Displays 39 (2015) 42–54

Contents lists available at ScienceDirect

Displays

journal homepage: www.elsevier.com/locate/displa

Design and evolution of web-based screen management middleware for interactive multipurpose public displays

Tommi Heikkinen *, Timo Ojala

University of Oulu, Department of Computer Science and Engineering, P.O. Box 4500, FI-90014 University of Oulu, Finland

ARTICLE INFO

Article history:
Received 20 March 2015
Received in revised form 14 July 2015
Accepted 4 August 2015
Available online 6 August 2015

Keywords:
Screen real estate management
User interface toolkits
Middleware

ABSTRACT

We present the iterative design and evolution of a web-based screen management middleware developed for a network of interactive multipurpose public displays deployed in a city center since 2009. The middleware provides three principal user interface abstractions to content producers, application developers and display owners: the classical passive digital signage, a portal of interactive applications and a finite state machine based interaction model description. With these abstractions passive and interactive content can be spatiotemporally multiplexed onto the screen according to the given interaction model. We reflect on the evolution of the application development process and the real world usage of the middleware. We conclude with a discussion on our impressions on the web as an application development platform and the presented screen management middleware for interactive multipurpose public displays.

1. Introduction

The prices of large digital screens have been constantly decreasing which has allowed public displays to conquer city centers, highways, airport terminals and commercial centers [1]. The primary use of these displays has been advertising and information distribution, which are commonly referred to as digital signage. These displays typically show a relatively static playlist of video and images and are completely audience agnostic without any interaction capabilities. These displays are expected to become increasingly interactive as the quality and affordability of interaction technologies improve. As the investment for interactive technologies becomes feasible, the question is how to make the best use of them.

The act of using large displays is qualitatively different from personal computers [2]. In addition to different form factors, large displays are typically situated in particular context and shared between users. One distinction is whether a display is interactive or not. Interactive displays have to support interaction and the underlying interaction model, which dictates how inputs are mapped to outputs. Another distinction can be made on the displays' purpose, i.e. whether it provides a single application or multiple applications. Interactive multipurpose public displays (later MPDs) [3] differ greatly from non-interactive displays and single-purpose displays what comes to managing content and applications on the displays. Designing the screen management application programming interface (later API) defining the basic application container abstractions and controlling the user interface elements, i.e. controlling the screen real estate, is likely more challenging for multipurpose displays whereas many single purpose display systems do not have any screen management functionality at all. Davies et al. [4] provide a comprehensive review of current public display literature. They argue that the mainstream of public displays is dominated by four different applications: advertising, information boards, signage, and art and entertainment. These displays typically enhance the traditional static signage systems and are designed solely for one particular purpose. Making displays multipurpose can serve many goals. Obviously, there is the research motivation to allow different use cases to emerge freely. MPDs can also yield to a better cost-benefit ratio as a display can serve multiple purposes, which is especially beneficial in expensive installations. MPDs also allow supporting the needs of a more diverse group of stakeholders.

In contrast to popular window systems in personal computers, the designers of MPDs should not rely on users to manage the user interface elements themselves. MPDs located in public places can be used by anyone, and therefore they need to be designed for first-time use [5]. While some people may become experienced users of MPDs over time, it does not negate the fact that even years

Abbreviations: MPD, multipurpose public display.

* Corresponding author.
E-mail addresses: tommi.heikkinen@ee.oulu.fi (T. Heikkinen), timo.ojala@ee.oulu.fi (T. Ojala).

http://dx.doi.org/10.1016/j.displa.2015.08.003
0141-9382/

after the introduction of the MPDs completely new people will find the displays and use them for the first time. Therefore, we should not assume potential users to have a prior knowledge of the content and interaction capabilities of MPDs. This is particularly true for the walk up and use type of displays [6], a typical category of MPDs, which invite spontaneous and short lived interactions [3]. Combined with the often too coarse input technologies of large displays, such interactions call for system-driven layout management instead of user-driven layout management.

We contribute to the realization of MPDs by presenting the incremental design and evolution of the web-based screen management middleware for the UBI-hotspots, a network of interactive multipurpose public displays deployed around Oulu, Finland, since 2009. The middleware engages temporal and spatial multiplexing of content and a stage-based interaction model to realize a hybrid of scheduler based and interaction model based screen management. The middleware provides three principal user interface abstractions to content producers, application developers and display owners: a classical passive digital signage application dubbed the UBI-channel, a portal of interactive applications dubbed the UBI-portal and a finite state machine based description language for encoding interaction models with layout information. With these abstractions, passive and interactive contents are spatiotemporally multiplexed on the screen according to the given interaction model. We have previously published the three incremental versions of the screen management middleware in [7–9]. The novel contribution of this paper is the evolution of the middleware and application development process, together with a reflection on the real world usage of the middleware, and a thorough discussion on our impressions on the web as an application development platform and the presented screen management middleware for interactive MPDs.

2. Related work

2.1. Content multiplexing

The screen real estate is the principal output mechanism of public displays and therefore its efficient utilization is desirable. MPDs may require multiplexing multiple content items on a single screen, which can be done in different ways: temporally, spatially, by stacking, by frequency or by code [10]. In temporal multiplexing, the content is updated as the function of time. In spatial multiplexing the screen real estate is divided into smaller regions which can be shown side-by-side simultaneously. In stack order multiplexing, content is allowed to overlap, i.e. stacked on top of each other. Semi-transparency is a specific method of stacking that allows viewing multiple overlapping stacked layers simultaneously [11]. Alt et al. [12] have proposed an application level alternative for stacking called ‘integrated’, where an ad is carefully integrated into an application, e.g. as a logo shown on a map. In frequency multiplexing different frequencies can be dedicated for visualizing different content that may overlap spatiotemporally, but still allow interpretation of individual content by focusing on certain color [10]. Finally, in code multiplexing some dimension of the screen real estate is multiplexed based on some code, e.g. QR-code [10]. Different multiplexing methods can also be coupled for enhanced flexibility.

2.2. Screen real estate management

In the literature, two main approaches to the screen real estate management of multipurpose public displays can be identified: scheduler based models and interaction driven models. Further, another important aspect of a display system is the type of interaction: passive (no interaction), explicit, implicit, or both explicit and implicit. Fig. 1 shows a classification of selected multipurpose display systems and studies along these two dimensions. A system or a study enclosed in brackets includes multiple “hard-wired” applications, but as such does not provide any abstractions or interfaces for explicitly adding new applications. Single-purpose systems and studies are omitted intentionally.

2.2.1. Scheduler based models

Scheduler based models originate from the digital signage industry. Their principal idea is to use scheduling rules and constraints to define what, when, where and how are shown on the displays. Before becoming digital, signage surfaces were either completely static or content was updated manually. Some surfaces even contained a mechanism for switching the content automatically to allow multiple signage to be shown in succession, i.e., multiplexed temporally. The digitalization of the signage surfaces allowed updating the content any time and also efficient multiplexing without any technical limitations on the amount of content.

Traditionally, content management on public displays follows a playlist model, which is especially popular for advertisement displays that are typically viewer agnostic. In such displays content items are multiplexed temporally by showing them one after another. While the vast majority of such displays have been commercial deployments, some research deployments also exist. The player software architecture in the e-Campus system [13] allows content synchronization on multiple displays and it implements transaction semantics over content items for conflict management. Content was scheduled to the displays using the e-Channel system [14] that provided an abstraction layer between content providers and display owners. The e-Channel has been later replaced with specific decentralized Content Descriptor Set XML files along the Yarely software player [15]. The upgrade was made to achieve better scalability and robustness, and a more open and extensible system by supporting interoperability with third party content producers.

Another example of the scheduling based model are the iDisplays news and reminder displays [16], which also exploited spatial multiplexing by dividing the screen real estate into different zones to fit more content simultaneously. The system supports dynamic selection of content items called information chunks.

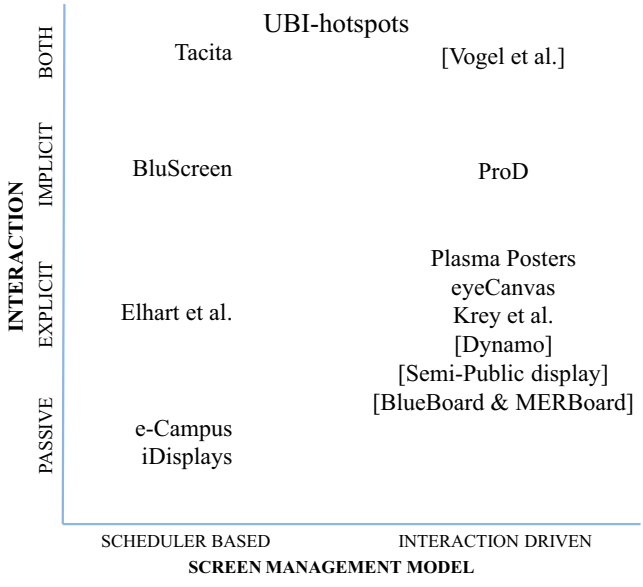

Fig. 1. Classification of multipurpose display systems and studies.

While news displays would simply show all the chunks, reminder displays filter chunks based on context. iDisplays supported an auction mechanism to select the chunks that are most important in a particular context in situations when there are more content available than what could fit on the screen at once.

There is also recent work of using a scheduler based model with explicit touch interactive applications by Elhart et al. [17]. In their system a distributed scheduler operates according to a set of rules and constraints in the presence of spontaneous user requests to manage spatiotemporally the content shown on a display. The rules include but are not limited to priority based, round robin, first come first served and default activity (i.e. triggered when no interaction). The constraints allow setting conditions for e.g. timing, priorities, precedence and ability to interrupt. Web-based applications could multiplex between four different spatial layouts: tickertape (a narrow bar on the bottom), sidebar, main area, and full screen.

Content scheduling can be also based on implicit information on users. BluScreen [18] scanned nearby Bluetooth devices and maintained the history of content items shown in the presence of a particular device. It could use this information and a specific auction mechanism for selecting fresh content to be shown on a display. Davies et al. [19] presented architecture for personalizing a public display hand in hand with scheduler based approach. User requests submitted via the Tacita mobile application could override the default playlist content. Tacita supports three personalization modes: 1. Walk-by personalization similar to BluScreen, 2. Longitudinal personalization similar to iDisplays, and 3. Active personalization through a mobile phone. The resulting system also supports the spatial multiplexing of screen real estate between multiple concurrent users.

2.2.2. Interaction driven models

The interaction driven models place the user in control within the constraints of an interaction model of some kind. This is highly affected by the nature of the interaction type. For touch enabled displays, interaction is explicit and occurs at arm's length. The BlueBoard and MERBoard displays [20] focused on fast single user interaction including quick checking of a personal calendar and shoulder-to-shoulder group use, as their 1.3 meters wide screen allowed collaboration of multiple people. No explicit system support was given for group use and users were expected to work collaboratively by taking turns. Screen real estate management was simple: one application at a time could be activated via the touch screen.

The Plasma Poster [21] displays used also touch interaction and the interaction model was designed to allow peripheral awareness and active reading. When not explicitly used, the displays would repeat a carousel of digital posters, each shown for half a minute. Users were able to pause the playlist anytime and freely browse the posters and open links via the touch interface. When the user left the display, the display would go back to the carousel mode. The Plasma Poster displays used also content stacking to orchestrate two distinct browser implementations: a completely web based leveraging the flexibility of web and a customized version that allowed better control over link behavior and cross domain support. To exploit their respective benefits, runtime stacking was implemented to swap between the two browsers, depending on which functionality was needed. The eyeCanvas [22] display deployed in a cafeteria in San Francisco was a derivative of Plasma Poster, thus having the same portrait orientation and software platform. The display showed café related content and visiting artists' works. It also had a mechanism to sing-up for receiving newsletter from the cafeteria and a 'Scribble' application for leaving messages by the clients.

Stacking dimension has been used also to multiplex content on multi-purpose displays. Kray et al. [11] evaluated different alternatives to multiplex two applications on the Hermes displays, including partial overlapping, full screen semi-transparent overlay and full overlapping. The display owners preferred partial overlapping and semi-transparent overlay as they allowed seeing the default application simultaneously.

Traditional mouse and keyboard inputs are also explicit, but allow longer interaction distance similar to a meeting room with a projector display. For example, Dynamo [6] focused on the group use with mouse and keyboard as interaction devices. Users could 'carve' out a personal space on a shared display and share content from their USB-drives with others. Dynamo displays were also later deployed in a school setting for evaluating communal use among students. The Semi-Public display [23] had four applications fixed spatially so that they would all be visible simultaneously. However, individual applications could temporally multiplex their content, if there were multiple content items to be shown.

Content multiplexing can be also proactively triggered based on implicit viewer information. For instance, ProD [24] provides a framework for creating audience aware proactive display systems based on three-staged content selection model: 1. Presence detection i.e. determining who matters, 2. Selecting content based on rules, and 3. Presentation i.e. layout rendering. The first stage includes detection, prioritization and annotation (e.g. social relations). At the second stage this information is used together with personal preferences to collaboratively select potential content to be shown. At the third stage a pluggable presentation engine takes care of rendering the content.

When both explicit and implicit interaction types are used, interaction model becomes more sophisticated. For example, Vogel and Balakrishnan [25] identified four phases or stages of interaction with a public display: ambient, implicit, subtle and personal. In the ambient phase, the display shows general information focusing on the top of the screen for maximum visibility. In the implicit phase, the display detects users' attention and responds by showing a moving vertical bar that mimics the users' movements to draw them to the next phase of interaction. In the subtle phase, a user is actively looking at the display and sees an overview of personal data. Finally, at the personal phase, the user queries detailed data via touch screen and the information is shown using small font visible only from close distance.

3. Case study: screen management of UBI-hotspots

3.1. Stakeholders and their requirements

The design of the hotspots commenced with a diverse range of interest groups in 2008. The stakeholders were involved via a range of activities ranging from brainstorming workshops to rapid ethnography and storytelling competitions [26]. While the resulting value network is reported in detail in [27], we discuss here briefly the stakeholders and the requirements that were relevant for the design and implementation of the resulting screen management functionality.

The City of Oulu, the place owner of the locations of the upcoming hotspots and the co-financier (11%) of the project, required the hotpots to have digital signage functionality for informing citizens on municipal services and events (requirement R1a), and to offer a range of interactive e-government services to be provisioned by the City's various service divisions (R1b).

The ERDF, the main (89%) financier of the project represented locally by the Council of Oulu Region, required the project to

develop a business model that would allow keeping the hotspots operational for five years after the conclusion of the funding (R2a). Further, the financier encouraged seeking opportunities for involving local industry in the implementation of the hotspots (R2b).

Advertisers requested for conventional digital signage with conventional media formats (R3a), largely due to their earlier exposure to and experience of digital signage. They wanted to buy predetermined and guaranteed visibility to their campaigns within a particular period of time (R3b) and asked for tangible evidence on the number of contacts created (R3c).

Potential users were involved in the design of the hotspots in multiple ways [27] and most participants felt a need for this kind of a public service. Specifically, they preferred direct touch screen interaction over interaction with mobile phones (R4).

Application developers were involved as stakeholders when the hotspot platform was opened for 3rd party applications. Their inputs have been collected in multiple ways including two open international application development challenges [28,29], semester long student course works, intense summer schools, hackathons and interested organizations wanting to deploy applications on the hotspots. In contrast to advertisers and content providers for digital signage, developers required access to input technologies of the hotspots and needed to have as much creative freedom as possible (R5a). They also needed a well-defined application platform (R5b) and a way to deploy and maintain their applications (R5c).

We UBI researchers as display owners, required that the hotspots would serve as a platform for a longitudinal field study on interactive public displays that would offer multiple interactive services provisioned in a distributed fashion by multiple service providers (R6a). UBI researchers also defined requirements for the screen management functionality based on our earlier experience as follows: have the system control the layout rather than the users (R6b), provide an interaction model supporting multiple implicit and explicit input mechanisms (R6c), leverage the large spatial screen capacity by allowing potentially multiple users at different stages to use and fetch information from the screen simultaneously (R6d) and finally allow interaction with the public display also via personal mobile phones using multiple interaction modalities (R6e).

3.2. Hardware and software

Eventually, 18 hotspots were deployed at pivotal indoor and outdoor locations around Oulu since summer 2009 (Fig. 2). A hotspot has a 57″ or 65″ Full HD LCD panel in landscape orientation at adult eye level. A hotspot is equipped with assorted hardware such as capacitive touch screen foil for touch input, RFID reader, two overhead cameras, speaker and microphone, and Bluetooth and Wi-Fi access points for wireless communication. All outdoor hotspots and two indoor hotspots are double-sided with two LCD panels back-to-back.

In terms of software, the host operating system of the hotspots is either Windows Embedded POSready, an XP variant designed for point of sales systems, or a standard Windows 7 professional. Middleware components are mostly located on a locally hosted virtual machine, to achieve OS independency, simplify management and improve robustness. The software meta-platform for application development was selected to be web-based so that the screen is treated as a very large dynamic web page. The web was chosen due to its many benefits: widespread use, ease of distributed application development and deployment, fast progressing technologies, no need for specific client software and platform independence. Many other display systems before and after have chosen web as their technology platform, e.g. [17,20–22,24].

(a)

(b)

Fig. 2. UBI-hotspots: (a) double-sided outdoor display; (b) indoor display.

3.3. Screen management abstractions and their evolution

A simple analysis of the requirements showed that the hotspots had to provide two principal abstractions for content providers and application developers (e.g. City of Oulu and commercial advertisers) for exploiting the screen: conventional digital signage dubbed as UBI-channel (R1a, R3a) and a portal of interactive services browsed via the touch screen foil dubbed as UBI-portal (R1b, R4, R6a) (Fig. 3). As such, UBI-channel requires temporal multiplexing of content. The implementation of the back-end of UBI-channel was outsourced to a local company specialized in digital signage systems, to save the project's resources and to incorporate local industry per the financier's request (R2b). The requirement of guaranteed visibility for a commercial campaign in UBI-channel during a given period of time effectively meant that UBI-channel had to be always visible (R3b). This in turn dictated that UBI-channel and UBI-portal had to co-exist, requiring their respective spatial multiplexing on a single screen. The UBI-portal was our own implementation and special attention was paid to make the application creation, deployment and maintenance straightforward by using the web as the fundamental development platform and providing a short list of requirements that the application had to fulfill to be eligible for inclusion in the application registry and subsequent deployment on the hotspots (R5a, R5b, R5c). For

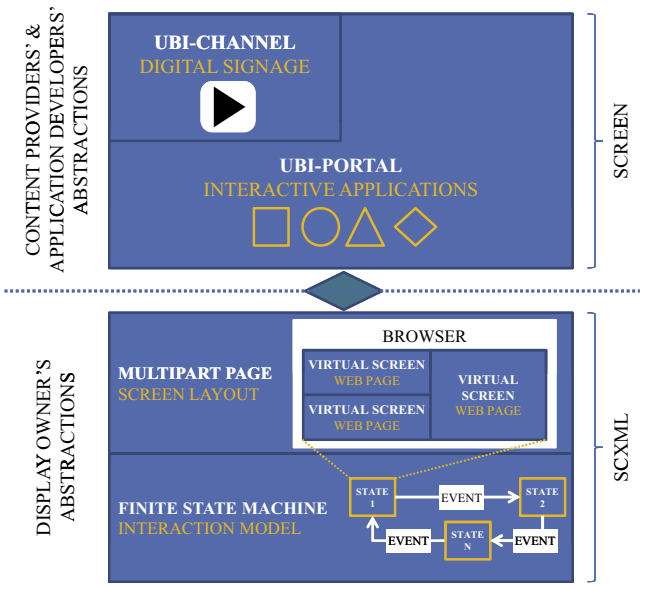

Fig. 3. Abstractions for specifying interaction behavior and provisioning content in UBI-hotspots.

implementing applications involving interaction via mobile phone with the hotspot, the UBI-portal needed to support also applications having distributed user interfaces (R6e).

The overall behavior of the hotspot is governed by an interaction model, which has fundamental implications on the screen management functionality at system level and is typically defined by display owner. As the initial interaction model we adopted a simplified version of the model of Vogel and Balakrishnan [25], where interaction proceeds through pre-defined discrete phases or stages based on information detected from multiple input mechanisms (R6c, R6d). Therefore, we decided to represent the interaction model with a finite state machine controlled by events from input mechanisms. Statecharts have been proposed to overcome the limitations of the popular event-action paradigm used in PC user interfaces [30]. The user interface management in hotspot is recognition-based, which needs to handle the states of users, applications and devices, i.e. their context [31].

In order to place the system in control of the layout, a finite state machine dictates all possible interaction states and their corresponding screen layouts constructed as multipart pages (R6b). The multipart pages consist of an arbitrary number of virtual screens that are the fundamental abstraction for placing content on the screen in form of web pages (R5b). The concept of virtual screens follows the principles of component systems [31], allowing the user interface to be composed by combining multiple separately constructed components.

The evolution of the interaction model and other key activities related to the exploitation of the UBI-hotspots are visualized on a timeline in Fig. 4. The three incremental versions of the screen management system are summarized in the following subsections.

3.3.1. Version 1: layout management framework

Fig. 5 shows the conceptual architecture of version 1 of the screen management system dubbed the Layout Manager [7]. It manages the layout state based on incoming control messages and controls the layout of the screen. The system is completely web based with the user interface running in a browser and the server backend running in a Tomcat web application container. The Layout Manager comprises of two main components, JavaScript partitioning library executing in the browser and Java web application running in the server. The partitioning library manages a dynamic web page, later referred to as the top frame, by manipulating its Document Object Model (DOM) in runtime. The top frame thus manages zero to many virtual screens by adjusting their size and position related to the full screen real

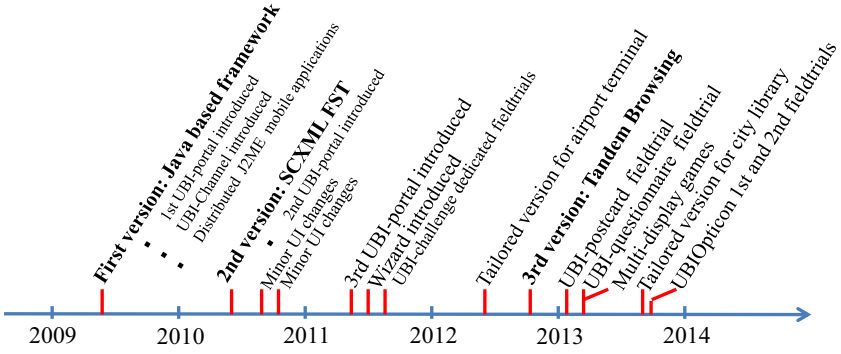

Fig. 4. Timeline of the evolution of screen management functionality.

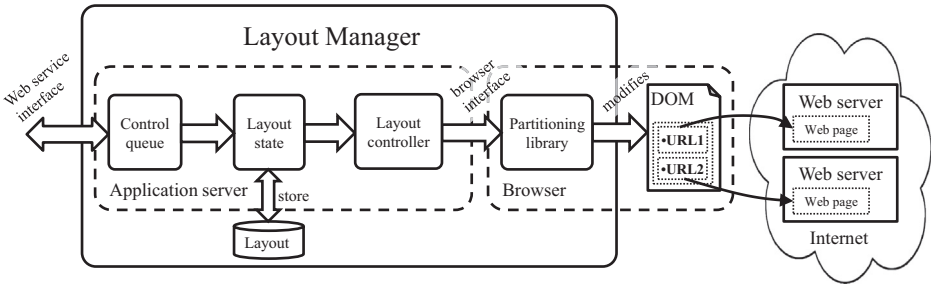

Fig. 5. Conceptual architecture of version 1.

estate. Virtual screens are implemented as iframe elements and they can thus host any web content referred via an URL. Further, the partitioning library implements a set of functions to manage virtual screens spatiotemporally and by stacking. Specific attention is paid on transitions, i.e. if a virtual screen is resized or moved, the partitioning library calculates a delayed animation to create an aesthetically smooth transition.

The Layout Manager keeps the runtime layout information in a Java object model including virtual screens and their parameters. All operations on the layout object model are also echoed to the browser user interface through the partitioning library as well as stored in a database for recovery and debugging purposes. The component also contains a finite state machine rule set written in Java that imperatively defines needed layout modifications in case of state transitions. The state transitions are effectively triggered by the input events constructed from the multiple input technologies of hotspots.

The Layout Manager was taken into use in the first hotspots deployed in May 2009. Example screen layouts are shown in Fig. 6. In the passive broadcast state the whole screen was dedicated to the UBI-channel, similar to the ambient state in the interaction model of Vogel and Balakrishnan. The hotspot transitioned directly into the interactive state in case of three different interaction events: a face oriented toward the display was detected (implicit interaction), or direct input from touch screen or a RFID tag was read by the RFID reader (explicit interaction). In the interactive state the physical screen was split into three virtual screens. UBI-channel was squeezed into the top left hand quarter for maximum visibility for the spectators similarly to Vogel and Balakrishnan and the right hand half was allocated to UBI-portal. The bottom left quadrant was allocated to distributed mobile applications (R6e). To use them the user first had to take over the hotspot by presenting an RFID tag to the RFID reader and then launch applications from the mobile phone with a pre-installed J2ME application [32]. When no mobile application was used, the bottom left quadrant showed a video instructing how to interact with the hotspot. The hotspot transitioned back to the passive broadcast state after a timeout of no interaction events.

3.3.2. Version 2: declarative layout state encoding

Version 2 of the Layout Manager was motivated by increasing demands for modifying the interaction model and for better support of third party application development which were missing from the first version. While the partitioning library did satisfy the new requirements, the Java based finite state machines did not for several reasons. First, any change to the interaction model or layouts required a recompilation of the whole component. Second, the imperative way of describing layout transitions turned out to be a poor approach when the number of transitions increased. Third, writing and especially debugging valid finite state machines was difficult.

To tackle these limitations, XML-based declarative layout state encoding was introduced in version 2 [8]. The finite state machines were implemented as State Chart XML (SCXML) documents which is a W3C working draft. The SCXML is a general purpose event-based finite state machine language based on Harel statecharts [33]. The Harel statecharts provide structural and economical notation for describing reactive behavioral systems by the use of states, transitions, state hierarchies and concurrency. The statechart notation is very suitable for describing causality, concurrency and synchronization of systems, which makes it perfect for building reactive systems such as the hotspots.

For executing SCXML files, the Apache Commons open source SCXML engine was integrated into the Layout Manager. SCXML files can be hosted anywhere in the Internet and they are referenced via URL similarly to the content of virtual screens. All content related descriptions are thus easily accessible and updatable similarly to the common web development process. One particular problem in changing from imperative to declarative descriptions was that they no longer define how transitions between states happen exactly, e.g. would the previous virtual screens be resized first and then the new ones added or would these operations be done simultaneously. Therefore, an algorithm was developed to determine the transition between any two layout states based on the difference on the virtual screens (removed, resized and new). The basic philosophy of the algorithm is to try to minimize the overlapping of virtual screens during transitions. The transition algorithm has been later upgraded to support a display being in multiple parallel states simultaneously, to follow more closely the Harel statechart notation.

Version 2 of the Layout Manager was deployed together with an upgraded interaction model in June 2010. A new subtle state was introduced between the broadcast and interactive states as shown in Fig. 7, again inspired by the model of Vogel and Balakrishnan. Similar to proxemics interactions [34], the purpose was to inform a user in the proximity of a hotspot and looking at the hotspot about its interactivity. The subtle state layout was realized as a semi-transparent overlay page that showed an eye catching animation with 'call-to-action' text (aka proxemics hint) in the top right corner (Fig. 8).

Together with version 2, a new version of the UBI-portal was introduced. It comprised of two parts, an always visible bottom bar menu that allowed opening a semi-transparent pop-up menu containing the launch icons of applications (Fig. 8). This design was later discarded and the menu of launch icons was moved to the same stack level as the other virtual screens, to streamline the number of clicks required to access applications (Fig. 8).

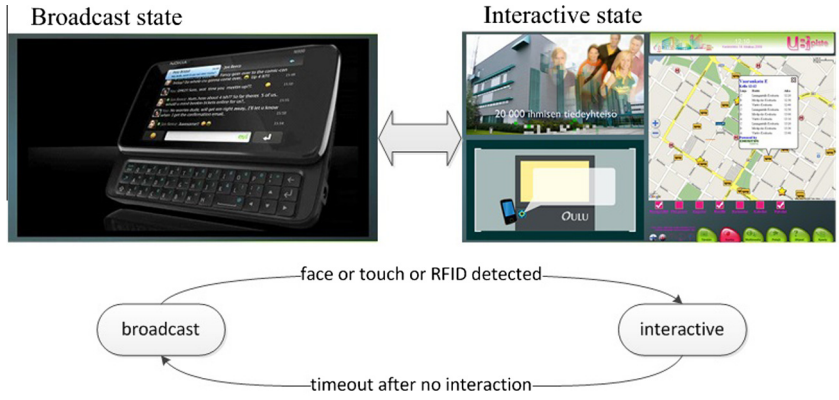

Fig. 6. Example screen layouts and interaction model in version 1.

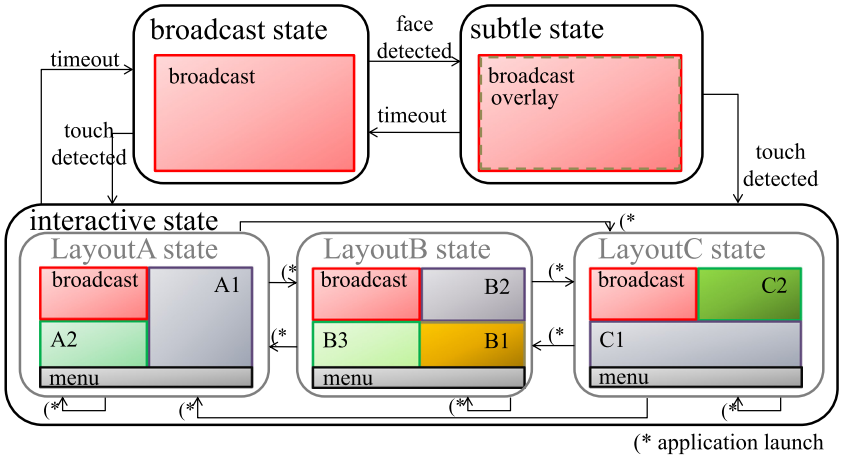

Fig. 7. Interaction model with corresponding layouts of version 2.

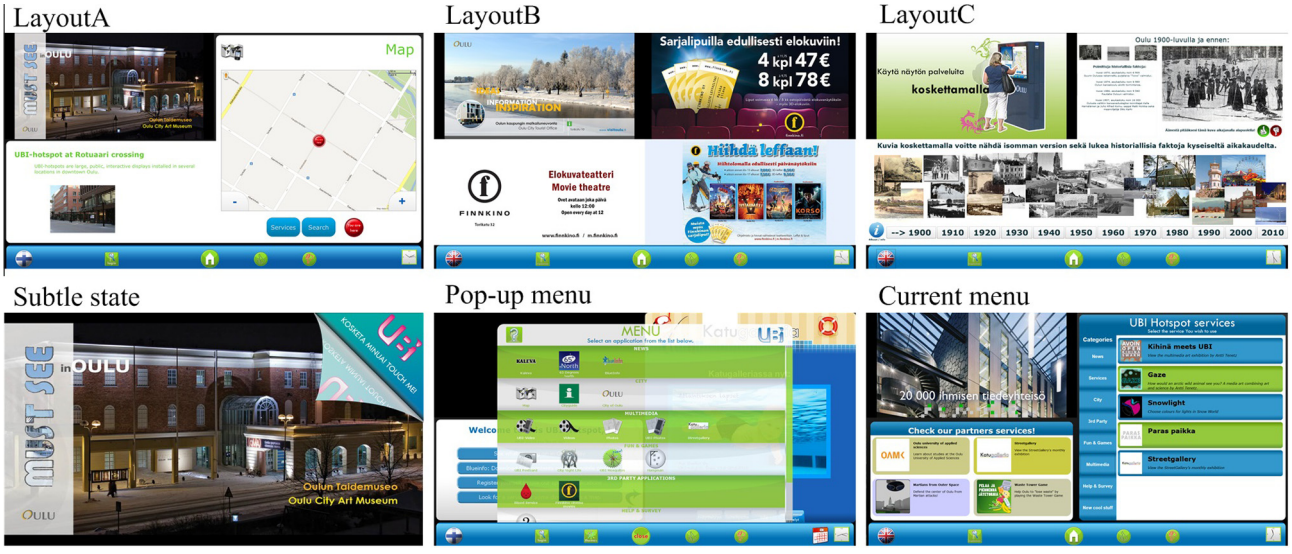

Fig. 8. Example screen layouts in version 2.

Whereas applications in version 1 were directly embedded into the UBI-portal with hyperlinks, in version 2 applications were registered into an application registry with metadata including application to hotspot mapping, application layout and virtual screen URLs. Thus, the concept of launching an application was introduced. When a user selects (launches) an application from the UBI-portal menu, its associated metadata is fetched from the registry and the particular layout and virtual screens are requested from the Layout Manager, which then sets the layout and loads the user interfaces into the virtual screens from their respective URLs.

In the interactive state, there are three different ways to spatially multiplex the screen as illustrated in Figs. 7 and 8: right half and bottom left quarter screens (LayoutA); top right, bottom right and bottom left quarter screens (LayoutB); and bottom half and top right quarter screens (LayoutC). Developers can use any of these three alternate layouts in their applications.

3.3.3. Version 3: Tandem Browsing Toolkit for building distributed multi-display interfaces

Version 3 dubbed as the Tandem Browsing Toolkit extended the functionality of Layout Manager to creating distributed multi-display interfaces [9]. The toolkit implements the concept of "tandem browsing" with the finite state machines and XML layout state encodings used in the version 2. The toolkit allows creating strongly coupled multi-display interfaces whose state is temporally synchronized across devices [35]. The participating displays can have different form factor and software platform as the toolkit is fully web based. Also, as another benefit of being web based, the toolkit does not require any application installation or custom setups on the client devices [36]. This was observed as one reason for the lack of use of the distributed mobile applications in version 1, as it required users to go through a cumbersome registration process at the beginning. With version 3, the focus also evolved from a single reference implementation into more general purpose toolkit targeted for the developers of multi-device interfaces [37]. The implementation of the concept of "tandem browsing" required minor changes in the SCXML finite state machine descriptions, to extend the concept of layout to cover multiple devices in tandem browsing.

The Tandem Browsing architecture is shown in Fig. 9. The Tandem Browsing Toolkit enforces runtime layout inside a browser based on finite state machine similarly to version 2, but in addition it synchronizes the session among multiple client devices which share the same 'extended' layout. The toolkit can therefore orchestrate the layout on multiple devices simultaneously based

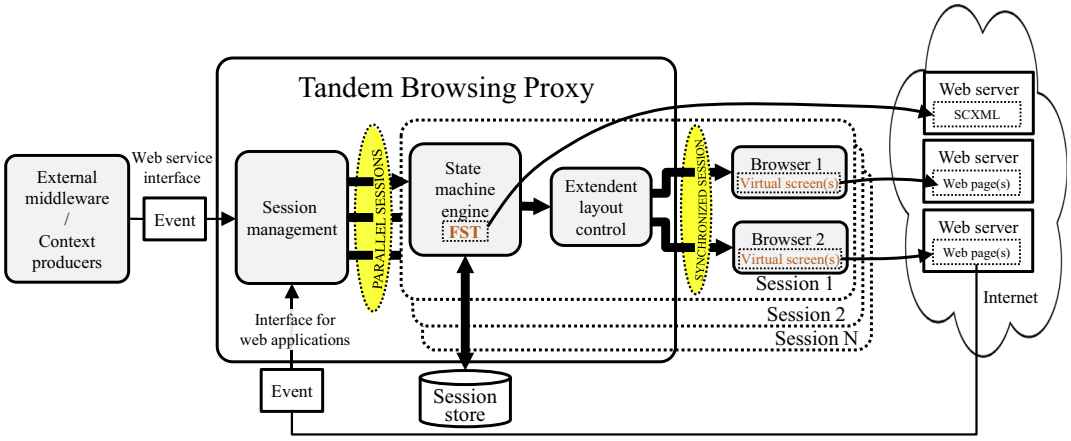

Fig. 9. Proxy-based tandem browsing architecture.

on a finite state machine description containing information about how different virtual screens should appear on available devices. The previous versions had a fixed connection between a single client and a Layout Manager which has been removed in the version 3 by introducing a proxy-based architecture. This architecture makes the toolkit effectively a multi-session capable. This provides additional flexibility for the development and deployment of display experiences as a single proxy server instance can manage multiple finite state machines with their corresponding client devices concurrently. These finite state machines could be controlling the same (nested) or completely separated user experiences.

All the client devices maintain a persistent connection with the proxy server. Devices belonging to the same session share the same session id, which is passed as a parameter in the proxy server URL. These same session ids are used by any external middleware for mapping the control events to the right sessions. A parallel control interface was also added to the proxy server for the web applications accessible from applications rather than middleware, which allows the creation of self-contained applications without external control. In order to be able to link to SCXML files directly, a link embedding method was introduced that allows accessing the proxy server instance and the finite state machine with a single HTTP request. This is similar to the fat link concept of [38].

The toolkit was taken into use in the hotspots in October 2012, replacing the second version of the Layout Manager. At the same time additional layout states were introduced to the finite state machines, to support mobile phone interaction with the hotspots. For each application layout shown in Fig. 8, an additional virtual screen was included, which allowed designing distributed tandem browsing user interfaces for the hotspots, as illustrated in Fig. 10. The coupling of a mobile phone with the hotspot could be made alternatively with QR codes or short URLs both containing the proxy server URL with session id, thus avoiding any specific application installation other than browser and general purpose QR code reader application. The motivation behind the setup was that users would need to make the coupling just once and after that the mobile phone would follow the hotspot session, allowing multiple application launches until the user left the hotspot.

Fig. 10. The concept of tandem application in version 3.

3.3.4. Summary of the evolution of the layout management middleware and application development

Table 1 summarizes the key features introduced in each version of the layout management middleware. Version 1 focused on introducing the core framework and principal content abstractions to satisfy the main stakeholder requirements. In version 1 the hotspots already supported digital signage content, interactive applications and mobile application interactions for Java based mobile apps. Since version 1 the basic technology choices have persisted, such as the web based framework and application platform, as well as the basic implementation of the digital signage system. In version 1 applications were embedded into the UBI-portal, which made the development and integration of third party applications cumbersome, as the addition of any new application required significant changes to the UBI-portal. At this point all applications were programmed in-house by the UBI researchers, with the exception of the Oulu Today service provided by a local newspaper that as a partner of the UBI program had priviledged access to the hotspot content creation process.

Version 2 solved two main shortcomings of version 1: the inflexibility of the interaction model and layout, and the lack of support for 3rd party application development. Since the

Table 1
Comparison of the three different versions of the layout management middleware.

Feature	Version 1	Version 2	Version 3
Web based layout management framework	√	√	√
Digital signage content	√	√	√
Application platform	√	√	√
Mobile application interaction	√	√	√
Configurable interaction model (FST)	–	√	√
3rd party applications	–	√	√
Application registry	–	√	√
Web based multi display applications	–	–	√
'Tandem browsing' applications with extended layout	–	–	√
General purpose toolkit for multi-device interfaces	–	–	√

introduction of version 2, the SCXML FSTs have allowed rapid configuration of the hotspots and the customization of hotspots at specific locations based on emerging requirements as visualized in Fig. 4. 3rd party application development was also launched along version 2. Due to the application registry the development process evolved to a level where the support required from UBI researchers is minimal, potentially including simply a confirmation of the deployment. New applications were no longer cumbersomely embedded into the UBI-portal. Instead, they were added via a web page into an application registry with metadata including the desired application layout and the URLs of the web applications providing the user interfaces. Since then the number of applications developed by 3rd parties has grown rapidly and UBI researches have worked in the interface between display owners and application developers, collecting feedback and developing the system further.

Version 3 focused on the implementation of the 'tandem browsing' concept with multi-display applications. The applications are fully web based and the same content creation process applies for hotspots as well as to other screens such as mobile phones, making the development simple and allowing reusing the same content on each display device. Developers could also coordinate content more flexibly using the SCXML FSTs similar to the single display case. The complexity of the application development process did not significantly increase for the tandem version as the distributed user interface elements are just additional virtual screens in the 'extended' layout. The biggest challenge was the implementation of communication between the devices that goes beyond the synchronization of the layout such as passing strings between virtual screens. For this purpose we employed an event-based communication system capable of routing messages [39]. Together, these changes made the system more general purpose and formed a platform for developing multi-display interfaces.

3.4. Reflection on real-world usage

The screen management middleware provides two principal abstractions to content producers and application developers, a classical digital signage service dubbed the UBI-channel, and a portal of interactive applications dubbed the UBI-portal. The screen real estate is spatiotemporally multiplexed between the two abstractions according to the third abstraction: the SCXML finite state machine with layout encodings describing the overall interaction model.

Since 2009, about 2000 campaigns of varying duration and content have been shown in the UBI-channel by about 100 commercial customers and six non-commercial customers. The commercial campaigns have contributed about 90% of the about 500,000 EUR total revenue that has been invested in covering the operational expenses of the hotspots. The remaining 10% of the revenue has come from customers who have paid for the right to offer an interactive application in the UBI-portal. Indeed, the hotspots have created a real-world value network [27] (R2a). The average daily number of faces detected looking at a hotspot was about 5500 in the first half of year 2014 providing an estimate of the number of eye contacts with the UBI-channel (R3c).

So far, about 100 interactive applications have been developed for the UBI-portal, of which about 80 have eventually been deployed on the hotspots. About half of the applications have been created by external third party programmers such as international researchers, industry programmers, new media artists and students. Of the three layout options introduced in the second version, about 89% of applications have used LayoutA. Typically, 25–30 applications have been deployed in the UBI-portal at any given point of time, thus the applications have had a certain turnover time. A few applications have been available in the UBI-portal for the whole six years, possibly with some upgrades on the way. The UBI-portal applications have been constantly used by real users, independently on their own. For instance, the 4-year average from June 2010 to June 2014 was 620 service launches daily across all hotspots. Our goal has been to allow remote development and deployment of UBI-portal applications with the help of realistic emulators and application registry. However, we have learned that testing on the real hardware is still an important step and is something that cannot be fully avoided. All in all, as to our own requirement (R6a) the hotspots have served as a platform for longitudinal field study offering multiple interactive services by multiple providers.

In terms of the evolution of the interaction model of the hotspots, in total 16 different SCXML FSTs have been created so far. While their creation has been our researchers' (display owners) responsibility, the Tandem Browsing Toolkit has been offered also for use in student projects in two distinct undergrad university courses. Overall, in terms of learning, the state chart concept and the toolkit appear to require some additional effort on top of the standard web application development process. However, their benefits become obvious with more complicated application scenarios with multiple input technologies and in settings requiring robust operation from the system – qualities that are challenging to accomplish with standard web technologies.

4. Discussion

4.1. Web as a development platform

We introduced a web based screen management framework for multipurpose interactive public displays. The user interface of the hotspots runs entirely in a standard web browser, thus applications are standard web applications. Therefore, application providers have often used their own web servers for developing and hosting applications to which the hotspot software simply refer by their URLs.

Many display systems have chosen web as their technology platform. The widespread use of web in public displays is due to its familiarity, the ease of development, and the ease of deploying and distribution of content. The hotspots have a web based software platform for these same reasons. The web based approach has been one key reason for the success in promoting third party application development for the hotspots. Developers with different skill levels have been able to contribute applications, due to the familiarity and flexibility of web technologies. Web also solves many problems for free such as accessing and naming resources and achieving platform independence.

Web is also a widely used platform for public displays perhaps due to the lack of any better alternative. Currently, there are no operating systems specifically designed for public displays. The operating systems for personal computers and mobile phones are optimized for their respective form factors and interaction models. However, web is increasingly gaining ground in the form of desktop like web applications and cloud services, i.e. working as the operating system across devices.

Despite its many benefits, using a web based platform in a public deployment requires some careful design. First, it is vulnerable to network connectivity problems, as the standard web application architecture does not have a standalone client that could work offline. Extra effort is needed to achieve offline operation, either by having a local web server for hosting critical content, or by making web clients to work offline in a browser. The first approach is used

in the hotspots, and while it is more difficult to setup, it will work even if the network is unavailable during the first request.

Second, the concept of application platform is more complex than simply referring to any web page by its URL. The choice of the underlying browser and the available plugins effectively define the execution environment for applications. In our case this essential information has been collected into the so-called UBI-portal media card, which effectively serves an informal contract between us display owners and application developers (R5b). Thus, the initial choice of the browser is crucial since it might be impossible to change it afterwards as all the third party applications have been developed with that particular browser in mind. In practice, the application platform tends to change over time as new security updates, web standards and technologies, and browser plugins are released and the browser is updated accordingly.

Due to the inevitable changes in the software platform, one has to consider how to persist the link between the platform, an application and the developer throughout the whole life cycle of the application (R5c). While the development process is typically very active in the beginning, the maintenance and updating of the application can be eventually forgotten. We have had to remove some applications from the hotspots when they no longer worked and the developers could not be reached or were no longer willing to fix them. While the application registry has been a step toward maintaining this connection, it has the limitation of being designed for display owners rather than application developers. There is ongoing research on app stores for public displays [40] that could potentially help in this regard.

Web is an attractive platform for interactive public displays for many reasons. Development time and effort can be significantly decreased as technologies are familiar, simple and many features are available for free. However, care needs to be taken, as web has not been designed for standalone services, which are a necessity for always on deployments. Web based middleware for public displays should address this gap.

4.2. Screen real estate management

We presented a web based toolkit for managing the screen real estate of multipurpose interactive public displays. The input mechanisms of the hotspots allow implicit to explicit interaction and different interaction inputs are causally related. The toolkit is based on finite state machines that allow structuring different inputs by providing an abstraction to create an overall interaction design that helps creating correctly working user interfaces (R6c).

The finite state machines used to define the interaction model are based on the Harel statecharts. Their expressive power allows the implementation of complex systems, but at the same time keeping the behavior in an easily conceivable format. The introduction of the SCXML layout state encoding language facilitated tailoring the interaction model and the user interface of the hotspots as the system evolved. The main interaction model consisting of the three states of broadcast, subtle and interactive has remained the same since summer 2010. However, additional screen layouts have been introduced for the interactive state, and specific interaction models have been defined on need basis. Also, layout encodings have been slightly adjusted as new elements have been introduced to the user interface.

4.2.1. Combining interactive content with scheduling

The hotspots are interaction driven displays with explicit and implicit interaction capabilities required by the City of Oulu (R1b) and users (R4). They also incorporate static scheduling based model, to provide conventional digital signage according to the requirements of the City of Oulu (R1a) and advertisers (R3a, R3b). UBI-channel player rendering the playlist is considered to be a regular application that operates inside a virtual screen like any other application. As public displays can remain unused for long periods of time, UBI-channel fills this space by exploiting the whole screen in the passive mode. Passive mode has been employed to show general purpose relevant content in many other displays, e.g. [21,41]. As half of the playlist is provided by the City of Oulu, the content may be relevant to the users by informing what is going on in the city and when in interactive state, it can be viewed from distance while another user is using an interactive application at arms distance (R6d).

The challenge of showing digital signage type of content in passive mode is that the interactivity of the hotspots is not clearly visible. In fact, they resemble a conventional digital signage display and can cause interaction blindness [3]. While the subtle state has been designed to entice interaction, the underlying face detection algorithm may miss the user due to poor alignment or challenging lighting conditions, or the user can miss the "call-to-action" message possibly due to hotspot showing the message a bit too late or it not being visible enough. This challenge has been addressed by attaching an always visible printed "call-to-action" sticker on the enclosures of the hotspots.

The disassociation between the interactive applications in UBI-portal and the content of UBI-channel playlist has been very clear cut so far. This has been acceptable for most UBI-portal application developers as they are typically different from UBI-channel content producers. However, some parties providing both an UBI-portal application and content to the UBI-channel have asked for a mechanism to associate the two so that the portal application would for example be launched when the corresponding spot in the playlist is clicked. While such an association could be made for the whole UBI-channel, individual spots cannot be singled out as the playlist is managed by a separate commercial system outside our administrative control and not providing such functionality. Nevertheless, this functionality dubbed by us as interactive digital signage is on our to-do list.

4.2.2. Virtual screen abstraction

The proposed middleware provides the virtual screens as the fundamental abstraction for application developers. They design web sites that are referenced in the virtual screens. The toolkit handles only the virtual screens and is agnostic to the content inside them. Each virtual screen has a particular size and position in the layout, which allows the high level design of the user interface including the isolation of its different parts. The virtual screen abstraction is similar to windows in personal computers, but is different in the sense that users cannot freely resize or move them, instead the layouts in different interaction stages are defined in the SCXML file, thus placing the system in control of the layout instead of users (R6b).

As a consequence of this design, application developers are obliged, or have the freedom, to define all user interface widgets they need. Web browser provides a set of default widgets for HTML elements, but they are based on using a mouse and are typically too small for touch interaction. Although the UBI-portal media card lists recommended design practices such as using large enough elements with proper relative sizes and placing interactive elements at the bottom of the screen for accessibility, the actual implementation is in the hands of the developers, a requirement stemming from the creative freedom of application developers (R5a). The use of common widgets (buttons, scrollbars, on-screen keyboard, etc.) could allow having a common look and feel among applications, thus lowering technological barriers for developers. On the other hand, this design could lead to more constrained applications and as to the feedback from application developers, this situation is clearly undesirable, while it could be beneficial for other stake holders such as display or place owners.

Since the introduction of version 2, application developers have needed to address the communication between virtual screens. It is complicated by a security feature of the browsers known as cross site scripting. Therefore, for example listening JavaScript events for button clicks in another virtual screen requires a bridge to pass the messages across. In a single display setting, the simplest way was to set the content of the second virtual screen using a normal hyperlink and specify the second frame as the link target. This worked only if both web pages were hosted on the same domain and the functionality was limited to replacing the whole content. For dynamic synchronization, developers needed to use server side technologies such as AJAX and web sockets. However, recent advances provide a straightforward method at the browser level using HTML5 web messaging which is available in most modern browsers. If the user interface is distributed across multiple devices, developers have to again resort to server side techniques until completely peer-to-peer based solutions such as WebRTC are gaining more support in browsers. All these methods are logically based on point-to-point connections. A step forward is to employ a communication system capable for routing messages such as the UBI-broker used by us [39].

A user interface toolkit for public displays need to provide application developers with sufficient abstractions that define the API. It is important to document the API properly and to present best practices for developers to comprise a well-defined application platform (R5b). These abstractions for the hotspots are simple, i.e. just the virtual screens. While the layouts of the virtual screens can be defined exactly, their inner layout and widgets are not defined. This is a crucial trade-off, which works in the hotspots partially due to the use of the web platform, which simplifies the development of applications from scratch.

4.2.3. Multi-device interfaces

The Tandem Browsing Toolkit allows developing multi-device interfaces where devices are strongly coupled, i.e. they are synchronized and share the same state. The toolkit allows defining clients for each virtual screen and thus allows controlling which clients get which virtual screens. The toolkit also allows multiple client sessions, i.e. multiple finite state machines executed simultaneously. The utility of the toolkit spans to public–private separation of distributed user interfaces, device specific application user interfaces, synchronizing content on networked displays, etc. and thus provides a platform for building multi-device applications with different interaction modalities (R6e).

The goal of the toolkit is to simplify the development of multi-device applications, which can be very cumbersome. Web technologies can make this easier due to the existence of many synchronization technologies such as web sockets, AJAX, reverse AJAX and Java Applets. However, similar to the single display case, finite state machines allow designing overall behavior, which is even more crucial in the multi-device case as the input and output mechanisms are distributed over multiple devices and the causal relations between the interactions can be difficult to track.

The presented toolkit synchronizes navigation between coupled displays. Providing implementations for virtual screens stays essentially the same, but the distribution of the user interface complicates the development process slightly. First, the dynamic content synchronization addressed in previous section becomes more challenging across devices. Second, the toolkit is missing a built-in mechanism for dynamically establishing and managing sessions between devices. However, the concept of session ids makes it easy to design session initialization on top of the toolkit. The basic structure is that all the devices participating in a session have to access the same toolkit instance and share the same session id. The chosen method depends on the use case, i.e. the setting and the devices. For example, QR codes and short URLs enable ad hoc coupling of mobile phones without prior installed software with co-located public displays. However, if the coupling is fixed or if the devices are not physically co-located, meeting rooms or pre-coupling of devices should be considered.

Despite these technical issues, the more pressing challenge is the slow user adoption of distributed public–private user interfaces. Earlier experiments have found the concept to be difficult for users and observed that people tend to approach the main display instead of using their mobile phones [42]. Experiences with the hotspots have been similar in that the users chose to collaborate directly on the public display instead of using their mobile phones [43]. Thus, it seems that in this use case advanced technology comes in the way of the interaction instead of supporting it.

4.3. Stakeholder requirements for MPDs

In this paper we have discussed the direct and indirect requirements for the layout management middleware from various stakeholders who have been involved in the process of designing, implementing, leveraging and operating the UBI-hotspots for six years. Some of the stakeholders – place owners, financiers, advertisers and content creators – had a bigger role at the beginning of the process by setting up the ground rules for the hotpots how they were supposed to operate and basically defining them as MPDs. Other stakeholders – users, application developers and display owners – have been more active in the operation phase of the hotspots, when the impact of the display infrastructure has been observable and potential development directions have been more visible.

UBI-hotspots are MPDs that by their placement support naturally spontaneous walk up and use interaction. The designed middleware enables the spatiotemporal multiplexing of content in the user interface using the scheduling model based on place owner and advertiser requirements. At the same time the middleware enables spontaneous user interactions required by the place owner, users, application developers and display owners. While the final model has resulted from a specific design process of the hotspots, systems with similar qualities have been created elsewhere [17,21]. When a display network is opened up to all relevant stakeholders early in their development process, it is very likely that content requirements and interaction modalities will become heterogeneous, i.e. leading to a MPD. Therefore, the requirements presented here should generalize also to other multi-stakeholder contexts than that of the hotspots.

The key stakeholders for this work have been (1) content creators and application developers, who have been responsible for creating the content for the MPDs, and (2) the display and place owners, who have been responsible for defining the interaction model and the rules for the spatiotemporal multiplexing of the content. The content creators and application developers have been involved in the process early on and their requirements have been analyzed while the development of hotspots has advanced. In the context of the hotspots, advertisers and local officials have been served with a conventional digital signage system. There, the only potential dilemma has been competition for screen real estate. However, interestingly, the interactive services in the UBI-portal have been regarded only as a positive aspect in attracting attention toward ads rather than as competitors for the visibility of the ads. Application developers have adopted this new media well and the main feedback has been related to some implementation challenges due to the multipart page. The best evidence of success is requests from arbitrary organizations who want to build an application for the hotspots out of their own interest and its subsequent successful execution. However, it is to be noted that the fight for the visibility also occurred between application developers. We responded by creating a special menu structure that gives more

visibility for desired applications. Further, some developers have required an exclusive access to the whole screen real estate for a certain period of time.

The place owners, the City of Oulu and the specific locations have been also involved in the process early on. Their requirements have been mostly related to the outlook of the whole apparatus and getting their own content to the hotspots. The UBI-channel has been much more popular 'channel' for the place owners than the UBI-portal, mainly due to the lack of sufficient web development skills in the place owner's organization. Although UBI researchers have occasionally helped place owners in the application development process, the fundamental challenge remains that many of the potential opportunities for place owners have been missed due to the lack of skills needed to create interactive content, even when the creation is made relatively trivial. UBI researchers have also served as display owners for the hotspots, which naturally has made it challenging to identify authentic requirements for that stakeholder group. As a research group, there can be slight tendency to over resource things and over design the features relevant for display owners, compared to the truly minimum mandatory functionality. However, due to the real-world nature and long duration of the deployment, challenges have been authentic, and over resourcing has quickly turned into tight resourcing, which is likely the case in most long deployments. The hotspots have provided a realistic testbed for a holistic real-world study of MPDs, where proved lab experiments have turned out to be less successful and only certain features have proved to be valuable in the real world setting. Then, on the other hand, features that where initially designed only per research motivation such as instrumenting the software components, have turned out to be valuable also in the real world case as application developers are interested in their application's use compared to other applications, which is not possible to determine based on solely from the application specific log data.

5. Conclusion

We presented the incremental design and evolution of screen management middleware along the over six years long deployment of the hotspots. The APIs of the middleware, particularly the familiar abstraction of applications as web pages, has allowed a wide range of application developers to create dozens of applications that have been used by tens of thousands of real users. This demonstrates the practical importance of the work in the development of a diverse range of interactive applications for interactive multipurpose public displays. An obvious limitation of the middleware is that it does not allow non-web applications, although the underlying operating system could support them. This limitation appears acceptable as demonstrated by the many third party applications developed for the hotspots.

Note

The Tandem Browsing Toolkit is available as open source at http://tandembrowsing.org/, to allow third parties to use it as a middleware toolkit for building multi-display interfaces and also invite others to contribute to the source code.

Acknowledgments

The authors are grateful for the support of the Academy of Finland, the Finnish Funding Agency for Innovation, the ERDF, the City of Oulu and the UBI consortium.

References

[1] V. Kostakos, T. Ojala, Public displays invade urban spaces, IEEE Pervasive Comput. 12 (1) (2013) 8–13.
[2] M. Weiser, The computer for the 21st century, Sci. Am. 265 (3) (1991) 94–104.
[3] T. Ojala, V. Kostakos, H. Kukka, T. Heikkinen, T. Linden, M. Jurmu, S. Hosio, F. Kruger, D. Zanni, Multipurpose interactive public displays in the wild: three years later, Computer 45 (5) (2012) 42–49.
[4] N. Davies, S. Clinch, F. Alt, Pervasive displays: understanding the future of digital signage, Synth. Lect. Mob. Pervasive Comput. 8 (1) (2014) 1–128.
[5] P. Peltonen, E. Kurvinen, A. Salovaara, G. Jacucci, T. Ilmonen, J. Evans, A. Oulasvirta, P. Saarikko, It's Mine, Don't Touch!: interactions at a large multi-touch display in a city centre, in: Proc. CHI '08, 2008, pp. 1285–1294.
[6] S. Izadi, H. Brignull, T. Rodden, Y. Rogers, M.Underwood, Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media, in: Proc. UIST'03, 2003, pp. 159–168.
[7] T. Linden, T. Heikkinen, T. Ojala, H. Kukka, M. Jurmu, Web-based framework for spatiotemporal screen real estate management of interactive public displays, in: Proc. WWW'10, 2010, pp. 1277–1280.
[8] T. Heikkinen, T. Lindén, M. Jurmu, H. Kukka, T. Ojala, Declarative XML-based layout state encoding for managing screen real estate of interactive public displays, in: Proc. MUCS'11 (PERCOM Workshops), 2011, pp. 82–87.
[9] T. Heikkinen, J. Goncalves, V. Kostakos, I. Elhart, T. Ojala, Tandem Browsing toolkit: distributed multi-display interfaces with web technologies, in: Proc. PerDis'14, 2014, pp. 142–147.
[10] M. Ostkamp, J. Hülsermann, C. Kray, G. Bauer, Using mobile devices to enable visual multiplexing on public displays: three approaches compared, in: Proc. MUM'13, 2013 (article no. 2).
[11] C. Kray, K. Cheverst, D. Fitton, C. Sas, J. Patterson, M. Rouncefield, C. Stahl, Sharing control of dispersed situated displays between residential users, in: Proc. MobileHCI'06, 2006, pp. 61–68.
[12] F. Alt, A. Schmidt, J. Müller, Advertising on public display networks, Computer 45 (5) (2012) 50–56.
[13] O. Storz, A. Friday, N. Davies, Supporting content scheduling on situated public displays, Comput. Graph. 30 (5) (2006) 681–691.
[14] S. Clinch, N. Davies, A. Friday, C. Efstratiou, Reflections on the long-term use of an experimental digital signage system, in: Proc. UbiComp'11, 2011, pp. 133–142.
[15] S. Clinch, N. Davies, A. Friday, G. Clinch, Yarely: a software player for open pervasive display networks, in: Proc. PerDis'13, 2013, pp. 25–30.
[16] J. Müller, O. Paczkowski, A. Krüger, Situated public news and reminder displays, in: Proc. Am I'07, 2007, pp. 248–265.
[17] I. Elhart, M. Langheinrich, N. Memarovic, T. Heikkinen, Scheduling interactive and concurrently running applications in pervasive display networks, in: Proc. PerDis'14, 2014, pp. 104–109.
[18] T. Payne, E. David, N.R. Jennings, M. Sharifi, Auction mechanisms for efficient advertisement selection on public displays, in: Proc. ECAI'06, 2006, pp. 285–289.
[19] N. Davies, M. Langheinrich, S. Clinch, I. Elhart, A. Friday, T. Kubitza, B. Surajbali, Personalisation and privacy in future pervasive display networks, in: Proc. CHI'14, 2014, pp. 2357–2366.
[20] D.M. Russell, C. Drews, A. Sue, Social aspects of using large public interactive displays for collaboration, in: Proc. UbiComp'02, 2002, pp. 229–236.
[21] E.F. Churchill, L. Nelson, L. Denoue, J. Helfman, P. Murphy, Sharing multimedia content with interactive public displays: a case study, in: Proc. DIS'04, 2004, pp. 7–16.
[22] E.F. Churchill, L. Nelson, G. Hsieh, Café life in the digital age: augmenting information flow in a café-work-entertainment space, in: Proc. CHI'06 Extended Abstracts, 2006, pp. 123–128.
[23] E.M. Huang, E.D. Mynatt, Semi-public displays for small, co-located groups, in: Proc. CHI'03, 2003, pp. 49–56.
[24] B. Congleton, M.S. Ackerman, M.W. Newman, The ProD framework for proactive displays, in: UIST'08, 2008, pp. 221–230.
[25] D. Vogel, R. Balakrishnan, Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users, in: Proc. UIST'04, 2004, pp. 137–146.
[26] H. Kukka, A. Luusua, J. Ylipulli, T. Suopajärvi, V. Kostakos, T. Ojala, From cyberpunk to calm urban computing: exploring the role of technology in the future cityscape, Technol. Forecast. Soc. Change 84 (2014) 29–42.
[27] T. Ojala, V. Valkama, H. Kukka, T. Heikkinen, T. Linden, M. Jurmu, F. Kruger, S. Hosio, UBI-hotspots: sustainable ecosystem infrastructure for real world urban computing research and business, in: Proc. MEDES'10, 2010, pp. 196–202.
[28] T. Ojala, V. Kostakos, UBI Challenge: research coopetition on real-world urban computing, in: Proc. 10th International Conference on Mobile and Ubiquitous Multimedia (MUM 2011), Beijing, China, 2011, pp. 205–208.
[29] T. Ojala. 2nd International UBI challenge 2013, in: Proc. 12th International Conference on Mobile and Ubiquitous Multimedia (MUM 2013), Luleå, Sweden, 2013 (article no. 51).
[30] I. Horrocks, Constructing the User Interface with Statecharts, Addison-Wesley, Boston, 1999.
[31] B. Myers, S.E. Hudson, R. Pausch, Past, present, and future of user interface software tools, ACM Trans. Comput. -Hum. Interact. 7 (1) (2000) 3–28.
[32] S. Hosio, M. Jurmu, H. Kukka, J. Riekki, T. Ojala, Supporting distributed private and public user interfaces in urban environments, in: Proc. HotMobile'10, 2010, pp. 25–30.

[33] D. Harel, Statecharts: a visual formalism for complex systems, Sci. Comput. Programm. 8 (3) (1987) 231–274.
[34] N. Marquardt, S. Greenberg, Informing the design of proxemic interactions, IEEE Pervasive Comput. 11 (2) (2012) 14–23.
[35] A. Dix, C. Sas, Public displays and private devices: a design space analysis, in: Proc. CHI'08 Extended Abstracts, 2008 (article no. 4).
[36] R. Atterer, A. Schmidt, M. Wnuk, A proxy-based infrastructure for web application sharing and remote collaboration on web pages, in: Proc. CHI'07, 2007, pp. 74–87.
[37] R. Han, V. Perret, M. Naghshineh, WebSplitter: a unified XML framework for multi-device collaborative Web browsing, in: Proc. CSCW'00, 2000, pp. 221–230.
[38] B. Johanson, S. Ponnekanti, C. Sengupta, A. Fox, Multibrowsing: moving web content across multiple displays, in: Proc. UbiComp'01, 2001, pp. 346–353.
[39] T. Heikkinen, P. Luojus, T. Ojala, UbiBroker: event-based communication architecture for pervasive display networks, in: Proc. PD-Apps'14 PerCom Workshop, 2014, pp. 512–518.
[40] S. Clinch, N. Davies, T. Kubitza, A. Schmidt, Designing application stores for public display networks, in: Proc. PerDis'12, 2012 (article no. 10).
[41] J. Finney, S. Wade, N. Davies, A. Friday, Flump: the flexible ubiquitous monitor project, in: Proc. Cabernet Radicals Workshop, 1996.
[42] N. Kaviani, M. Finke, S. Fels, R. Lea, H. Wang, What goes where? Designing interactive large public display applications for mobile device interaction, in: Proc. ICIMCS'09, 2009, pp. 129–138.
[43] P. Luojus, J. Koskela, K. Ollila, S. Mäki, R. Kulpa-Bogossia, T. Heikkinen, T. Ojala, Wordster: collaborative versus competitive gaming using interactive public displays and mobile phones, in: Proc. PerDis'13, 2013, pp. 109–114.

Tommi Heikkinen is a doctoral candidate at the Department of Computer Science and Engineering at the University of Oulu. His research has focused on middleware components for interactive public displays ranging from screen management to multi-display interfaces and event-based communication.

Timo Ojala is a Professor of computer science at the Department of Computer Science and Engineering at the University of Oulu, where he leads the Urban Computing and Cultures research group and the UBI (UrBan Interactions) research program. His research interests include ubiquitous computing, urban informatics and hybrid spaces.