# Patterns for First Class Interactions: Introspection, Intercession, Instrumentation, and Composition

| | |
|---|---|
| **Duration** | 3 years |
| **Location** | Lille, France |
| **Team** | Loki — centre Inria de l'université de Lille & CRIStAL |
| **Contacts** | Damien Pollet (damien.pollet@inria.fr) |
| | Bruno Fruchard (bruno.fruchard@inria.fr) |
| | Stéphane Huot (stephane.huot@inria.fr) |

## Context

Because interactive systems are driven by events, they often become difficult to maintain as their code suffers from the so-called *spaghetti of callbacks* [6]. Graphical User Interface (GUI) toolkits help by providing reusable widgets, but at the cost of expressivity: they have limited features to build new widgets and interaction patterns, while the provided widgets are also tedious to customize (e.g., implement a long press on a button) and to bind together (e.g., record the travel duration between two widgets).

Interactive systems are inherently non-linear: they react to events, take part in feedback loops, and are structured around *causality* [5]. Modifying or composing interactions thus requires a model of *causal relationships*, which are implicit in callbacks. Several works have proposed to raise the abstraction level beyond the spaghetti of callbacks by structuring the control flow of interactions using state machines and dataflows. These approaches make it easier to engineer interactive applications: to understand, maintain, test, and compose them [1, 2, 3, 4].

## Objectives

The overall goal of this PhD thesis is to identify software engineering patterns and tools, based on and going beyond reified interactive state machines, that augment the malleability of interaction code, and enable the design of first-class composable interactions.

Seeing interactions as first class components impacts several levels in the design of an interactive system. For instance, one can *introspect* a state machine to provide feedback and feedforward. Reified state machines also provide *intercession* points to *instrument* or to *compose* interactions together, without multiplying callbacks. This changes the structure of the program, and likely impacts the tooling used to maintain it; it might help, for instance, to visualize the state machine while programming an interaction. Finally, it should facilitate designing interaction patterns and *composing* them to create combinations leading to novel interaction techniques.

In summary, this PhD thesis focuses on the software patterns that contribute to making interactive code malleable and composable. The goals are to:

- identify software patterns to design interactive systems using interactions as first class components,
- to apply these patterns to build instrumented systems for user experiments and evaluate their advantages and limitations,
- to evaluate how one can compose patterns to design and implements novel interaction techniques.

## Candidate

The PhD candidate will join the Loki research group, based in the Inria research center of the university of Lille and the CRIStAL laboratory. Lille is at the northern tip of France and its metropolitan area, situated at the crossroads of northern continental Europe, is the 5th biggest in France.

A successful candidate must hold a MSc or equivalent in Software Engineering or Human-Computer Interaction, and demonstrate a strong research interest. He or she should have experience or a strong interest in software development, and strong programming skills. Creativity, independence, team spirit, and communication skills are valuable assets. An excellent level of technical and scientific English is also required. The candidate will join a dynamic and multicultural team of young researchers at Inria. The Loki team members come from different backgrounds (Germany, Colombia, Canada, China, France) and communicate daily in English.

If interested in this position, email Damien Pollet (damien.pollet@inria.fr). All applications are welcome, regardless of age, gender, social or ethnic origin, sexual orientation, or disability. For the integration of people with disabilities, we are working on possible adaptations of the positions to be filled —within the limits of the applicable rules for the safety of people. Do not hesitate to contact us to tell us about your situation.

## References


[1] Caroline Appert and Michel Beaudouin-Lafon.
"SwingStates: Adding state machines to Java and the Swing toolkit."
In: *SoftwarePractice and Experience* 38.11 (2008), pp. 1149–1182. DOI: 10.1002/spe.867.

[2] Caroline Appert et al.
"FlowStates: Prototypage d'applications interactives avec des flots de données et des machines à états."
In: *33e*. Association for Computing Machinery, Oct. 2009, pp. 119–128. DOI: 10.1145/1629826.1629845.

[3] Arnaud Blouin and Jean-Marc Jézéquel. "Interacto: A Modern User Interaction Processing Model."
In: *IEEE Transactions on Software Engineering* (2021), pp. 1–20. DOI: 10.1109/TSE.2021.3083321.

[4] Mathieu Magnaudet et al.
"Djnn/Smala: a Conceptual Framework and a Language for Interaction-Oriented Programming."
In: *PACMHCI (Proceedings of the ACM on Human-Computer Interaction)* 2.EICS (2018), pp. 1–27.
DOI: 10.1145/3229094.

[5] Alice Martin, Mathieu Magnaudet, and Stéphane Chatty.
"Vers la complétude interactive: exigences pour une machine abstraite orientée interaction."
In: *32e conférence francophone sur l'Interaction Homme-Machine (IHM)*.
Association for Computing Machinery, Apr. 2021. DOI: 10.1145/3451148.3458644.

[6] Brad A. Myers. "Separating application code from toolkits: Eliminating the spaghetti of call-backs."
In: *UIST (symposium on user interface software and technology)*.
Association for Computing Machinery, Nov. 1991, pp. 211–220. DOI: 10.1145/120782.120805.