Computational Instruments: Concept-level Tools for Collaboration with Intelligent Interactive Systems

Daniel Buschek
LMU Munich
Munich, Germany
daniel.buschek@ifi.lmu.de

This paper appeared at the CHI '18 workshop *Rethinking Interaction: From instrumental interaction to human-computer partnerships*.
https://ex-situ.lri.fr/workshops/rethinkinginteraction-18

Abstract
I propose to leverage computational intelligence to elevate digital instruments to a more abstract role- and concept-level (think "builder" instead of "hammer"). Such *Computational Instruments* (CIs) reify roles of contributing intelligence to a task. To inspire these CIs, I first present a framework for structuring the distribution of roles in (creative) tasks among human and machine "thinking". By varying these role assignments, I generate design patterns giving rise to different kinds of CIs. I describe examples which I envision and discuss as reusable future tools to be employed and appropriated across currently common boundaries of applications, devices, and fixed control settings.

Author Keywords
Framework; Instrumental Interaction; Computational Interaction; Intelligent User Interfaces; Machine Learning

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction
In a world where intelligence permeates interactive artefacts, we might think of tools in terms of concepts and roles of contributing task intelligence, instead of action items (e.g. "builder" not "hammer"). After all, the smart machine

Step	Task/Role	Basic Question(s)
Divergence	Inspiration	What to work on?
	Proposition	How might we do it? Concrete ideas?
Iteration	Feedback	Where are we? Right direction?
	Refinement	What more to do?
Convergence	Evaluation	How good is it?
	Decision	What to accept?

Table 1: Framework overview. Each subtask/role might be given to human or machine (or a mix of both). By varying the assignments of these roles, the framework inspires different design patterns for *Computational Instruments*.

might as well figure out some of the nitty gritty details for us. These concept-level instruments still imply user initiative and are limited in their agency to their contributing role(s). Conceptualising this as *Computational Instruments* (CIs), I thus outline an intermediate perspective between HCI as tool use and as dialogue with agents or servants.

To inspire and arrive at such *Computational Instruments*, I first propose a framework to structure (creative) tasks in interactive systems into three steps with up to six conceptual subtasks and roles. Table 1 presents an overview.

These dimensions were inspired by a mix of ideas from related work, including 1) Norman's "Double Diamond Model of Design" [7] (human problem solving), 2) search and optimisation in Simon's "Sciences of the Artificial" [12] (machine problem solving), and 3) trends in recent applications of computational intelligence to HCI [8] and interactive media (e.g. [1, 6, 11, 13]). Following this, the roles mainly relate to *creative* tasks. I regard these as highly relevant and interesting for partnerships of users and intelligent instruments, since they go beyond simply automating repetitive to-dos and purely replacing user actions.

Computational Instruments (CIs)

I further describe the concept of CIs by relating it to others and outlining its integration into traditional GUIs.

Relationship to Instrumental Interaction
Similar to interaction instruments [2], CIs act as mediators between users and "objects of interest". However, they do not necessarily transform user actions into commands, but rather *reify* [4] *roles of contributing intelligence to a task*. Acting within such a role, they may replace some user actions ("tool use" / first person view [3]) with machine intelligence (servant / second person view [3]). Moreover, they may also ask for feedback from the user at certain points.

Relationship to Macros
One might be tempted to roughly think of CIs as "interactive intelligent macros". However, they go far beyond replaying a sequence of user actions, since they ideally offer 1) intelligent decision making and 2) collaboration with the user.

Integration into GUIs
In the spirit of macros, CIs could be imagined and integrated as sitting on top of a GUI, meaning that they operate some GUI elements for the user. This evokes *partnership* by highlighting the agency of CIs as equivalent to the user's, at least in the limited realm of the GUI. Moreover, this might be a practical approach to integrate CIs into existing software and to allow users to (somewhat) dynamically appropriate them across applications.

However, this form of integration is not the only possibility. CIs could also be seen simply as another feature and part of the "normal" GUI (cf. the "fix" button in *Sketchplorer* by Todi et al. [13]). Or they might be presented as a general toolbox independent of a specific application GUI, highlighting potential for appropriation across applications. Most of the following examples allude to this latter kind.

Example Patterns and CIs

I now use the framework in Table 1 as a design space: By varying the distribution of these roles between human and machine, the framework inspires design patterns for *Computational Instruments*. Note that the point here is to spark inspiration and discussion – and not to specify a comprehensive set of patterns or "definitive" role assignments.

Generator [Human: FRED, Machine: IP]
The system diverges, while the human (iterates and) converges. For example, imagine further developing a colour picker tool to provide textures. Cast as a *Generator* CI, it might "imagine" patterns from which users can choose.

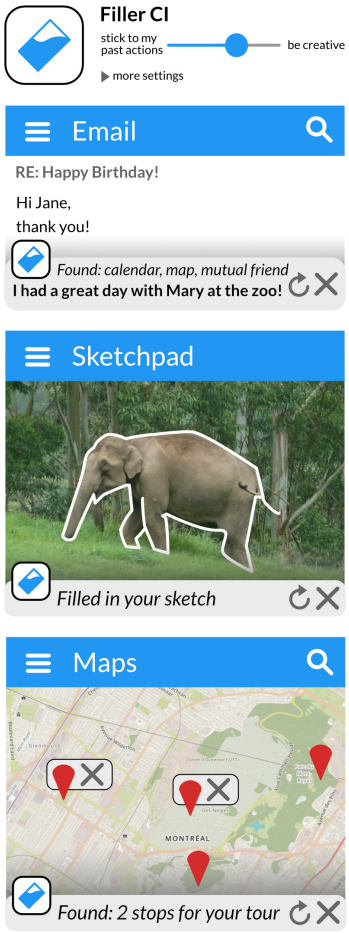

Figure 1: A *Filler* CI, representing the abstract concept and role of an *"auto-completer"*. From top: 1) The CI offers settings. 2) It is applied to an email draft. 3) After sketching, the user applies the CI to render an image. 4) The user employs the CI to fill in further stops for a tour.

Users might also give feedback to guide the generation towards currently favourable results, and/or refine patterns manually (e.g. after adding them to a canvas or 3D object).

Filler [Human: IFED, Machine: PR]
This is a variation of the *Generator* pattern, fed with an initial starting point by the user, as well as intermittent feedback. As an example CI, imagine a flexible *"auto-completer"* (Figure 1): Applied to an email (or parts of a draft), it could guess an appropriate completion or reply (cf. Gmail Smart Reply [6]). Used on a sketch, it might render an image (e.g. see [11]). On a tour map, it might fill in further stops. Focussing more on Proposition than Refinement, classic recommenders also fit this role (e.g. movie suggestions).

Filter [Human: IPR, Machine: FED]
Now the human diverges and the system converges. Imagine a *"quality spotter"* CI: The user might apply it to her camera app to get live feedback on scene composition (and maybe auto-triggered shots) during a holiday trip. She might later use the same CI on her stock images to find suitable media for her project work as a digital illustrator. She could further apply this CI directly to other machine output, for example the results page of a web image search.

Refiner [Human: IPED, Machine: FR]
Here, the human diverges, then the system iterates, before the human converges. The *Sketchplorer* tool by Todi et al. [13] presents an example: Users sketch layouts, while the system suggests refinements, with which users can continue working. As a flexible CI, envision a *"polisher"*: Applied to an email, it might highlight spelling mistakes and overly long sentences (and suggest improvements). On an article draft, it might further suggest formatting and layout images. For notes from a talk or meeting, it might highlight key points and to-dos. Handwritten (digital ink) notes could get optimised for readability or even corrected, too (see [1]).

Finaliser [Human: IF, Machine: PRED]
Here, the human only gives some inspiration and potentially feedback. This rather autonomous pattern is promising for situations where it is more important to get "something" than to determine exact details. Consider a *"disengager"* CI: Apply it to a music player app to generate and run a (context-based) playlist (cf. [5]). Or use it on a chat app to generate and set a status update (e.g. based on a user-provided keyword, context, calendar, etc.). Use this CI on a phone to prime it towards interpreting input in the spirit of "casual interaction" (cf. [10]) while relaxing on the couch.

Workshop Questions

The workshop [3] raised three questions on 1) flexible appropriation, 2) combining human and artificial intelligence, and 3) flexible degrees of control/automation. How do *Computational Instruments* relate to these questions?

Flexible Appropriation
I envision CIs to support appropriation, since the abstract nature of concept-level tools potentially renders them very flexible, for example for transfer and reuse across applications (see examples). However, it might be hard for users to anticipate the effects on specific objects. Moreover, it is also a great technical challenge to enable CIs to work in understandable and useful ways in unexpected contexts.

Combining Intelligence
CIs only replace *some* user actions related to a task: The design patterns and examples are based on the assumption of a partnership, in which both human and computer contribute intelligence in varying roles (Table 1). The resulting instruments are defined by their role and concept, not by specific action items. In summary, these concept-level instruments thus offer collaboration opportunities (and decision-making), moving beyond traditional tool use.

Flexible Degrees of Control
By varying role assignments (Table 1), different CIs imply different degrees of control (e.g. *Refiner* vs *Finaliser*). Fluidly shifting between levels of control thus manifests in the act of choosing CIs. Moreover, users flexibly shift degrees of control and automation by combining and chaining CIs. For example, a user might start with a *Generator* CI to get initial work done with the intention of manual continuation, yet then spontaneously decides to also use a *Filler* CI to automate further details as well.

Discussion

I conclude by discussing further aspects of the concept.

Anticipating Results
A main question is: Can users anticipate how a concept-level tool interacts with a given domain object? What might a "builder" create with a set of planks? This is far less obvious than judging what my own swing of a hammer would do. This issue is important for appropriation. To address it, we need to develop schemes of collaboration, feedforward/back, as well as self-revealing and explainable CIs (see explanation example in the email case in Figure 1).

Wielding Computational Instruments
The brief examples mostly refer to "applying" CIs, yet using them need not be limited to one-click triggers. Users might control and guide concept-level tools regarding, for example, 1) the point of application (e.g. apply *Refiner* only to selected content parts), 2) the instruments' extent of autonomy and variability (e.g. how close should a *Filler* stick to what's been established already; see Figure 1 top), or 3) the underlying assumptions (e.g. user demonstrates preferable propositions, refinements, or decisions). A main HCI question is how to realise these ideas as concrete interactions and GUI representations for CIs.

Learning and Adaptation
An integral part of the envisioned concept of CIs is that they leverage their computational nature to learn and grow with continued use and appropriation (e.g. via reinforcement learning), also across different domain objects. This presents an interesting transfer learning problem [9] in HCI.

Application Contexts
The given examples mostly address tasks on desktop computers or mobile devices. It will be interesting to think further about CIs in a larger context of ubiquitous computing.

Conclusion

I envisioned *Computational Instruments* – tools which use computational intelligence to work on a conceptual level, thus reifying roles of contributing intelligence to a task. To inspire CIs, I presented a framework with six (creative) task roles, assigned to either human or machine intelligence.

CIs support: 1) *appropriation*, by representing reoccurring abstract concepts; 2) *partnership*, by contributing to decision-making via collaboration in specific limited roles (Table 1); and 3) *varying degrees of control*, by allowing users to flexibly choose and chain CIs with different roles.

In contrast to full agents, CIs only replace *some* intelligent task roles. Like tools, they assume user initiative (and must "be used"). Thus, CIs offer an intermediate perspective on human-computer partnerships, sitting in-between HCI as tool use and HCI as dialogue with agents or servants.

I raised several questions for future work: 1) How can users anticipate what a CI might do to an object? 2) How can they control/guide CIs beyond one-click "magic sauce" implementations? 3) How can CIs be integrated into GUIs, in particular such that they unfold their potential for broad appropriation? 4) How can CIs learn/grow across such uses?

REFERENCES

1. Emre Aksan, Fabrizio Pece, and Otmar Hilliges. 2018. DeepWriting: Making Digital Ink Editable via Deep Generative Modeling. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA.
2. Michel Beaudouin-Lafon. 2000. Instrumental interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press. DOI:http://dx.doi.org/10.1145/332040.332473
3. Michel Beaudouin-Lafon. 2018. Rethinking Interaction: From instrumental interaction to human-computer partnerships. *Workshop Rethinking Interaction - CHI'18* (2018).
4. Michel Beaudouin-Lafon and Wendy E. Mackay. 2000. Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '00)*. ACM, New York, NY, USA, 102–109. DOI: http://dx.doi.org/10.1145/345513.345267
5. Daniel Boland, Ross McLachlan, and Roderick Murray-Smith. 2015. Engaging with Mobile Music Retrieval. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '15)*. ACM, New York, NY, USA, 484–493. DOI: http://dx.doi.org/10.1145/2785830.2785846
6. Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 955–964. DOI: http://dx.doi.org/10.1145/2939672.2939801
7. Don Norman. 2013. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books.
8. Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes (Eds.). 2018. *Computational Interaction*. Oxford University Press.
9. Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (oct 2010), 1345–1359. DOI:http://dx.doi.org/10.1109/tkde.2009.191
10. Henning Pohl and Roderick Murray-Smith. 2013. Focused and Casual Interactions: Allowing Users to Vary Their Level of Engagement. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2223–2232. DOI: http://dx.doi.org/10.1145/2470654.2481307
11. Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2017. Scribbler: Controlling Deep Image Synthesis With Sketch and Color. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
12. Herbert A. Simon. 1996. *The Sciences of the Artificial*. MIT press.
13. Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. ACM Press. DOI: http://dx.doi.org/10.1145/2901790.2901817