Perspectives on the Viable Mobile Virtual Community for Telemedicine

Jan-Willem van ’t Klooster
Pravin Pawar
Bert-Jan van Beijnum
Chariz Dulawan
Hermie Hermens
Remote Monitoring and Treatment Group, University of Twente, the Netherlands.

INTRODUCTION

A virtual community is an electronically supported social network: it can be seen as a group of people who have regular social interaction, independent of time and space, because of a common interest such as a problem, task, or feeling exchange (Eysenbach, Powell, Englesakis, Rizo, & Stern, 2004; Rheingold, 1993). When independence of time and space is achieved through the use of mobile devices and wireless communication technologies, such a virtual community is called a Mobile Virtual Community (MVC). Existing research interests in the MVC domain are grouped into technology-centered interest, user-centered interest and business-centered interest (El Morr & Kawash, 2007). The technology-centered aspects include issues such as platform design, development framework, mobile network bandwidth limits and intelligent agents. The user-centered issues include user interface, behavior, personalization, privacy, data security and trust. Business-centered aspects include marketing, investment and business models.

In another paradigm known as telemedicine, information and communication technologies are being investigated and employed in applications such as health discussion & maintenance, alleviation, cure and prevention of diseases. In recent telemedicine scenarios, sensors attached to the patient’s body collect patient’s vital signs, transmit them to a mobile gateway device being carried by the patient, which in turn uses wireless communication technologies to transmit the data to a healthcare center, for purposes like vital signs analysis and offering emergency assistance to the patient if needed. The sensors and the gateway mobile device together form a so called Body Area Network (BAN). Konstantas, van Halteren, & Bults (2004) describe such a BAN for telemedicine purposes. Other supporting actors involved in such a scenario are the technicians, healthcare specialists, doctors and (informal) caregivers.

According to the definition, the telemedicine scenario may be viewed as a virtual community if the patient and other actors could communicate with each other for the purpose of providing medical assistance and counseling to the patient. In situations where patient and caregiver mobility exists, this virtual community, may be said to correspond to a MVC. Such a MVC is an aggregated kind of community as defined in Demiris 2006, where communities of healthcare professionals only, patients/informal caregivers only, combinations of them, and general public communities are defined. The technical system that supports the realization of MVC is referred to as MVC platform (Van Beijnum, Pawar, Dulawan, & Hermens, 2009).
Considering the architecture of the MVC platform, it can be argued that the MVC potentially evolves around a set of services based on the principles of Service Oriented Architecture (SOA) (Papazoglou, 2003; Pawar, Subercaze, Maret, Van Beijnum, & Konstantas, 2008).

Based on the findings of Broens et al., (2007) and Maloney-Krichmar and Preece, (2005) we argue that to be a viable MVC, it should have a tailored focus and robust technical platform: it should be of clear interest to the users and the technology incorporated should be reliable. This chapter contributes to this area in general and mobile patient monitoring and treatment in particular, by 1) analyzing in detail the robustness and other requirements to be fulfilled by the technical platform for MVCs, 2) providing guidelines for MVC platform development based on service orientation, and 3) discussing the actors, front-end views and service components involved.

The reminder of this chapter is organized as follows. The second section of the chapter illustrates a possible telemedicine scenario focused on patient monitoring and treatment which help to elicit the specific requirements to be supported by the MVC platform. Based on the requirements and services elicited in that section, the third section of this chapter presents a possible graphical user interface (GUI) for the platform depicting the requirements to be fulfilled from an end-user perspective. The fourth section discusses the internal design of the possible MVC platform and conclusions are presented in the last section.

**SCENARIO BASED REQUIREMENTS ELICITATION**

Scenario analysis is the process of understanding, analyzing, and describing system behavior in terms of particular ways the system is expected to be used (Hsia et al., 1994). Drawing use cases from the scenarios and relating requirements to the use cases is a popular approach in the system design process (Whittle & Krüger, 2004). We use a similar approach here to elicit the requirements for the MVC platform based on a scenario.

**Scenario description**

Herewith we present a visionary scenario showing the intended use of the MVC platform. On the MVC platform, a number of different sub-communities could function independent of each other. Member can join these sub-communities as well as the aggregate MVC. One of the sub-communities is used by the persons Bob and Alice in the following scenario.

*Bob (patient) and Alice (caregiver) join a MVC. The local healthcare center creates a sub-community called as telemedicine community. The MVC community management service recommends Bob, Alice and other members to join the telemedicine community. Bob and Alice join the community by updating their profile with information specific to the telemedicine community (e.g. problems Bob is suffering from). The MVC matchmaking service recommends the members to be a part of each other’s social network based on their telemedicine community centric profile (e.g. interest in the same health condition). Bob, Alice and other members of the telemedicine community, including patients and (informal) caregivers, socialize with each other for sharing their experiences after joining the same sub-community. Meanwhile, the healthcare center announces offering of a mobile patient monitoring service for the patients suffering from epilepsy. To facilitate mobile patient monitoring, a number of BAN manufacturers recommended by health-care center offer their BAN to the patients. Using the MVC context-aware matchmaking service, the MVC platform recommends Bob a particular BAN which is compatible with his smartphone. Bob purchases the recommended BAN and subscribes to the mobile patient monitoring service via the community platform. On the day Bob suddenly suffers from an epileptic seizure; the medical specialist at the healthcare center requests the MVC platform to search for the nearest caregiver. On this request, the MVC context-aware matchmaking service searches for the nearest caregiver service based on Bob’s and caregivers’ current locations. Fortunately, Alice is the nearest informal caregiver to Bob. The medical*
specialist instructs the MVC platform to notify Alice about Bob’s critical condition. Alice receives the notification and reaches to Bob’s location for providing assistance.

**Use cases**

Based on the given scenario, the following use cases can be derived. These use cases not only show possible uses of the MVC platform, but also identify who are involved and what are the tasks in fulfilling the use case. As elicited from the scenario, Figure 1 identifies the uses cases and the involved actor role(s), being patient, caregiver, medical specialist, product provider or service provider. For further illustration, a particular use case in which a sub-community is created is described in detail in Table 1. In this use case, it is shown that parameters of the new sub-community can be set, including policy rules for the members and whether or not anybody can join the sub-community (public) or only on invitation (private) depending on e.g. the topic and privacy sensitivity of the sub-community.

**Figure 1: Use Cases Derived from the Scenario and Corresponding Roles**

**Table 1: Description of the use case Create Sub-community**

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Create sub-community</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precondition</td>
<td>Medical specialist creating the sub-community is known to the MVC platform.</td>
</tr>
<tr>
<td>Success End-condition</td>
<td>Medical specialist creates the sub-community with specific criteria.</td>
</tr>
<tr>
<td>Failed End-condition</td>
<td>Medical specialist is not able to create the sub-community.</td>
</tr>
<tr>
<td>Actor role</td>
<td>Medical specialist</td>
</tr>
<tr>
<td>Description</td>
<td>1. The medical specialist logs in to the MVC platform using his/her username and password.</td>
</tr>
<tr>
<td></td>
<td>2. The medical specialist makes use of a module in the MVC platform that allows him/her to create a sub-community.</td>
</tr>
<tr>
<td></td>
<td>3. The medical specialist identifies and/or selects appropriate conditions for the sub-community to be created. Such conditions include which roles and services should be present; what policies govern the interactions between roles, and whether the sub-community is private/public.</td>
</tr>
<tr>
<td></td>
<td>4. The medical specialist saves information and creates the sub-community.</td>
</tr>
</tbody>
</table>
Requirements elicitation

In this section, the use cases derived in the above section are mapped to the high-level requirements to be fulfilled by the MVC platform. An overview of these requirements is provided in Table 2. As seen in this table, the platform should be able to perform various actions and provide various services. Afterwards, this section lists the services and policies necessary to meet these high level requirements.

Table 2. MVC Platform Requirements and Use Case Mapping

<table>
<thead>
<tr>
<th>High Level Requirement</th>
<th>Use Case(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The platform should allow creation of a sub-community.</td>
<td>Create sub-community</td>
</tr>
<tr>
<td>The platform should allow management of a sub-community such that the preferences and constraints can be made regarding members, roles and services.</td>
<td>Create sub-community</td>
</tr>
<tr>
<td>The platform should provide asynchronous and synchronous communication services such as email and instant messaging for inviting existing community members to join sub-community.</td>
<td>Join MVC, Join sub-community</td>
</tr>
<tr>
<td>The platform should allow creation and management of profiles.</td>
<td></td>
</tr>
<tr>
<td>The platform should provide synchronous and asynchronous communication service such for inviting non-community members to join the main community and transmission of information.</td>
<td>Send notification, receive notification, Social interaction</td>
</tr>
<tr>
<td>The platform should allow product offering and product purchase.</td>
<td>Product offering, product purchase</td>
</tr>
<tr>
<td>The platform should allow service offering and service subscription.</td>
<td>Service offering, service subscription</td>
</tr>
<tr>
<td>The platform should be able to identify who should be alarmed in case of an emergency.</td>
<td>Search caregiver</td>
</tr>
<tr>
<td>The platform should have information about the locations of the patient and caregiver.</td>
<td>Search caregiver</td>
</tr>
<tr>
<td>The platform should be able to support multiple devices (e.g. smartphones) associated with multiple services.</td>
<td>Social interaction, service offering</td>
</tr>
</tbody>
</table>

Services and Policies

As described in the Introduction, the architecture and software design of the MVC can be made using SOA principles. A service refers to a unit of work which can be done by a service provider to achieve desired end results for a service consumer (Subercaze et al., 2009). In traditional virtual communities, services would refer to web-based (interaction) services. However, the impact of mobile technologies makes available a number of mobile services, which are both produced and consumed using mobile devices. Based on the requirements listed above, we identify two groups of services: mobile device services and MVC platform services. Mobile device services run on the mobile device and platform services are those provided by the MVC platform.

Policies refer to the rules or constraints imposed on either the actor roles in the MVC or in the interaction with the services. In terms of the actor roles, there are specific guidelines about what an actor can do and can not do. For example, the product provider can not search for caregivers. Thus, for the interactions between a role and a service, rules could be related to permission constraints and prohibition constraints. Permission constraints refer to the prescription that a certain interaction is allowed to occur. A prohibition constraint is the opposite as it describes an interaction that must not occur at all. Specifying policies is also based on the analysis of scenarios, because policies depend on the purpose and use of the MVC as well as on the roles and services that are present in the MVC.
The following listing provides required services in the MVC platform architecture:

- **Member Management Service**: This service includes the invited registration of the new members, editing and managing member profiles, the member type (e.g. patient, caregiver), logging in, session handling etc.

- **Directory and announcement Service**: This service provides functionality for the community support providers to post news, list the offered services, and listing of events such as those leading to improvement in the psychological and physical health of the patients.

- **Purchase Service**: This service allows the registration and acquisition of health-related products and services offered in the MVC. This process needs to be moderated by the healthcare center to assure relevance for the members and to avoid that the MVC becomes a brand-oriented advertising channel, therefore a content moderation service is required as well.

- **Content moderation service**: Certain types of the contents which negatively affect the psychological and physical health of the patients are discouraged. Hence, the contents such as profile information, pictures posted by the members, product and services description are subject to publishing in the MVC after manual moderation. Automatic moderation techniques could be applied, for interactions such as instant chat. These features will be taken care of by the content moderation service.

- **Alarm Service**: This service enables the alarms, based on a predefined level of urgency. In case of an emergency, this service can be used to notify for example a caregiver.

- **Community Management Service**: This service consists of all the functionalities required to create, join, access and search sub-communities, (such as those of patients with a particular type of condition), publish, get and subscribe to information in the existing sub-communities.

- **Policy making and enforcement service**: To enforce the interactions between an actor role and a service, as well as to take into account the trust and privacy requirements in the MVC community, a set of policies need to be developed and enforced. The policy making and enforcement service takes care of the matters such as creation of a new policy and enforcing these policies during the MVC interactions.

- **Social Interaction Service**: This service handles the one-to-one, one-to-many and many-to-many interactions between the MVC members. This includes interaction functions such as instant messaging, group notifications, and subscription to the particular type of content (e.g. information posted by the medical specialist).

- **Context-Aware Matchmaking Service**: Semantic descriptions of the member profiles combined with description logic are powerful tools to perform matchmaking. The context-aware matchmaking functionality of this service could be used for example to recommend new members in the sub-community, or to search for the nearest available caregiver.

- **Content Exchange Service on the Mobile Device**: This service on the patient’s mobile device is aimed at sending the contents (e.g. text, images, and streams) generated at the mobile device to the community platform such that this content could be published in the community. Similarly, this service could also request/subscribe to the community content the user is interested in and present this content for user viewing.

- **Vital Sign Monitoring Service**: This service enables the monitoring of vital signs, such as blood pressure and oxygen saturation information.

- **Context Information Service on the Mobile Device**: This service obtains context information (such as location) of the patient and sends this information and subsequent context changes in real-time to the community platform. This information could subsequently be used by the context-aware matchmaking service.

- **Community Service**: This service indexes and allows modifications on what services are available to which sub-community.

- **Chat service**: As a sub-part of social interaction service, this service allows for instant voice, video or message chat amongst members of the MVC.
FRONT-END DESIGN OF THE MVC PLATFORM

To illustrate how the MVC platform supports the scenario presented in the above section, a design of how to specify some of the preferences is provided in this section. In Figure 2, the creator of a sub-community is asked to establish a set of policy rules related to the roles in the sub-community. As shown, different roles and cardinalities are entered, and rules for invitations to the sub-community apply.

![Figure 2: Specification of roles in a sub-community](image)

Next to specifying roles, services available for the sub-community should be selected. Alarm service, Chat Service, Location Service, Viewing Service, and Vital Sign Monitoring Service are examples of services for the sub-community as applicable in the scenario. These are shown in Figure 3.

![Figure 3: Specification of services in a sub-community](image)

In a chat service or other similar social interaction services, policies exist such as who initiates contact to whom. These policies are related to the purpose of the chat session. Figure 4 shows how this can be specified.

![Figure 4: Specifying policies for a chat service](image)
Regarding the location service, not all actors are allowed to view the location of other actors. For instance, a service provider does not need to know the location of a medical specialist, but a medical specialist should be able to know the patient’s location in an alarm situation. These policies define who is the provider of the location information as well as who are the authorized consumers of this information. Figure 5 shows how this concept can be specified.

**INTERNAL DESIGN OF THE MVC PLATFORM**

To support the desired functionalities of the platform described so far, a sound internal design is necessary. This section presents details on the possible internal design of the MVC platform. An overview of the high-level architecture of the MVC platform is presented in Figure 6. In order to support mobility in the platform, three layers are identified:

- **Platform Services Layer**: The platform services layer is responsible for providing the MVC platform services, for example those described in the requirements elicitation section.
- **Mobile Services Layer**: The mobile services layer is responsible for making available the MVC platform services to the mobile device and for providing services such as content exchange and context information service from the mobile device to the MVC.
- **Integration Layer**: Because of the use of different nature and technologies at the platform services layer and mobile services layer as well as to take into account the effects of the mobility, an integration layer is envisioned to support the mobile and platform services cooperation.

**Figure 6: Overview of the Layers in the Mobile Virtual Community Platform**

Internally, the detailed software architecture of the MVC platform could be represented using the Unified Modeling Language (UML). A UML class diagram is a type of diagram that describes the internal structure of a system by showing the system's classes, their attributes, and the relationships between the classes. One of the UML class diagrams for creating a sub-community is presented in Figure 7, with a description of the classes and their relationships in Table 3. A design decision is made to show that parties (actors) have no direct access on the services. The services are linked to the sub-communities since the motivation for using these services should be to support the purpose of the existence of the sub-
community was. In the same manner, not all roles have access to the sub-community services. Only authorized roles and parties as set in the policy for using the services are allowed to utilize the service.

Figure 7: UML class diagram representing MVC internal structure for the creation of a sub-community

Table 3: Description of the UML classes shown in Figure 7

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party</td>
<td>The party class refers to the parties (actors) of the MVC platform.</td>
</tr>
<tr>
<td>PartyRole</td>
<td>This model class shows the association type between a party and a role. The relationship shows that a party can take up multiple roles and that this is a requirement before a party takes up a role defined in a certain sub-community.</td>
</tr>
<tr>
<td>Role</td>
<td>The list of roles and its description are represented in this class.</td>
</tr>
<tr>
<td>CommunityRoles</td>
<td>This model class identifies the distinct roles that can only participate in the sub-community. These roles can be filled up by party roles.</td>
</tr>
<tr>
<td>Community</td>
<td>This model class refers to the sub-communities that are created in the MVC platform and which uses the services of the MVC platform. A sub-community may use multiple services in order to fulfill its goals. This class knows the name, description and purpose of the sub-community as well as the attribute related to the degree of openness of the sub-community.</td>
</tr>
<tr>
<td>CommunityRoleAssignment</td>
<td>The class contains the policy on role assignments in the sub-community.</td>
</tr>
<tr>
<td>CommunityServices</td>
<td>This model class refers to the services that are able to support the role of the sub-community.</td>
</tr>
<tr>
<td>Service</td>
<td>This model class refers to the related Telemedicine services that can be utilized by certain sub-communities and is a generalization of various concrete services.</td>
</tr>
</tbody>
</table>
CommunityService: This class refers to the related community service that maintains the mapping of Service elements to Community elements.

AlarmService: This class contains the policy on whom to notify depending on the alarm level.

ChatService: This class contains the purpose of the chat as well as the policy on who initiates the chat and to whom it was initiated.

LocationService: This class contains the policy on who provides location information and who can consume the provided information.

VitalSignService: This class identifies the vital sign that will be collected from the provider as well as the policy on who consumes the vital sign information.

ViewingService: This class contains the device that will be used for viewing the vital sign information. This service is used in conjunction with the VitalSignService, wherein the consumer role makes use of the device preference defined in this ViewingService in order to view the vital sign.

CONCLUSION

Mobile virtual communities (MVC) are being explored in the telemedicine domain for the purpose of social interactions and monitoring to support tasks such as health discussion & maintenance, alleviation, cure and prevention of diseases. In this chapter, we present perspectives on the MVC for telemedicine. By means of presenting a possible scenario and scenario based requirements elicitation, we identify requirements and services for such a MVC. The chapter shows how the architecture and software design of the MVC evolves around service oriented principles. Illustrative policy-related interface components are discussed. It is necessary to be able to define rules for the platform users, due to the nature of the actors involved (e.g. healthcare professionals, patients and product providers).

To incorporate mobile services in the platform, a layered high-level architecture is presented that enables a clear separation of concerns between mobile and platform services. Finally, the chapter presents an internal design and an elaboration on the internals of such a platform.

In sum, the chapter provides perspectives on how the viable MVC for telemedicine can be developed, based on extensive scenario analysis and using service oriented design principles.

ACKNOWLEDGMENT

This work is part of the IOP GenCom U-Care project (http://ucare.ewi.utwente.nl), sponsored by the Dutch Ministry of Economic Affairs under contract IGC0816, and the Freeband AWARENESS project (http://awareness.freeband.nl), sponsored by the Dutch government under contract BSIK 03025.

REFERENCES


**Terms and Definitions**

- **Mobile Virtual Community** (MVC): A virtual community (VC) is an electronically supported social network: it can be seen as a group of people who regularly interact because of a common interest, problem or task assignment. When independence of time and place is achieved through utilizing mobile devices and wireless communication technologies, such a VC is called a Mobile Virtual Community.

- **Telemedicine**: In telemedicine, information and communication technologies (ICT) are researched and employed in medicine areas such as health maintenance, alleviation, cure and prevention of diseases. Originally, telemedicine is a combination of ‘tele’, meaning (geographical) distance, and medicine. However, next to distance, also time can be bridged using ICT.

- **Body Area Network** (BAN): A BAN is a network of devices around the body, consisting of multiple devices such as sensors and a gateway device.

- **Service**: A service refers to a unit of work which can be performed by a service provider to achieve desired end results for a service consumer. Services are produced and consumed by both fixed and mobile devices.

- **Service Oriented Architecture** (SOA): A SOA is an architectural principle based on the services concept, in which services performed by a services provider can be reused as a standalone component in a web-based architecture. A SOA can for example be implemented by web services technology.

- **Unified Modelling Language** (UML) class diagram: A UML class diagram is a diagram following the rules of the UML, and used to represent the detailed internals of an information system.
- **Use case:** A use case defines a use of a system. It is generally used to concretize instantiated behavior of the system.
- **Sub-community:** A sub-community is a subset of an VC, and refers to a community within the virtual community.
- **Permission and prohibition constraints:** Permission constraints refer to the prescription that a certain interaction is allowed to occur; whereas prohibition constraints refer to permissions of interaction that are not allowed to occur.