Title: Interaction-Driven Software Development
Titre: Développement logiciel dirigé par l'interaction
Domain: Human-Computer Interaction
Keywords: Software development, Instrumental Interaction, Information Substrates, Co-adaptation

Location: LRI & INRIA Saclay, Orsay, France
Equipe: HCC / ExSitu
Supervisor: Michel Beaudouin-Lafon, PR Univ. Paris-Sud (mbl@lri.fr)
Lab director: Yannis Manoussakis (Yannis.Manoussakis@lri.fr)

Profile of candidate:
Masters degree in Human-Computer Interaction or related field. Fluency in written and spoken English. Solid programming skills, including Java, Javascript, C or C++, and web programming skills.

Summary:
The goal of this thesis is to develop and apply the concepts of information substrates, instrumental interaction and co-adaptation to the development of interactive software.

Objectives:
The Ex Situ group explores the future of interaction by studying extreme users and extreme environments. Software developers are extreme users who need advanced tools to develop, test, integrate, distribute and maintain software. However their tools are not very good. Interactive Development Environments are very much specialized in a small part of the spectrum of development tasks, namely programming and, to a certain extent, testing and debugging. Issues such as bridging the gap between rapid prototyping tools and development, or between development and deployment on a variety of platforms, for example, are poorly supported. The goal of this PhD. is to explore how the concepts of instrumental interaction, information substrates and co-adaptation can be applied to software development tools. This exploration will be conducted through participatory design sessions with developers and the development of prototypes.

Context:
Ex Situ seeks to re-invent interaction using three conceptual foundations:

- Information substrates are software artifacts that embody content, computation and interaction, effectively blurring the distinction between documents and applications. Substrates create a malleable medium that can be manipulated in a variety of ways, naturally supporting instrumental interaction and co-adaptation.

- Instrumental interaction separates the tools used to manipulate objects from the objects themselves. Interaction instruments empower users by letting them compose their own environment with the tools they choose to use. Interaction instruments also support distributed, collaborative, multi-surface environments.

- Co-adaptation is the process by which users adapt to the tools they use as well as adapt the tools to their needs. Supporting co-adaptation requires to create systems that are flexible so that they can be appropriated by their users in ways that were not anticipated by their designers.

This PhD will study the particular area of software design and development of interactive systems. This area is particularly interesting because current tools to develop interactive software are particularly limited. Software development is also an area where command-line interfaces are still heavily used while direct manipulation tools are not widely adopted, raising the issue of combining the power of CLIs with the simplicity of GUIs. Software development is also a collaborative activity, from peer programming to large-scale distributed development groups. A rich set of supporting tools support these activities but they are poorly integrated.
Finally, this domain gives us an opportunity to bootstrap the design process by building the tools with themselves.

**Method:**
Creating better tools to support software designers and developers first requires a better understanding of their work practices and the strengths and weaknesses of current tools. This work thus begins with observation of real designers and developers in a variety of areas, including experts and novices or intermittent users.

These observations will be analyzed and a participatory design process will be developed, driven by the three conceptual frameworks outlined above: information substrates, instrumental interaction, co-adaptation.

This work will lead to the definition of Interaction-Driven Development, an approach where the design and development process of an interactive software system is centered on end-user interaction, rather than the back-end or functional core of the system. To this end, the following issues will be of particular interest:

- Bridging the gap between the tools used by designers, including graphical, visual and interaction designers, and those used by the programmers and developers;
- Combining command-line and graphical user interface in a coherent framework;
- Exploring the use of tangible interfaces for software development, and its integration with CLIs and GUIs;
- Understanding the mental models of programmers when debugging programs, especially event-driven programs where the flow of control is highly non-linear, in order to provide better debugging tools;
- Extending test-driven design to the development of interactive systems, where tests are notoriously difficult to automate.

In order to validate Interaction-Driven Development, a number of prototypes will be developed iteratively and tested in a variety of settings, using a range of evaluation techniques, from laboratory experiments to field studies.

**Expected Results:**
The results of this work will be a deeper understanding and a refinement of the three conceptual frameworks. The tools developed in the project will constitute proofs of concept in a real-world application area with considerable potential impact.

The results will be published in the best conferences and journals in the fields, including ACM CHI, ACM UIST, ACM CSCW, VL/HCC, ACM TOCHI.

**References:**