Sharing Kinetic Interactions for Mobile Devices

Bashar Altakrouri[1,2], Darren Carlson[1], and Andreas Schrader[1]

[1] Ambient Computing Group, Institute of Telematics
University of Luebeck, Luebeck, Germany
{altakrouri,carlson,schrader}@itm.uni-luebeck.de
https://www.itm.uni-luebeck.de/
[2] Graduate School for Computing in Medicine and Life Sciences,
University of Luebeck

Abstract. Infrastructure for sharing, adapting and deploying interaction techniques remains an enduring challenge for real-world pervasive computing ecosystems (ambient spaces). In this paper, we address this challenge by introducing the concept of *Interaction Plugins*, which enables interaction techniques to be constructed as shareable units of functionality and dynamically deployed into a variety of ambient spaces during runtime. To this end, this paper will discuss two important issues in detail: community-based creation of interaction plugins and runtime deployment of interaction plugins. The paper also features a mobile-based implementation of this approach based on the Dynamix context framework.

Keywords: Ambient Assisted Living, Natural Interactions, Kinetic Interactions, Sharing Interactions.

1 Introduction

Interaction possibilities in many environments are rapidly increasing due to the emergence of interconnected mobile devices, smart objects, and seamlessly integrated context-aware services. Natural Interfaces (NI) is one of the most frequent solutions being proposed to support the flow of (inter-)action patterns in these hybrid environments. NI techniques enable human users to interact with the physical space using familiar physical body interactions and intermediaries [19]. This is clearly visible through broad adoption of natural interaction techniques as a primary source of interaction within ambient spaces, leading to a rapid increase in the number of techniques proposed by research and commercial efforts. Pioneering work on NI have been reported in HCI literature, ranging from scratch-based interactions [8], accelerometer-based interactions [18], sensor-based interactions [18], ambient gestures [10], etc. Hence, methods for sharing interaction techniques are becoming more important then ever for the successful adoption and distribution of NI. In this paper, we will focus on kinetic-based interactions, which are characterized by motion and movement activities [2].

Modelling, designing, and managing context information are significant problems facing application developers, particularly in the area of mobile computing.

A. Marcus (Ed.): DUXU/HCII 2013, Part III, LNCS 8014, pp. 327–336, 2013.

In [6] and [21], the authors point out that those problems are mainly caused by the absence of appropriate support for rich context types, approach extensibility, and easy-to-use context frameworks. In the literature, a variety of approaches have been proposed for solving those problems through specialized middleware [16], context servers [7], and environmental instrumentation [14]; however, such approaches fail to scale in wide-area mobile scenarios [6] [3].

In response to the increasing availability of commodity sensor data such as orientation and accelerometer information; geo-location; proximity; light levels; camera and microphone streams; etc [12]; we are seeing increasing interest in mobile context frameworks capable of providing abstractions for sensing; context modeling and representation; service discovery and binding; etc [4]. Such context information can be very useful to mobile applications, which can use it to fluidly adapt to the user's changing environment and context of use [12].

As an interesting milestone, dynamic component integration, which has been thoroughly investigated in ubiquitous and pervasive computing research, has been recently successfully applied in mobile environments. This approach enables software components to be discovered, downloaded and integrated on-demand as a means of adapting an application's behavior and enhancing its features [17]. For example, the Context-Aware Machine learning Framework for Android (CAMF) promotes plug-in-based adaptation on Android by introducing processing widgets as discrete abstractions used to hide machine learning complexities [20]. The Funf Open Sensing Framework[1] promotes statically-linked context modeling plug-ins integration. One of the first projects to support dynamic component integration on Android using OSGi container was the Mobile USers In Ubiquitous Computing Environments (MUSIC) system [5]. More recently, the Dynamix framework [4] was introduced as an open plug-and-play context framework for Android. It supports automatic discovery and integration of context sensing and acting plugins at runtime; 3rd party context plug-in support; and custom context representations based on Plain Old Java Objects (POJOs), which enables the creation of arbitrarily complex context events objects that are easy to parse and use.

In HCI research, Pruvost et al. [15] call for interaction environments to be open and dynamic, due to the complexity of interaction contexts. They argue for highly adaptable user interfaces that preserve utility and usability across contexts. In their described adaptation vision, they have presented the concept of Off-the-shelf Interaction Objects, which are pre-implemented bundles of code, intended to be reused and composed at runtime to provide the necessary adaptation required for the interaction technique. While their vision is focused on the structural adaptation of user interfaces and the adaptation of a running dialogue, our work is more concerned with the sharing aspects of natural interactions, especially kinetic interactions. The Gestureworks Core[2], which is limited to multitouch interactions, is one of the earliest multitouch gesture authoring solution for touch-enabled devices on a variety of platforms such as

[1] `http://funf.org`, visited on February 21th 2013.
[2] `http://gestureworks.com`, visited on February 23th 2013.

Flash; C++; Java; .NET; Python; and Unity. Based on the Gesture Markup Language (GML), the solution comes with a rich library of pre-built gestures and allows for new custom gestures and gesture sequences to be built by designers. The OpenNI is an open source SDK[3] used for the development of 3D sensing applications and middleware libraries. One of the main aims of this framework is to enhance the NI techniques development community; make it possible for developers to share ideas problems; share code with each other; and address the complete development lifecycle by a standard 3D sensing framework.

2 Approach

Despite rapid innovation in NI techniques, it is well understood that user interface adaptation and adoption in ambient spaces remain challenging problems [15], due to issues related to heterogeneity and distributivity; dynamic media mobility; and user mobility. In this paper, we argue that kinetic interactions are currently hindered due to the lack of:

- Systematic consideration of the increasing diversity of user populations in ambient spaces. In particular, existing approaches fail to address users with varying intrinsic sensorimotor capabilities and fail to comply with the general trend of designing for the whole body in motion.
- Effective means for documenting, adapting and deploying interaction techniques. Interaction techniques are currently hard-wired into applications, leaving few possibilities for the integration of new NI techniques at runtime. Consequently, the task of sharing interactions becomes unrealistic in many scenarios.

To address these hindrances, this paper presents the concept of **Interaction Plugins (IP)** as a novel approach for sharing kinetic-based NI techniques in ambient spaces. In our previous work [2], an IP is defined as *"an executable component in ambient interactive systems that encapsulates a single natural interaction technique with a set of interaction tasks as input and delivers higher level interaction primitives to applications based on specific interaction semantics"*. To the best of our knowledge, there is no research specifically targeted at community-based creation and sharing of encapsulated natural interaction techniques. Hence, we have defined our own structured approach based on three main design characteristics: matching users and NI physical context; precise and extensible NI descriptions (human and machine readable); and flexible deployment of NI plugins at runtime.

In this section, we will discuss two main design choices in our approach that foster the three aforementioned characteristics: 1.) Anthropometric driven matching and presentation of Interaction Plugins; and, 2.) On-demand wiring of interaction resources.

[3] `http://www.openni.org`, visited on February 23th 2013.

2.1 Anthropometric Driven Matching and Presentation of Interaction Plugins

NI techniques are inherently affected by a wide range of physical impairments and disabilities such as limited range of motion, tremors, impaired balance, gait, etc. These might occur only temporarily as a result of previous body exercises and contextual restrictions, but are particularly challenging for the senior citizens and elderly adults, due to the notable effect of ageing in one's physical and motor abilities [11][13]. In this work, we opted for an anthropometric approach for describing, discovering and presenting Interaction Plugins. As such, we've extended work from Labanotation (Kinetography) [9], which is one of the most powerful systems for describing physical movements for dance choreography, physical therapy, and drama. We defined a complimentary Labanotation XML representation, which is used to analyze, describe, document, and preserve a plugin's required physical movements, enabling plug-ins to reason about all physical movements required for a given interaction scenario. Our plugin discovery mechanisms utilize this information to optimize the selection of suitable Interaction Plugins for a given scenario and user.

Our investigation revealed three essential building blocks (information components) for anthropometric driven IP. First, we utilize a movement information component represented by a profile that is used to describe the essential physical movement required by the interaction from the user and interaction capture point of view. For example, for a balance interaction technique, this profile precisely describes the body balance, position and posture required by the interaction. The flexibility of Labanotation allows interaction developers to set the abstraction details of the movement to reflect essential and important aspects of the interaction technique. Interaction developers may also describe the detailed micro-aspects of the movement, which may lead to a complex but precise movement description. In addition, they may describe the movement on a macro level while leaving a lot of movement details out.

The ability profile is the second information component needed in our design. This component defines the physical abilities that are required for an appropriate execution of NI. It is composed from two building blocks: major physical activities vital for the interaction, such as holding, standing, balancing, lifting, walking, etc; and physical disabilities that may impact the quality execution of the interaction. Both aspects are evaluated through an impact score for each, which defines their importance and impact to the quality of interaction execution. We argue that movement and ability profiles are essential for designing interactions "for all" instead of focusing on a limited population percentile and to avoid inaccessible interfaces.

The third component in our model is the interaction profile that includes the main interaction semantics, including the interaction primitives such as pointing, selecting, dragging, etc. This is important in ambient spaces because it provides an indication of the main use of the interaction technique and its offerings. This profile is also used by the application when subscribing to available interaction plugins that provide the same type of interaction primitives.

2.2 On-Demand Wiring of Interaction Resources

Interaction sharing can only be realized with a comprehensive approach for on-demand discovery, selection and integration of interaction modules at runtime. Support for runtime provisioning of suitable Interaction Plug-ins should be based on the user's capabilities and a plugin's required physical movements. Figure 1 illustrates a conceptual view of Interaction Plug-ins deployment on mobile devices. IP are discovered, loaded into devices on demand, activated seamlessly, and exposed to interactive applications through an easy subscription mechanism.

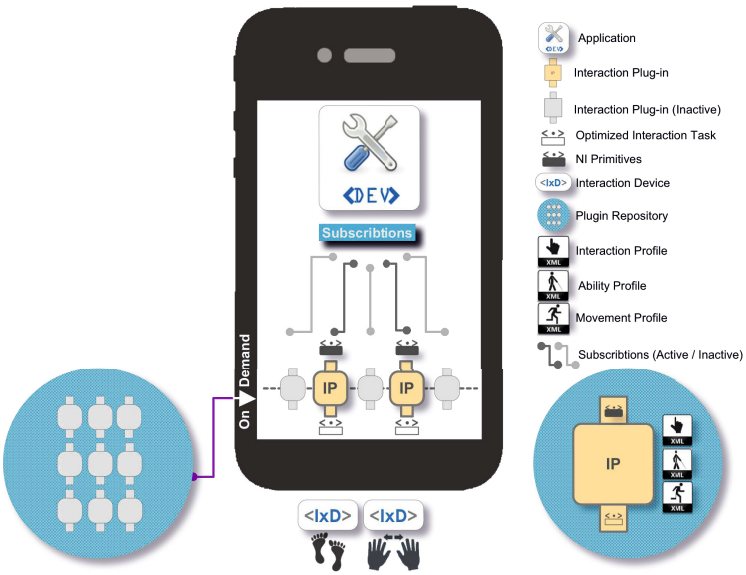

Fig. 1. Interaction Plug-ins Deployment - Conceptual Overview

Interaction Plug-ins are hosted by a plug-in repository, where they can be requested and downloaded on-demand by applications. IPs expose their offered interaction capabilities though interaction primitives (e.g., pointing or selection) through which interactive applications can subscribe to the plug-in. After deployment, the IP resides in the user's mobile device in an inactive state. This state is changed to active once a subscription is received by the plug-in, in order to allow plug-ins to only occupy the needed interaction and computation resources. This allows to attend to certain amount of plug-ins at a time, leading to save resources and computation power on the device. During runtime, interaction events are fired and delivered to the application. In our current model, applications can be informed about which plug-in to subscribe to and activate based on the physical abilities and qualities required by an application in a particular user context.

3 Sharing IP Based on the Dynamix Framework

In our previous work on Ambient Dynamix (our plug-and-play context framework for Android) [4], we described our comprehensive approach for on-demand discovery and runtime integration of context sensing and acting capabilities in wide-area mobile contexts. In this work, we leverage Dynamix as a mechanism for sharing natural interactions on mobile devices, due to its unique capabilities and flexibility, especially related to dynamic discovery and deployment of suitable context plugins during runtime.

Figure 2 illustrates our underlying technical approach for sharing natural interactions as IPs based on the Dynamix framework on the Android platform. In this approach, the Dynamix framework runs as a background service (Dynamix Service) and is situated between Dynamix enabled applications and the device's hardware (in our case interaction resources).

Dynamix-enabled applications are standard Android applications with extra context modeling functionality provided by a local Dynamix Service. We developed an Interaction Manager module for applications, which controls the

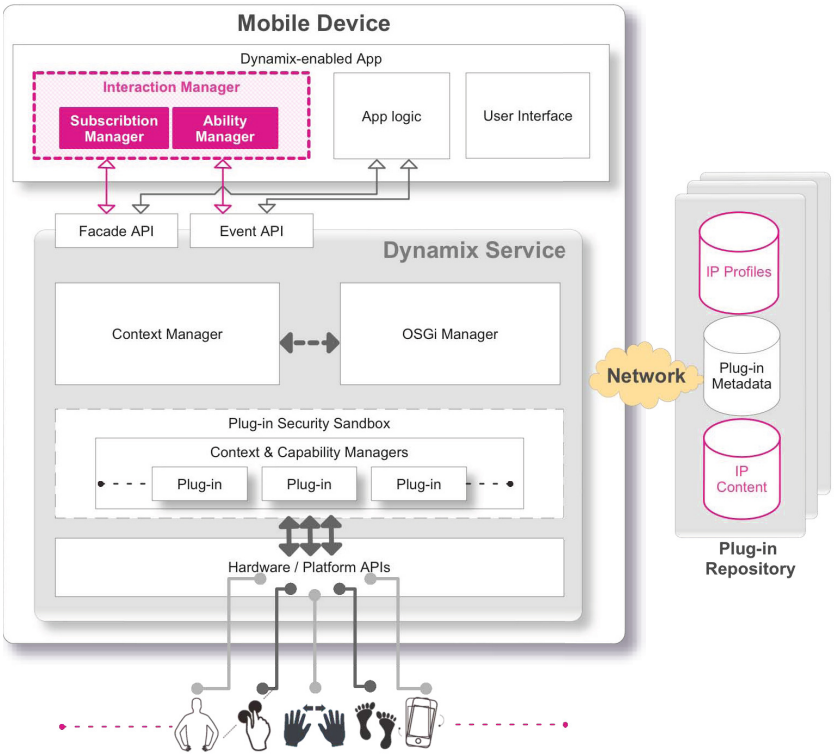

Fig. 2. Interaction Plugins Realisation Using Dynamix (Based on [4])

activation of the available interaction plugins based on the ability, movement, and interaction profiles. The Interaction Manager and the application logic communicate with Dynamix service through its Facade API, which enables apps to request and control context modeling support; and its Event API, which enables apps to receive framework notifications and context events.

Interaction plugins are tailored OSGi-based Bundles, which are loaded into the Dynamix embedded OSGi container at runtime. Once loaded and activated, the plug-in sends interaction encoded events to subscribed applications (as interaction primitive events) using POJOs. An Internet-based plugin repository hosts IP Bundle files, the IP profiles, and (optionally) additional related plugins. In addition to context sensing tasks, the IPs hosted by Dynamix can be queried by the application's Interaction Manager to access the information encoded in the available IP profiles. Currently, the Dynamix Service provides all plug-in discovery services, plug-in filtering based on the interaction requirements (interaction primitives required), and plug-in installation support. Filtering and activating the available interaction plugins are currently handled by the Interaction Manager; however, in future work we will introduce highly configurable plugin selection and activation functionality as a core extension of the Dynamix architecture.

4 Provisioning Interaction Plug-ins

The provisioning of an Interaction Plug-in is triggered once the interaction technique passes the usual design and implementation phases and satisfies the function and utility defined by an interaction developer (i.e., designer, developer, or team). The interaction developer should then define the interaction's movement profile, based on an acceptable level of movement description using Labanotation; define the interaction's ability profile, based on the most important physical qualities that impact the interaction; assign the interaction semantics, based on the envisioned utility and user study of the technique; wrap the interaction's internal logic as a Dynamix plug-in; and place the IP in an accessible repository. Interactive application developers can then easily create Android-based on the Dynamix framework and available IPs (according to the required semantics).

For illustration, Spin&Swing [1] is a spatial interaction technique that leverages the user's orientation for browsing temporal data on mobile phones. The movement profile defines the user's 360° medial rotation with the mobile held with right hand facing the front of the body. The ability profile defines important physical qualities for this interaction such as body balance, normal grip, and absence of hand trembling (the later quality has low importance, hence is given a low score). Moreover, the interaction profile defines selection as the main interaction primitive. Finally, the Spin&Swing IP wraps the core orientation-to-selection mapping based on the technique's original logic.

5 Discussion and Implications

In this paper, we have argued that flexible sharing techniques represent an essential cornerstone for adapting interactions to ambient spaces. Such techniques are increasingly vital due to the advancement of interaction technology; innovative interaction design; and large-scale adoption of natural interactions and applications. Without standardized means of describing and sharing interactions, the HCI community will suffer from a number of key problems, such as:

- a lack of proper documentation and preservation of novel interactions;
- unrealistic chances for collaborative interaction development and enhancement;
- continued inaccessibility of novel interaction techniques; and
- overly complicated real-world deployment of interaction techniques.

For the successful sharing of NI, interaction developers and designers should carefully consider the targeted user group and try to maximise the usability of the interaction based on the user's physical skills and abilities. The analysis of such information should be defined and encoded in the interaction's ability profile, which is used to help applications discover, select and activate associated plugins at runtime. The complexity and details of this profile can be left to the designer; however, describing the movement profile is a challenging task due to the flexibility and details of movement description provided by the movement annotation language (Labanotation in our case). The community should strive to find the right balance between abstraction and detail.

6 Conclusion

Ongoing research across various HCI disciplines has resulted in the development of a broad range of novel interaction techniques that emphasize the simultaneous use of multiple body parts during interactions with the surrounding environment. Unfortunately, these emerging interaction techniques often remain unavailable to real-world ambient computing systems due to a lack of wide-area deployment infrastructure. In this paper, we address this challenge by introducing the concept of *Interaction Plugins* as a means of sharing, adapting and deploying interaction techniques on-demand during runtime. We also describe the details of our approach for community-based plug-in creation and runtime plug-in deployment.

Acknowledgement. This work was partially supported by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative [DFG GSC 235/1].

References

[1] Altakrouri, B., Kawsar, F., Kortuem, G.: Spin&swing: Spatial interaction with orientation aware devices. In: The Eighth International Conference on Pervasive Computing, Pervasive 2010, Demo Section, May 17-20 (2010), http://www.pervasive2010.org/demos.html

[2] Altakrouri, B., Schrader, A.: Towards dynamic natural interaction ensembles. In: Devina Ramduny-Ellis, A.D., Gill, S. (eds.) Fourth International Workshop on Physicality (Physicality 2012) co-located with British HCI 2012 Conference, Birmingham, UK (September 2012)

[3] Blackstock, M., Lea, R., Krasic, C.: Toward wide area interaction with ubiquitous computing environments. In: Havinga, P., Lijding, M., Meratnia, N., Wegdam, M. (eds.) EuroSSC 2006. LNCS, vol. 4272, pp. 113–127. Springer, Heidelberg (2006), http://dx.doi.org/10.1007/11907503_9

[4] Carlson, D., Schrader, A.: Dynamix: An open plug-and-play context framework for android. In: Proceedings of the 3rd International Conference on the Internet of Things, IoT 2012 (2012)

[5] Consortium, I.M.: Òist-music: Context-aware self-adaptive platform for mobile applications (2010), http://ist-music.berlios.de

[6] Davies, N., Gellersen, H.W.: Beyond prototypes: challenges in deploying ubiquitous systems. IEEE Pervasive Computing 1(1), 26–35 (2002)

[7] Endres, C., Butz, A., MacWilliams, A.: A survey of software infrastructures and frameworks for ubiquitous computing. Mob. Inf. Syst. 1(1), 41–80 (2005), http://dl.acm.org/citation.cfm?id=1233803.1233806

[8] Gary Halajian, J.W.: Gesture recognition based on scratch inputs. Tech. rep., Cornell University (April 26, 2009), eCE 4760

[9] Hutchinson, A.: Labanotation The System of Analyzing and Recording Movement, 4th edn. Routledge, NewYork and London (2005)

[10] Karam, M., Hare, J., Lewis, P., Schraefel, M.: Ambient gestures. Tech. rep., Intelligence, Agents, Multimedia Group, University of Southampton (2006)

[11] Mahmud, M., Kurniawan, H.: Involving psychometric tests for input device evaluation with older people. In: Proceedings of the 17th Australia Conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future, OZCHI 2005, Computer-Human Interaction Special Interest Group (CHISIG) of Australia, Narrabundah, Australia, pp. 1–10 (2005), http://portal.acm.org/citation.cfm?id=1108368.1108399

[12] Niemelä, E., Latvakoski, J.: Survey of requirements and solutions for ubiquitous software. In: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, MUM 2004, pp. 71–78. ACM, New York (2004), http://doi.acm.org/10.1145/1052380.1052391

[13] Piper, A.M., Campbell, R., Hollan, J.D.: Exploring the accessibility and appeal of surface computing for older adult health care support. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, pp. 907–916. ACM, New York (2010), http://doi.acm.org/10.1145/1753326.1753461

[14] Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom 2000, pp. 32–43. ACM, New York (2000), http://doi.acm.org/10.1145/345910.345917

[15] Pruvost, G., Heinroth, T., Bellik, Y., Minker, W.: User Interaction Adaptation within Ambient Environments, ch. 5, Next Generation Intelligent Environments: Ambient Adaptive Systems edn., pp. 153–194. Springer, Boston (2011)
[16] Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: a middleware platform for active spaces. SIGMOBILE Mob. Comput. Commun. Rev. 6(4), 65–67 (2002), http://doi.acm.org/10.1145/643550.643558
[17] Schrader, A., Carlson, D.V., Busch, D.: Modular framework support for context-aware mobile cinema. Personal Ubiquitous Comput. 12(4), 299–306 (2008), http://dx.doi.org/10.1007/s00779-007-0151-6
[18] Scoditti, A., Blanch, R., Coutaz, J.: A novel taxonomy for gestural interaction techniques based on accelerometers. In: The 15th International Conference on Intelligent User Interfaces, IUI 2011, pp. 63–72. ACM, New York (2011), http://doi.acm.org/10.1145/1943403.1943414
[19] Wachs, J.P., Kölsch, M., Stern, H., Edan, Y.: Vision-based hand-gesture applications. Commun. ACM 54, 60–71 (2011), http://doi.acm.org/10.1145/1897816.1897838
[20] Wang, A.I., Ahmad, Q.K.: Camf - context-aware machine learning framework for android. In: Rey, M.D. (ed.) Iasted International Conference on Software Engineering and Applications, SEA 2010, CA, USA, November 8-10 (2010)
[21] Want, R., Pering, T.: System challenges for ubiquitous pervasive computing. In: The 27th International Conference on Software Engineering, ICSE 2005, pp. 9–14 (May 2005)