Interaction modularity in multi-device systems: a conceptual approach

Emmanuel Dubois, Augusto Celentano

Interaction modularity in multi-device systems: a conceptual approach

Emmanuel Dubois
IRIT – Elipse – University of Toulouse
Toulouse, France
emmanuel.dubois@irit.fr

Augusto Celentano
Università Ca' Foscari Venezia
Venezia, Italy
auce@unive.it

ABSTRACT

In this paper we propose a conceptual view on the modularization of multi-device interactive systems. Our view is supported by the re-visitation and adaptation of some software engineering concepts. We provide the definition of interaction module and its interface and discuss how these concepts contribute to the design of such systems.

KEYWORDS

Interaction module, interaction interface, multi-device

1 INTRODUCTION

Complex interactive systems based on multiple devices in a wide spatial environment and operated by a variety of interaction styles are receiving increasing attention and adoption in several application domains [12, 14, 17, 18]. In such systems interaction is more articulated, hence more dynamic and complex than in single device systems: users take control f a variety of devices during the execution of a complex procedure by changing role, task, location and attention in several smooth or discrete dynamic sequences that globally define the interactive experience. According to the task dynamics, information and representation change, interaction varies and splits among the devices, and information is distributed and/or replicated on them and over time.

A way to describe such situation consists in modelling the system behaviour as a graph where each node represents a phase of the application and the arcs denote the transitions between the phases. Without entering into details that are out of the size of this paper, a phase corresponds to a task having a self-contained identity, involving one device in a short span of time using a specific interaction technique and applied to one set of data (see [3] for details). In principle a task can be designed and implemented on different devices according to different interaction techniques. Hence the transitions between phases, besides leading to the execution of a different task and possibly a focus on a different dataset, can involve a change of device, a change of interaction technique, a displacement to another location, events that influence the *fluidity*, i.e., the smoothness of the interactive experience [11, 15].

In this view a phase is reified, at development level, into a software module executing an algorithm and interacting with the user. During the system execution the phase graph is traversed, generating a sequence of activations of software modules that build a specific interactive experience, more or less fluid according to the modules composition. This view reaffirms the presence of the user in the execution loop of a complex interactive system and raises a set of non-trivial questions related to the definition of the scope of a phase and how it can be interconnected to others, since they depend on the user's behaviour. A design space for supporting the reasoning about and design of multi-device systems is thus required together with suitable integration models.

2 SOFTWARE ENGINEERING CONCEPTS IN THE HCI CONTEXT

The literature reports several studies about the integration of software engineering (SE) principles and HCI, examined from different points of view: the integration of usability studies into SE [7, 9, 19]; the comparison of the requirements of the two domains and the active role of the user in modifying the system behaviour even in unanticipated way [2, 6]; the interdisciplinary approach needed to join human related concerns to the technical design issues [5, 8].

Taking a different perspective, to support the view introduced above we approach the design of multi-device interactive systems by adopting in a re-visited fashion some fundamental principles of software design. Among them, top-down design and modularity have a primary role, contributing to ensure that a system can evolve and be maintained in a sustainable way.

Top-down design. Interaction design should be layered in a sort of top-down fashion from the conception of the different

https://doi.org/10.1145/3206505.3206559

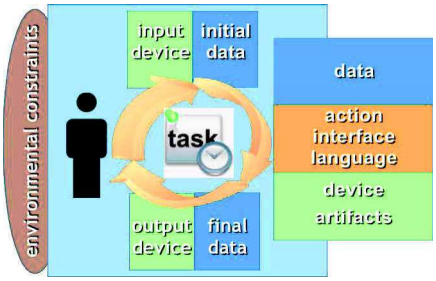

Figure 1: A module and its interface.

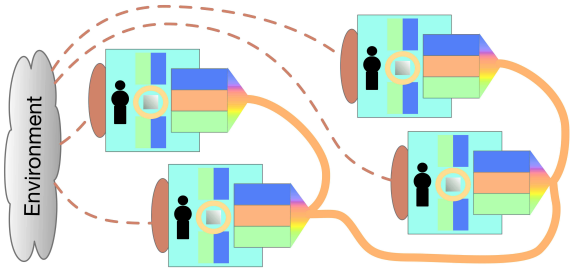

Figure 2: A sequence of modules with interfaces and environment dependencies.

phases and their interconnections down to the details of the individual low-level operations with a single device.

Modularity. To support the modularization of an interactive application, each phase should correspond to a module explicitly identified and described in terms of internal components and behaviour. A concise summary, the interface, should be exposed to the other modules. Different design choices and different implementations will impact not only the usability of each module, but also the overall interactive experience. Rules for assessing the compatibility of the interfaces in composing the phases/modules must thus be proposed and must include information related to the different points of view involved in their description.

The proposition of such rules raises two other principles of interest: the *analysis of the execution dynamics* that can be driven by the concept of *interaction trajectory* [1, 11, 20], and the distinction between *global vs local design*, leading to the duality between interaction design *in-the-large* and *in-the-small* [4, 10, 13]. These two last points will not be discussed here; their role in the design of multi-device interactive system has been partly explored in [4, 11].

3 MODULES AND INTERFACES

In the context of multi-device interactive systems, we adopt the concept of *interaction module* as an instance of a module in software engineering. It is an autonomous unit transforming an initial state of the system into a final one, but it is also under the control of a user. According to the modularity principle above, the designer should identify the *contour* of a module, describe its *content* and define the necessary *interface* to expose the module's role and capabilities (Figure 1).

The *internal component* and *behaviour* of a module implementing a function of an application program are normally defined according to some more or less formal requirement, e.g., a mathematical function, a model of the process to be implemented, etc. In the context of HCI the design of an interaction module is mostly the result of a creative process [16] grounded not only on functional specifications but also on the designer experience, the end-user co-design, the result of user tests, the environment, etc. In such cases modularization, i.e., what to put in a module and what to distribute over several modules, is not dominated by the functional specifications of the application but involves also the non-functional requirements imposed by the environment (e.g., a local vs a spatially distributed layout), the technology (e.g., the devices), the interaction model (e.g., gestural vs touch), the time and space constraints, etc. Hence, both the content of a module and its contour, i.e., the border defining what is put inside and what should be in other modules, must reflect this complex and multi-faceted design process: providing a description of the module thus involves different points of view including the user's task, the data processed, the user's interaction, and the technology required.

Given the targeted interactive system, the module's set of *data* is further described in terms of its initial and final states, including the data representation, structure and extension, and in terms of the *devices* used to be acquired and delivered to the user. In our view an interaction module is defined with reference to a specific device class, and supports a set of actions appropriate for that class.

An *interface* of an interaction module makes evident the articulation between modules supporting a top view on the capability and behaviour of the module. It must contain four dimensions: the *domain*, the *usage*, the *technology*, and the *environmental constraints*. The domain refers to the type of data the module is acquiring, transforming and delivering. The usage refers to the interaction style, the language in which it is expressed and the interface the user manipulates. The technology highlights the devices required to allow the use of the module. The environmental constraints refer to the parameters external to the module that influence its behaviour as noted above, among which the application playground, i.e., the different locations where the elements required by the module are used, and potentially other time and social constraints.

Concluding, it is traditionally admitted that the quality of the design of a set of modules implementing a sequence of phases (Figure 2) depends not only on the interface parameters describing the state transformation enacted by the modules, but also on the relations between the modules and the interaction environment. Our view highlights the fact that interruptions and changes in the type and use of devices (exposed in the modules' interfaces), spatial displacement and execution timing of different phases (defined in the environment constraints), globally affect the interaction-in-the-large properties and are relevant in modifying the perception of the system fluidity, ease and naturalness of use, and in general usability. These aspects are a solid scaffold towards the assessment of the assembly of a complex modular system.

REFERENCES

[1] S. Benford, G. Giannachi, B. Koleva, and T. Rodden. 2009. From Interaction to Trajectories: Designing Coherent Journeys Through User Experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 709–718.

[2] J. Brown. 1997. HCI and Requirements Engineering: Exploring Human-computer Interaction and Software Engineering Methodologies for the Creation of Interactive Software. *SIGCHI Bull.* 29, 1 (1997), 32–35.

[3] A. Celentano and E. Dubois. 2017. Interaction-in-the-large vs Interaction-in-the-small in Multi-device Systems. In *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter (CHItaly '17)*. ACM, New York, NY, USA, 19:1–19:10.

[4] A. Celentano and E. Dubois. 2017. A layered structure for a design space dedicated to rich interactive multimedia content. *Multimedia Tools and Applications* 76, 4 (2017), 5191–5220.

[5] S. Chatty. 2008. Supporting Multidisciplinary Software Composition for Interactive Applications. In *Proceedings of the 7th International Conference on Software Composition (SC'08)*. Springer-Verlag, Berlin, Heidelberg, 173–189.

[6] J. Coutaz. 1991. Architectural Design for User Interfaces. In *Proceedings of the 3rd European Software Engineering Conference (ESEC '91)*. Springer-Verlag, London, UK, UK, 7–22.

[7] F. Daniel, J. Yu, B. Benatallah, F. Casati, M. Matera, and R. Saint-Paul. 2007. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing* 11, 3 (2007), 59–66.

[8] M.G. de Paula, S.D.J. Barbosa, and C.J.P. de Lucena. 2005. Conveying Human-computer Interaction Concerns to Software Engineers Through an Interaction Model. In *Proceedings of the 2005 Latin American Conference on Human-computer Interaction (CLIHC '05)*. ACM, New York, NY, USA, 109–119.

[9] T. Di Mascio, L. Tarantino, and G. De Gasperis. 2015. If Usability Evaluation and Software Performance Evaluation Shook Their Hands: A Perspective. In *Product-Focused Software Process Improvement — 16th Int. Conf., PROFES 2015*. 479–489.

[10] A. Dix, D. Ramduny, and J. Wilkinson. 1998. Interaction in the large. *Interacting with Computers* 11, 1 (1998), 9–32.

[11] E. Dubois and A. Celentano. 2016. Analysing Interaction Trajectories in Multi-device Applications. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI '16)*. ACM, New York, NY, USA, Article 91, 6 pages.

[12] N. Elmqvist. 2011. Distributed user interfaces: State of the art. In *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem*, J. A. Gallud, R. Tesoriero, and V. M.R. Penichet (Eds.). Springer-Verlag, London, UK, 1–12.

[13] H. Johnson, E. O'Neill, and P. Johnson. 1999. Interaction in the Large: Developing a Framework for Integrating Models in HCI. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems (CHI EA '99)*. ACM, New York, NY, USA, 165–165.

[14] K. Luyten and K. Coninx. 2005. Distributed User Interface Elements to support Smart Interaction Spaces. In *IEEE International Symposium on Multimedia*. IEEE Computer Society, Los Alamitos, CA, USA, 277–286.

[15] R. Morris, P. Marshall, and Y. Rogers. 2008. Analysing fluid interaction across multiple displays. In *Workshop on designing multi-touch interaction techniques for coupled public and private displays at AVI 2008*. 5.

[16] B. Myers. 1994. Challenges of HCI Design and Implementation. *interactions* 1, 1 (1994), 73–83.

[17] F. Paternò and C. Santoro. 2012. A Logical Framework for Multi-device User Interfaces. In *Proc. of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '12)*. ACM, New York, NY, USA, 45–50.

[18] J.C. Roberts. 2007. State of the art: Coordinated & multiple views in exploratory visualization. In *Proc. of the Fifth Int. Conf. on Coordinated and Multiple Views in Exploratory Visualization, CMV '07*. IEEE Computer Society, 61–71.

[19] Metzker E. Seffah A., Desmarais M.C. 2005. HCI, Usability and Software Engineering Integration: Present and Future. In *Human-Centered Software Engineering — Integrating Usability in the Software Development Lifecycle*, Desmarais M.C. Seffah A., Gulliksen J. (Ed.). Human-Computer Interaction Series, Vol. 8. Springer, Dordrecht.

[20] R. Velt, S. Benford, and S. Reeves. 2017. A Survey of the Trajectories Conceptual Framework: Investigating Theory Use in HCI. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 2091–2105.