UI Delegation: The 3rd Dimension for Cross-Platform User Interfaces

Dagmawi Lemma Gobena
Addis Ababa University
IT Doctoral Program
Addis Ababa, Ethiopia
dagmawi.Lemma@gmail.com

Abel Gomes
University of Beira Interior
Covilhã, Portugal
agomes@di.ubi.pt

Dejene Ejigu
Addis Ababa University
IT Doctoral Program
Addis Ababa, Ethiopia
ejigud@yahoo.com

***Abstract*—Two of the prominent dimensions behind the development of cross-platform UIs are the UI distribution and UI migration. In UI distribution, since UI elements of a given application has to be distributed across more than one device, some UI elements can be even duplicated. In UI migration, the description and construction of UI elements are centralized using a client-server model of computing over a computer network. Thus, we end up having limitations with respect to scalability and maintainability of the computing environment. Also, UI distribution and migration mostly support explicit HCI for interactive systems. However, in ubiquitous computing, implicit HCI is the most desired interaction approach. In this paper, we present the theoretical concept of *UI delegation* as the third dimension that ideally supports implicit HCI and trans-modality by assuring autonomy of the platforms using a peer-to-peer model**.

Keywords – cross-platform UI; multi-platform UI; user interface design; ubiquitous computing.

I. INTRODUCTION

The amalgamation of various technologies to support the needs of new computing models has become prevalent in computing environments like ubiquitous computing. For example, in the ranking system shown in Figure 1, which we developed to rank and produce outcomes for athletes, we have the following: (1) results are redisplayed using a web application; (2) setting to radio frequency identification (RFID) system is made using a GUI application; (3) every athlete wears RFID tag that is uniquely encoded, hence it is possible to create a sort of implicit interaction between the athlete and the system; (4) the RFID reader box is configured using a monochrome display; (5) athletes can receive their results on their mobile phones, or on any other personal device they may use. Such amalgamation of various technologies results in heterogeneous environment.

Nowadays, heterogeneity of personal devices is inescapable. Heterogeneity is one of the characteristics of ubiquitous computing [1], and it is caused by the coexistence of various devices in the same computing environment. Furthermore, the heterogeneity is a result from the diversity of software, users, interaction modalities, and environments.

Nevertheless, Weiser's vision of ubiquitous computing, which demands that computer is an invisible servant [2] [3], has not been achieved yet [4]. With regards to the human-computer interaction (HCI), invisibility of computers can be achieved, partly through implicit HCI (*i*-HCI) [5] and context aware systems. On the other hand, the explicit HCI (*e*-HCI) development for interactive systems requires consideration of capabilities and constraints of diverse platforms and users, in addition to provide interaction modalities in a human fashion (e.g., speech, gesture, etc.)

The platform heterogeneity, together with additional needs of interaction modalities, as well as the proliferation of new technologies, poses unique challenges to designers and developers of user interfaces (UIs). Analyzing user profiles and platform capabilities and constraints in the usability engineering lifecycle [6] is certainly challenging due to the heterogeneity of platforms (devices and software) and users. Therefore, UIs are expected to be cross-platform. That is, a UI that runs on a certain platform (e.g., desktop screen) shall be able to appear on another platform (e.g., small handheld device) without losing its usability.

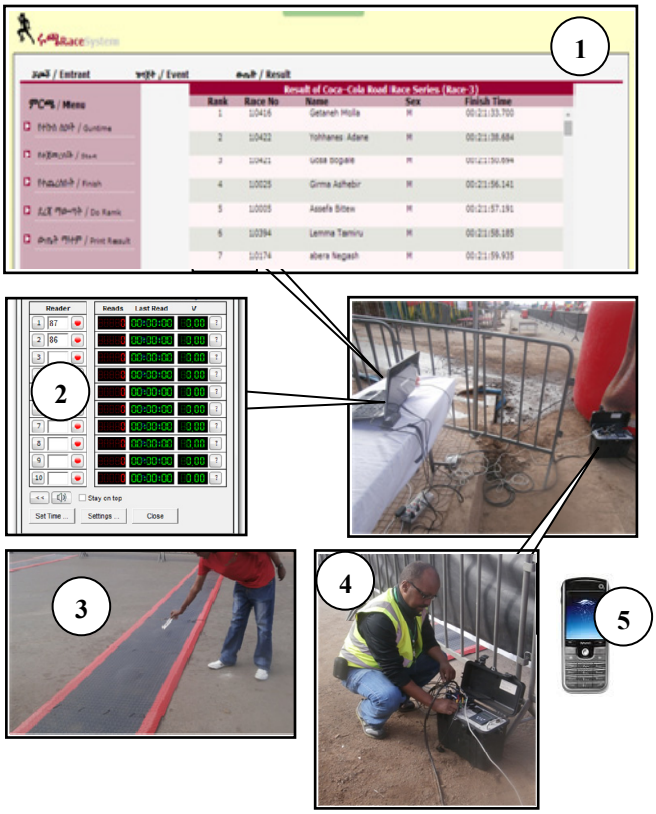

Figure 1. Heterogeneous race system for athletics.

 ISBN: 978-1-61208-382-7

To overcome the challenges of heterogeneity, there are models and theoretical frameworks suggested and developed in the HCI community, in order to sustain the notion of cross-platform UIs. In this paper, by cross-platform UI (respectively, cross-platform interaction), we mean UIs (respectively, interaction modality), no matter whether it is implicit or explicit, through which computers and users interact across various platforms in conformity with capabilities and constraints of users (user profiles) and platforms, without compromising the usability. Thus, cross-platform UIs should consider the context of the systems (e.g., applications, tools, interaction modalities, etc.), devices and users.

Two of the prominent dimensions behind the development of a cross-platform UI are those concerning UI distribution and migration. In the literature, we found that both approaches focus on a particular aspect of the heterogeneity – mostly the device [7]. However, generating UIs in a heterogeneous environment based on a specific context (e.g., device, user, task, interaction modalities, etc.) most likely reduces the usability of the system, which entails several usability issues [8]. Thus, we propose the concept of *UI delegation* as the third dimension to help in the development of cross-platform UIs.

Our motivation is based on three main points. Firstly, in order to automatically (or at design time) generate a cross-platform UI, we have to consider the merger of diversified contexts from the system (i.e., including interaction modalities, web services, etc.), device and user aspects, so as to meet usability requirements [6] [9]. Therefore, we found it important to introduce a different approach that compels the consideration of ternary views (the *system*, *device,* and *user*) in cross-platform UIs, but not in partiality of any of the views. In fact, though it is common practice to consider these three views in UI development, most works and techniques related to cross-platform UIs only focus on one of the views (user, device or system) at the time of automatically generating a specific UI. For example, the pattern based-approach proposed by Lei et al. [10] as well as responsive web development (RWD) focus on screen size adaptation, while Nichols et al. [11] focus on the functionality of the appliances, and Sauter et al. [12] only consider the device type.

Secondly, both distributed and migratory UI concepts are often implemented using the client-server model. While the final UI runs on the client side, the appropriate UI for a specific platform is generated at the server side [13]. Thus, the server is responsible to maintain the UI description [11] [14], to update and preserve the UI state [7], to store a duplicate version of the UI [12], and so forth. But, centralizing the description of capabilities of each platform often imposes limitations to the scalability and maintainability of the environment. Furthermore, in a ubiquitous environment, the peers are desirably autonomous, so that each peer shall be able to generate a UI as per its capabilities and its own autonomy.

Finally, heterogeneity may also be a result of the presence of various sorts of interaction modalities (including *i*-HCI). Nevertheless, most works we found in the literature are about *e*-HCI. Otherwise, despite the fact that the ubiquitous computing aims at invisible UIs [15], interaction modalities in the arena of *i*-HCI are not well covered. It has to be noted that *i*-HCI can be also achieved by using various technologies (i.e., sensors, motion capturing tools, etc.); and this in turn leads to another aspect of heterogeneity. Therefore, we need a new approach to the development of cross-platform UIs.

The purpose of this paper is to provide a new theoretical concept that complements the efforts made so far to support UI development for heterogeneous environments. We consider the problem of heterogeneity as a result from the need of collaboration between platforms (i.e., *device*-and-*system* units) that are owned or controlled by a *human user*. Hence, we focus on the concept of delegation as it is applied in [16] for supporting collaboration between agents in an agent-based environment. Accordingly, we propose the concept of *UI delegation* with autonomy.

Autonomy of nodes that collaborates in environments like ubiquitous computing can be more effective if peer-to-peer approach is followed instead of client-server. Thus, considering the benefits of peer-to-peer model, we present the concept of *UI delegation*. Furthermore, the UI-related data, which are exchanged between the peers, shall be based on a protocol elaborated on a common interface language (CIL).

In the context of the present work, the *UI delegation*:

- insures autonomy of peers to render UI according to its own capabilities, having also into consideration the user capabilities listed in his/her profile;
- takes into account the heterogeneity of interaction modalities, including *i*-HCI;
- includes a protocol that facilitates collaboration as in the peer-to-peer model;
- contributes to the usability of the system in the sense that it provides a comprehensive understanding of the usability concerns related to human, system, and platform views;
- advocates decentralization to attain autonomy, and intends to resolve scalability and maintainability issues that may prevail as a result of centralization.

Our approach is different from other works in three ways. Firstly, it attempts to simultaneously take onboard the *system*, *device* and *user* aspects in the process of generating UI at run time, instead of relying only on one of those aspects. Secondly, it is proposed in the context of a peer-to-peer model, where multicasting is used and collaboration is maintained using a CIL protocol between peers. Thus, it is possible to achieve autonomy of peers and resolve scalability and maintainability issues. Finally, it is different since our concept uses the *system* view to include various interaction modalities, as well as *i*-HCI, instead of limiting the system view to describe software capabilities and constraints.

The rest of this paper is structured as follows. In Section 2, we discuss works related with cross-platform UI development, as well as UI distribution and UI migration. In Section 3, we discuss the concept of *UI delegation*, including the requirements deemed to satisfy this concept. Finally, in

 ISBN: 978-1-61208-382-7

Section 4, we draw relevant conclusions about the concept of UI delegation, discussing what more should be done to materialize it across ubiquitous environments.

II. LITERATURE REVIEW

Model-based UI development (MBUID) is one of the principal approaches that strive in developing UIs that can run on multiple or across heterogeneous platforms. In MBUID, users, data, tasks and functions can be modeled in order to turn them into interaction concepts [11] [17] [18]. The models are then used to guide the UI development, as well as to automatically generate the end UI [10] [11].The Cameleon Reference Framework abstracts models to describe the UI at different levels of abstraction, namely: abstract UI, concrete UI, and final UI [13]. The abstraction in a model signals the list of candidate widgets for the interaction. For example the "choice" concept can be an abstraction of combo box, list box, check box, and radio group [16].

Vanderdonckt [19] classifies the UI design for heterogeneous platforms as per the situation that causes the diversity. Therefore, the UI design may focus on the presence of multiple users or, alternatively, on the usage of multiple monitors, devices, platforms, and displays [20]. In this regard, the UI distribution and migration are followed as general UI development approaches [4] [7] [21] [22] [8]. UI distribution is the concept of spreading UI components "across one or more of the dimensions of input, output, platform, space, and time" [21].

In [8], distributed and migratory UIs are discussed as two independent concepts. Migratory UIs can be in the form of distributed UIs, but they shall enable the user to continue the interaction without losing the state (content) of the UI [7] [8]. In UI distribution, UI elements are distributed across platforms, and, in some cases, this may create duplication of UI elements [8]. For example, in [12], a multi-client (multi-platform) UI is presented using the model-view-controller (MVC) architecture that stores different versions of a webpage (UI) on the server for each predefined platform, and where the controller selects one of the UI versions that most fits a particular platform. But, this approach is prone to maintainability and scalability issues. For example, if a UI element has to be modified or added, such an operation has to be done for each version of the respective UI.

Elmqvist [21] pointed out in that distributed UIs can have multi-device environment and/or interaction modalities aspects, including application and content redirection in addition to UI migration. In due case, usability is a concern when adapting the application interface to another device with different capabilities and constraints. The notion of plasticity of UI is thus presented as another concept to refer the ability of UI to withstand variations across platforms, while preserving its usability [23].

If usability is a concern, then both the platform and user capabilities have to be addressed while generating an UI [6] [9] to be distributed or migrated. However, most works put a focus on the capabilities of one sort of participating entities, mostly the user or platform. For example, Nichols et al. described the interface and function of appliances using a UI description language, which is applied to create "specifications for 33 appliances, including several with more than 100 functional elements" [11]. Thus, only the platform (appliance and application) capabilities are the main consideration for generating the UI. Also, Lei et al. considered that the device context is to adapt UIs across devices with various screen sizes [10]. MARIA [14] was also proposed as a description language to support migratory UIs and to design and develop multi-device UIs by using Web services following the form of *e*-HCI. But, it has to be noted that UIs are regarded as a means of communication between the user and the computing environment, and this should include invisible UIs (or *i*-HCI), in which interaction ideally takes place with no perceived mediation, and in a more real-world interaction style [15]; and such modality is the one that most fits the notion of ubiquitous computing. Some works [24] [25]have attempted to address the personalization of users by automatically generating interfaces that are customized to an individual user profile.

Paternò et al. [8] pointed out that in UI distribution, at least two devices are involved when rendering UI. Despite the main focus could be elsewhere (i.e., user, task, environment, modality, etc.), it is vital to consider the device context in any case. Therefore, we formally consider both the platform (device and system) and user capabilities in our conceptual approach.

Four concerns are discussed in [11][21] focusing on the multiplicity of displays, platforms, operating systems, and users, before proposing their toolkit developed in the peer-to-peer model. The user aspect in [20] considers distributing UI for multiple users, but not on the heterogeneity of users. This differs from our concept of user view, since we consider the user capabilities and constraints as part of the user view, in addition to the number of users to which the UI is migrated. Similarly, Elmqvist [26] introduced a peer-to-peer middleware, which is "tailored for high-performance visualization" [26] within an environment with diversified display sizes, such as tabletop display, wall-mounted display, and mobile devices, but without the *user* view.

In [6], during the usability engineering lifecycle of UI development for interactive system, the *user profile* is mainly about characterizing the user, not only naturally (physically), but also with respect to the cognition model and the psychological makeup of the user. It is important to note that supported human capabilities (what we are proposing to be included in generic terms) is different from the user profile studies that focus on classifying the behavior of specific user groups by culture, experience, knowledge, etc. Such constraints are contextual, but our concept lays on the physical and technical capabilities and constraints of users coexisting in the same environment, or those participating in the UI distribution or migration. For example, if an UI is shared between users who have visual impairment and those who have not, then the process of UI delegation shall consider this situation by creating the UI on the *delegatee* side as per the user profile, which can be different from the user profile on the *delegator* side.

After considering the various solutions, concepts, and theories in the literature, we noted the following gaps:

- The works we found in the literature focus on *e*-HCI and solutions that are most related with interactive systems; however, ubiquitous computing can be smart as well. Thus, it requires *i*-HCI [5].
- Distributed and migratory UIs are generated by considering a particular view, mostly the device capabilities (e.g., screen size), but rarely the user and interaction modalities. Furthermore, the migration or distribution is often between similar modalities [7]. Thus, a generic approach that focuses on the merging of *user*, *device* and *system* is important, so as to support heterogeneity of users and the interaction modalities.
- In the general setting of ubiquitous environments, with which computing and interaction with heterogeneous platforms is carried out on the fly, scrutinizing the platform capabilities and constraints would be endless and impractical due to the numerous options of interaction modalities and technologies, not to mention those that are expected to emerge in the future. Hence, scalability should be one of the prominent considerations to be taken in the new concept of UI delegation.

III. USER INTERFACE DELEGATION

The rationale for the emergence of the UI distribution and migration concepts result from the need for enabling users to continue performing their tasks on the go and pervasively. We consider the concept of *UI delegation* as the third dimension (in addition to UI distribution and UI migration) that sustains the development of cross-platform interfaces, so that interaction can be extended and usability can be improved by sharing capabilities of *delegatee* platform. For example, a **list box** widget can be used "*on behalf of*" **radio button** for implementing "*choice*" concept in the interaction. Similarly, instead of visually reading a text from the screen, it can be converted to audio and played if the capability exists. Thus, **audio listening** can be used "*on behalf of*" of **visual reading** across platforms, so that trans-modality can be achieved after all.

The notion of "*on behalf of*" is driven by the cooperation and collaboration between the *delegator* (i.e., the one that requires the UI to be rendered on a remote platform) and *delegatee* (i.e., the one that renders a UI on behalf of other peer), and this should happen when the *delegator* desires to perform the task but knows there is a better capability on the *delegatee* side. For example, while composing a message the user can type using a keyboard on desktop/laptop more easily and efficiently than using keypad of a smartphone. On the other hand, smartphone may possess the connection and SMS service. Therefore, the notion of "*on behalf of*" exists if the desktop/laptop is delegated only for the purpose of delivering the input modality as per its capability. Considering this example, the idea could be similar to the concept of UI granularity (or, in our case, granularity of the interaction modality) that is manipulated during distribution or migration of the UI (or part of it) as discussed in [27]. However, in *UI delegation,* the UI element is not distributed but created at runtime as per the capability of the platform that renders the UI element – the *delegatee*.

UI delegation, in addition to supporting cross-platform UIs as distribution and migration approaches do, it is also useful to create a merger of capabilities in a certain computing environment. As discussed above, the heterogeneity may occur as the result of the diversity of capabilities owned by *systems* (application and interaction modalities), *devices*, and *users*. The merger of the capabilities can be thus used to extend the *capability domain.*

$$C_1 = \{a, b, c, d\}$$
$$C_2 = \{a, b, x, y\}$$
Therefore
$$C_d = C_1 \cup C_2$$

Figure 2. Representation of capabilities domain

For instance, as shown in Figure 2, let C_1 and C_2 be sets of capabilities (interaction modalities) of platforms P_1 and P_2, respectively, (i.e., those coexisting in the same computing environment). Then, C_d is the *capability domain* we can benefit from using such computing environment. It is apparent that both platforms have shared common capabilities $\{a, b\}$, but each of them also has exclusive capabilities, $\{c, d\}$ for P_1 and $\{x, y\}$ for P_2. Therefore, UI delegation can be applied to enable one platform to use one or more capabilities of another platform. In due process, if the modalities between the two platforms are the same, then it is said that we have mono-modality; otherwise it is trans-modality. Thus, if P_1delegates P_2, then P_2 is running the desired interaction modality *"on behalf of"* the *delegator*. We call P_1 the *delegator* and P_2 the *delegatee*.

A. Theory of Delegation

Castelfranchi et al. relate delegation to agents since it is related with the notion of "*task*" of "*on behalf*" in addition to the need of autonomy and collaboration [15]. Thus, "*task*" of "*on behalf*", autonomy, and collaboration are the three prominent reasons leading to the theory of delegation [15]. Similarly, we use the theory of delegation, but applied to the cross-platform context of HCI.

The notion of "*task*" of "*on behalf*" is discussed above. In HCI, the autonomy is satisfied by letting platforms (peers) to run delegated UI (i.e., the complete or partial version of the UI) in their own capabilities, instead of generating the UI from server side. Finally, the collaboration is met using the communication protocol between peers.

In the literature, we found that the theory of delegation is well presented in works related to agent-based systems. Haddadi develops the theory by taking "*an internal perspective to model how individual agents may reason about their actions*" [26]. This is further developed in [15], where it is stated that "*in delegation an agent A needs or likes an action of another agent B and includes it in its own plan, thus, A is trying to achieve some of its goals through*

B's action". According to Castelfranchi et al., *A* is said to be the "client", while *B* is the contractor [15]. In our context, *A* decides to whom to delegate as an autonomous peer, although *B* may reasonably agree or not to be delegated. We call to *A* and *B* the "*delegator*" and "*delegatee*", respectively.

In spite of not being defined as a theory in [27], the concept of delegation is used with the intent of compensating the low computational performance of small handheld devices by delegating in high performing computers the execution of tasks requiring higher performance computation.

We draw the theory to support cross-platform UIs within peer-to-peer model, such that interaction modalities include the practice of *i*-HCI, as well as the notion of invisible UI. Furthermore, the capabilities used to generate a UI component shall be defined from the *human, device* and *system* views. Thus, in UI delegation, we have the following:

- a peer (*delegator*) shall demand a capability of another peer in the same computing environment;
- all peers are responsible to register and maintain their own capabilities locally, and advertise them when required;
- a peer looking for a capability shall advertise it, and only peers that own such a capability shall respond;
- a *delegator* is in control only before transferring the UI-related information to the *delegatee*; and
- a *delegatee* is in control only while delivering the UI, loosing such control when the UI state is changed as a result of interaction.

In order to maintain the collaboration between peers, and to standardize how capabilities are represented, the peers shall use the CIL that serves as a protocol between peers.

B. The Protocol (CIL)

The UI delegation concept we propose in this paper is meant to fit in a peer-to-peer model that requires decentralizing UI-related information, as well an enabling peer that is intended to serve as *delegatee*. Hence, as in [15], where the agent has to select the task to be run for another agent, the *delegatee* has to invoke some of its own capabilities (i.e., locally stored) that are adequate to deliver the required UI (or part of it) on behalf of the *delegator*. To achieve this requirement, we propose a set of rules governing the UI-related information exchange between peers. In addition, how each peer registers its capabilities locally has to be standardized. Therefore, CIL is conceptualized as the protocol that serves these needs.

In order to apply CIL as a protocol between the peers, it shall play three basic functions as: *syntax and semantics*, *description language*, and *communication rules*.

1) Syntax and semantics

The peers taking advantage of the concept of *UI delegation* shall use a standardized and common way of describing the UI-related information. This includes standardizing the syntax and semantics of the language to be used between peers. Also, it requires a decision about which aspect of UI-related information to be represented using the protocol. As discussed above, one of our main goals is to consider and use the merger of the *system, device*, and *user* contexts during the cross-platform UI development, provided that are deemed important for the UI generation (i.e., at runtime or design time). Therefore, at this stage of our work, we consider the *human*, *system*, and *device* as views to be integrated in the *CIL-definition*. The human and device views can be taken into account for identifying physical and technical capabilities supported and available on each peer. Hence, a *delegator* can use the information to select the *delegatee* that optimally meets the desired capabilities.

On the other hand, the system view is required to define available capabilities related to interaction modalities, available support for *i*-HCI, tools useful to support conversion between modalities (e.g., text-to-speech), and so forth. In particular, the system view covers three broad aspects of the interaction:

- *How interaction is presented:* the presentation of UI can be in the form that the user shall react to (e.g., web form), or implicitly (e.g., ambient display) in which users can be passive in respect to the presentation.
- *How interaction is triggered:* interaction can be triggered as a result of the occurrence of a specific event, command, periodic instance, etc.
- *The type of modality-state:* peer might have the capability to use mono-modality or trans-modality. In trans-modality, peers are capable of converting one modality into a different type of modality (e.g., text-to-speech)

As shown in Figure 3, the *CIL-definition* is the foundation on which UI related-information and messaging are described during the process of *UI delegation*. The *CIL-definition* constructs the syntax and semantics of the CIL in general, which has to be followed by each peer. Also, since the definition can be improved from time to time, it has to be associated with version identifier.

2) Description protocol

Once each peer knows how and what to specify, the protocol can be used to describe the capabilities of each peer, as well as the presentational information of the current UI desired to run on the *delegatee* side.

Each peer shall describe locally the capabilities it supports in accordance to the *CIL-definition* with the *human, device*, and *system* views. Thus, when the *delegator* decides to delegate a peer, the selected peer (*delegatee*) shall present the UI in its own capability as described locally.

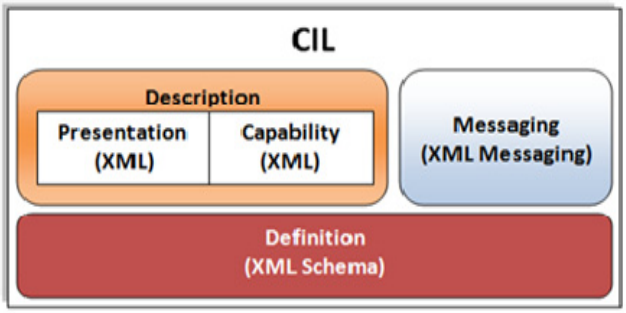

Figure 3. Structure of CIL

Description is also required to communicate and use presentational UI information (i.e., the structure of UI element, the entire UI or interaction modality) useful to create the UI presentation on the *delegatee* side.

The presentation is created using the *CIL-definition* in two stages. In the first stage, the *delegator* has to create the CIL version of the UI intended to run on the *delegatee* side. Then in the second stage, the *delegatee* shall map the presentation (*CIL-description*) in accordance to its capabilities, and generate the new presentation of the delegated UI. More discussion about mapping capabilities is given further ahead.

3) Communication rule

The exchange of UI-related information shall follow a standard that can be understood and interpreted by each peer. Therefore, in addition to creating the *CIL-definition* as rule upon which the *CIL-description* of capabilities and presentation is made, we found it valid to consider a third role within CIL through which peers collaborate: *CIL-Messaging. CIL-messaging* is the third role that must be played in three situations:

- when the *delegator* sends a *delegation request* to peers;
- when peers respond to a *delegation request;*
- after the *delegator* selects one of the peers as *delegatee* and, when the described UI or interaction modality is transmitted; and
- when the *delegatee* decides to transmit the UI with its new state back to the *delegator*.

C. UI Delegation Process

The delegation process can be started on-demand or automatically, and there are five important requirements to be fulfilled:

- Describing capabilities
- Creating *delegation request*
- Responding to a *delegation request*
- Selecting and appointing *delegatee* and
- Mapping UI/interaction modality

The *CIL-description* and/or *CIL-messaging* are used to fulfill each of these requirements, while the *CIL-definition* serves as a standard for maintaining consistency and interoperability across peers in the process of messaging, as well as to describe the capabilities and presentation.

1) Describing Capabilities

Local capabilities of each peer can be described by the UI designer, and verified as per the *CIL-definition* (the XML schema).

2) Creating Delegation Request

Delegation request has to be created first by translating the current UI deemed to be delegated into *CIL-description*. During *delegation request*, the *CIL-description* is in a more abstract form only to depict the desired capabilities from user, device, and system points of view.

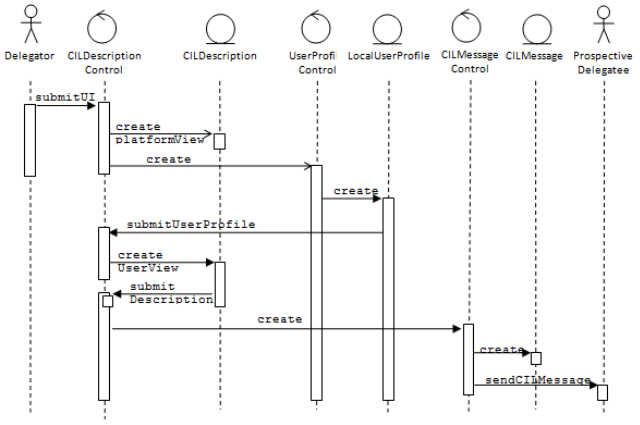

Figure 4. *Delegation request*

In addition, the required user profile shall be used to create the human-being view description. Once the description is done, it is used to define the *delegation request* message using the CIL-messaging format, being then the message sent to the prospective CIL-enabled peer, as shown in Figure 4.

3) Responding to delegation request

Each peer receiving the *delegation request* shall compute the degree-of-matching between the requested capability coming in the CIL message and its own capabilities. The degree-of-matching *M* can be computed using the algorithm (pseudo code) shown in Figure 5. Basically, each element *e* in the *CIL-message* is searched within the local list of capabilities of the prospective *delegatee*. Three situations may occur:

- if *e* is *identical* to a capability of the delegatee (first condition), the prospective *delegatee* is probably similar to the *delegator*. Hence, the value of *M* is incremented by two;
- if *e* is *found* to be similar to one of the capabilities of the delegatee (second condition), the prospective *delegatee* has similar capability but may not be in the same way as in the *delegator* (e.g., browser of different type). Hence, the value of *M* is incremented by one;
- if *e* is not found at all, the value of *M* is decremented by one.

```
Line 1  | Set M →0, i→1
Line 2  | Read Local_capability []
Line 2  | for each element e in CIL_message
Line 2  |     Search for e in Local_capability
Line 3  |     If identical found
Line 4  |         M:=M+2
Line 5  |     Else if found
Line 6  |         M:=M+1
Line 7  |     Else
Line 8  |         M:=M-1
```

Figure 5. Simplified algorithm for calculating degree of matching (*M*).

4) Selecting and appointing delegatee

Once the value for degree-of-matching *M* is received from each prospective *delegatee*, the *delegator* will select the *delegatee* that responds with the largest *M* value. In due

process, the presentational *CIL-description* will be sent to the selected *delegatee*.

5) Mapping UI/interaction modality

Once a peer is appointed and receives the presentational *CIL-description*, the *delegatee* will replace the *CIL-description* with its local capability. For example, the description of an HTML tags for *single line text* input tag of a web interface in Figure 6 (a) can be mapped to Figure 6 (b) that of an a multiline *text box* or vice versa. In the mapping process, either part of the description is depicted to fit the local capability, or it could be expanded. However, the major structure descriptor and the state of the widget are maintained as-is (see the bold and underlined part).

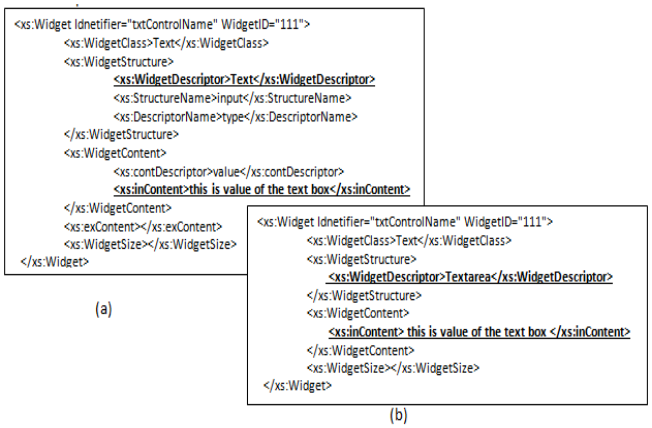

Figure 6. Description of UI concept using different capability

Nevertheless, UI elements that should not change the original structure should not pass through the delegation process. For example, some text inputs (e.g., username) might be required to be just one line.

Therefore, to correctly perform the mapping, a standard has to be followed between the *delegator* and the *delegatee* on how to describe the widgets (or other UI-related information). Hence, the CIL plays important role during the mapping.

IV. CONCLUSION

UI distribution and UI migration are two prominent dimensions that are useful to support the development and usage of cross-platform UIs. These concepts are often applicable if a client-server model is followed and it is not desired to have autonomous platforms. Thus, a dedicated server has to be assigned to orchestrate the distribution or migration. Furthermore, a server of this sort requires higher degree of reliability; otherwise, it can be a point of failure that jeopardizes the entire cross-platform operation. In addition, most works following these approaches focus primarily on a specific context (device, user, system, task, etc.). However, the use of *human*, *system*, and *device* views should be apparent, and shall not be split, so that the usability of the system can be improved even in cross-platform UI development.

Therefore, we draw from the theory of delegation – which is most applicable in agent-based system – the new concept of *UI delegation* as the third dimension in cross-platform UI development. In our work, we propose the UI delegation concept to follow peer-to-peer approach, so as to assure that peers remain autonomous. In due process, protocol for UI-related information exchange is important. Accordingly, we have discussed the notion of CIL together with the process of *UI delegation.*

Therefore, we claimed that if each peer is able to describe and maintain its capabilities and constraints, then new peers can be added easily. In due process, we consider *i*-HCI as interaction modality, which can be defined by the amalgamation of contextual information and intelligent technology. Thus, it is one dimension to be satisfied by the use of *CIL-messaging*, as per the *CIL-description*, which is built from the human, device and system points of view. Furthermore, considering delegation as per the capabilities of the *delegatee* peer would help to define autonomous peers, which are limited by the capabilities and constraints defined at the server.

In order to materialize the concept of *UI delegation,* in the future, more work has to be done to complete standardization of the CIL. It is also important to define a framework for CIL-enabled peers. In due course, though the computational power of small handheld devices is higher than ever, in the future, it is important to address the performance aspect of the delegation process since *delegatee* peers are responsible for mapping the UI description into their context.

REFERENCES

[1] K. Byeong-Ho, "Ubiquitous computing environment threats and defensive measures," International Journal of Multimedia and Ubiquitous Engineering, vol. 2, no. 1, 2007, pp. 47-60.

[2] A. Greenfield, Everyware: The Dawning Age of Ubiquitous Computing. Berkeley, USA: New Riders Publishing, 2006.

[3] M. Weiser, "The computer for the 21st century.," Scientific American, vol. 263, no. 3, 1991, pp. 94-104.

[4] H. Sørensen, D. Raptis, J. Kjeldskov, and M. B. Skov, "The 4C framework: principles of interaction in digital ecosystems," ACM International Conference on Pervasive and Ubiquitous Computing (UbiComp 2014), ACM Press, 2014, pp. 87-97, doi: 10.1145/2632048.2636089.

[5] S. Posland, Ubiquitous Computing: Smart Device, Environment, and Interactions. Chicester, UK: John Wiley& Sons Ltd, 2009.

[6] D. J. Mayhew, The Usability Engineering Lifecycle: a Practitioner's Handbook for User Interface Design. San Francisco, USA: Morgan Kaufmann Publishers, 1999.

[7] S. Berti, F. Paternò, and C. Santoro, "A taxonomy for migratory user interfaces," Interactive Systems. Design, Specifcation, and Verification Lecure Notes in Computer Science, Springer, 2006, pp. 149-160.

[8] F. Paternò and C. Santoro, "A logical framework for multi-device user interfaces," The 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2012), ACM, 2012, pp.45-50, doi:10.1145/2305484.2305494.

[9] J. Nielsen, Usability Engineering. Cambridge, MA, USA: Academic Press, 1993.

 ISBN: 978-1-61208-382-7

[10] Z. Lei, G. Bin, and L. Shijun, “Pattern based user interface generation in pervasive computing,”The 3rd International Conference on Pervasive Computing and Applications (ICPCA 2008), IEEE Press, vol. 1, 2008, pp. 48-53, doi: 10.1109/ICPCA.2008.4783636.

[11] J. Nichols and B. A. Myers, "Creating a lightweight user interface description language: an overview and analysis of the Personal Universal Controller project," ACM Transactions on Computer-Human Interaction (TOCHI), vol. 16, no. 4, 2009, article 17.

[12] P. Sauter, G. Vogler, G. Specht, and T. Flor, "A Model–View–Controller extension for pervasive multi-client user interfaces," Personal and Ubiquitous Computing, vol. 9, no. 2, 2005, pp. 100-107.

[13] G. Calvary et al., "The CAMELEON reference framework. Deliverable 1.1," 2002.

[14] F. Paternò, C. Santoro, and L. D. Spano, "MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments," ACM Transactions on Computer-Human Interaction (TOCHI), vol. 16, no. 4, 2009, article 19.

[15] K. P. Fishkin, T. P. Moran, and B. L. Harrison, "Embodied user interfaces: towards invisible user interfaces," The IFIP TC2/TC13 WG2.7/WG13.4 7th Working Conference on Engineering for Human-Computer Interaction, IFIP, vol. 22, 1999, pp. 1-18, ISBN:0-412-83520-7.

[16] R. Castelfranchi and C. Falcone, "Towards a theory of delegation for agent-based systems," Robotics and Autonomous Systems, vol. 24, no. 3, 1998, pp. 141-157.

[17] E. G. Nilsson, J. Floch, S. Hallsteinsen, and E. Stav, “Model-based user interface adaptation,” Computer and Graphics, vol. 30, no. 5, 2006, pp. 692-701.

[18] M. Welie, “Task-based user interface design,” SIKS Dissertation Series No. 2001-6, Dutch Graduate School for Information and Knowledge Systems, Vrije Universiteit, 2001.

[19] J. Vanderdonckt, "Distributed user interfaces: how to distribute user interfaces elements across users, platforms and Environments," The 11th Congreso Internacional de Interacción Persona–Ordenador (Interacción 2010), Universidad Politécnica de Valencia, Valencia, Spain, Sept. 2010, pp. 3-14.

[20] J. Melchior, D. Grolaux, J. Vanderdonckt, and P. Roy, "A toolkit for peer-to-peer distributed user interfaces: concepts, implementaiton and applications," The 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2009), ACM Press, 2009, pp. 69-78.

[21] N. Elmqvist, "Distributed user interfaces: state of the art," J. A. Gallud, R. Tesoriero, V. M.R. Penichet (eds.), Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem, Human-Computer Interaction Series. London, UK: Springer-Verlag, 2011, pp. 1-12.

[22] L. Frosini and F. Paternò, "User interface distribution in multi-device and multi-user environments with dynamically migrating engines," The 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2009), ACM Press, 2009, pp. 55-64.

[23] D. Thevenin and J. Coutaz, "Plasticity of user interfaces: framework and research agenda," M. A. Sasse and C. Johnsson (eds.), The 7thIFIP Conference on Human-Computer Interaction (INTERACT 1999). IOS Press, 1999, pp. 110–117, ISBN: 09673355074274903087.

[24] K. Gajos and D. S. Weld, "SUPPLE: automatically generating user interfaces.," The 9th International Conference on Intelligent User Interfaces (IUI 2004), Funchal, Madeira, Portugal. ACM Press, Jan. 2004, pp. 93-100.

[25] D. Grolaux, P. Roy, and J. Vanderdonckt, "Migratable user interfaces: beyond migratory interfaces," The 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004).Boston, Massachusetts, USA: IEEE Press, 2004, pp. 422-430, doi: 10.1109/MOBIQ.2004.1331749.

[26] S. K. Badam, E. Fisher, and N. Elmqvist, "Munin: apeer-to-peer middleware for ubiquitous analytics and visualization spaces," IEEE Transactions on Visualization and Computer Graphics, vol. 21, no. 2, 2015, pp. 215-228.

[27] A. Haddadi, "Communication and cooperation in agent systems: a pragmatic theory," Lecture Notes in Computer Science, vol. 1056, 1996.

[28] A. Q. Pham, "Privilege delegation and revocation for distributed pervasive computing environments," G. Abraham and B. I. P. Rubinstein (eds.), The 2nd Australian Undergraduate Students’ Computing Conference, 2004, pp. 136-141, ISBN: 0-975-71730-8.

 ISBN: 978-1-61208-382-7