. Artificial Intelligence 186, 38–64 (03/2012 2012), `http://www.sciencedirect.com/science/article/pii/S0004370212000252?v=s5`
5. Esteva, M.: Electronic Institutions: From Specification to Development. Ph.D. thesis, Technical University of Catalonia (2003)
6. Hazelden, K., Yee-King, M., Amgoud, L., d'Inverno, M., Sierra, C., Osman, N., Confalonieri, R., de Jonge, D.: Wecurate: Designing for synchronised browsing and social negotiation. Dubrovnik, Croatia (15/10/2012 2012)
7. Lauterbach, D., Truong, H., Shah, T., Adamic, L.: Surfing a web of trust: Reputation and reciprocity on couchsurfing.com. In: Computational Science and Engineering, 2009. CSE '09. International Conference on. vol. 4, pp. 346–353 (2009)
8. Noriega, P.: Agent Mediated Auctions: The Fishmarket Metaphor. Ph.D. thesis, Autonomous University of Barcelona (1997)
9. Trescak, T., Rodriguez, I., Sanchez, M.L., Almajano, P.: Execution infrastructure for normative virtual environments. Engineering Applications of Artificial Intelligence 26(1), 51 – 62 (2013), `http://www.sciencedirect.com/science/article/pii/S0952197612002540`
10. Yee-King, M., Confalonieri, R., de Jonge, D., Hazelden, K., Sierra, C., d'Inverno, M., Amgoud, L., Osman, N.: Multiuser museum interactives for shared cultural experiences: an agent based approach. Saint Paul, Minnesota, USA (06/05/2013 2013)