A Digital, Networked, Adaptive Toolset to Capture and Distribute Organisational Knowledge Speckle CI

Felix Deiters[1(✉)], Giovanni Betti[2], and Christoph Gengnagel[1]

[1] Department of Structural Design and Technology (KET), Berlin University of the Arts (UdK), Hardenbergstr. 33, 10623 Berlin, Germany
felixdeiters@udk-berlin.de

[2] Center for the Built Environment, University of California, Berkeley, CA, USA

Abstract. This work explores the potential of networked, composable tools as a means to capture and distribute organisational knowledge. To solve a design problem, teams will draw from a collective body of formalised, implicit, and tacit knowledge and synthesise it into a proposal. We argue that tooling can play a part in meeting new challenges for the discipline as well as broader trends in knowledge work.

Drawing from the concepts of *Version Control* and *Continuous Integration,* a widely adopted collaboration strategy in software development making heavy use of automation, this work proposes an automation platform for *AEC* workflows as a demonstrator of such a networked tool. The prototype is implemented using *Speckle*, a data platform for *AEC.* As a case study, three automation workflows are implemented and their potential to improve planning accuracy and capture and disseminate intra- and inter-organisational knowledge is reflected on. We argue this will ultimately require a renegotiation of the relationship between planners and their tools and we speculate about a culture of toolmaking as a means to capture and evolve organisational knowledge.

Keywords: Organisational knowledge · Continuous integration · Automation · Collaboration · Toolmaking

1 Introduction

Architecture, and the built environment in general, share the paradoxical quality of being both accelerator and key to many of the overlapping and mutually reinforcing crises unfolding around us. In order to unlock its potential to store carbon rather than emit greenhouse gases, preserve biodiversity rather than devour habitats, and foster community rather than divide and displace vulnerable demographics, planners need to radically transform the way we design, build, operate, and recycle our built environment. Materialising this new sustainable, robust, and resilient architecture calls for new levels of precision while facilitating deliberate experimentation, the integration of unfamiliar

C. Gengnagel et al. (Eds.): DMS 2022, *Towards Radical Regeneration*, pp. 36–45, 2023.
https://doi.org/10.1007/978-3-031-13249-0_4

and rapidly evolving areas of knowledge as well as the rediscovery and cultivation of old ones, and the navigation of unprecedented complexity in order to arrive at something simple (Fowles 2021).

This collective learning process is interacting with and accelerating broader trends that affect not only the architecture, engineering, and construction industry, *"AEC"*, but knowledge work in general: 1. The drive to increase efficiencies, 2. The emergence of new actors and business models, 3. The move away from bespoke services, and, finally, 4. The decomposition and 5. Routinisation of professional work (Susskind Susskind 2015).

This reorganisation of work, combined with the demand to rapidly create and integrate new knowledge calls for a new generation of highly specialised, networked tools that can be composed in vertically integrated (Fano and Davis 2020), fluid workflows. A design process that emphasises as result not only the built artefact but also the production of knowledge of how to improve future iterations of itself can leverage tool making as a means to store this knowledge.

To design means to integrate a plethora of factors, often contradictory, into a coherent whole. Design problems are *wicked* problems. They can not be solved linearly, but only iteratively (Stefanescu 2020). Consequently, design solutions, or partial solutions to a subset of the problem, are usually approximate solutions. This requires the implementation of analytics to continuously assess when a proposed solution nears and eventually passes a minimum threshold to be considered acceptable. To solve a design problem, teams will draw from a collective body of formalised, implicit, and tacit knowledge and synthesise it into a proposal. Because design and planning is a highly functionally differentiated discipline, individual learnings run the risk of remaining siloed, especially since collaborators are spread across organisations. Note that the term organisation can apply at various scales here: Firms can be understood to be made up of sufficiently differentiated sub-organisations, joint ventures in turn also form organisations themselves.

Tools, especially digital tools, encapsulate knowledge and make it accessible to a broader user base (Witt 2010). Tools have thus the potential to capture and distribute organisational knowledge. But tools also encroach on the agency of their user. This happens explicitly, for example when vendors of tools intentionally limit their interoperability, as well as implicitly through nudges (Thaler and Sunstein 2008). That's why these new tools need to possess certain characteristics: They need to be scrutable, hackable, and composable (Witt 2010).

2 Methodology

To demonstrate the potential of networked adaptive tools as a means of capturing and distributing organisational knowledge, a prototype automation platform is developed and tested following a trigger-action paradigm. As a case study, three automation workflows are implemented and their potential to improve planning accuracy and capture and disseminate organisational knowledge is reflected on.

2.1 Concepts

Speckle. The prototype is implemented using *Speckle,* a data platform for *AEC* that enables sending and receiving data between a variety of authoring applications. Speckle achieves this by providing plugins, "Connectors", that translate application specific model data to a common, highly malleable data format (Stefanescu 2020). Each update to a set of data is sent to a *Speckle* server, along with metadata such as the author, authoring application, and a message describing the change. This allows collaborators to understand the evolution of a digital model over time. *Speckle* organises data in streams, branches, and commits. A stream can hold one or more branches, with each branch being defined by a series of updates to the data, called "commits".

Version Control. These features closely mirror those of *Version Control Systems* used in software development. *Version Control Systems* allow multiple collaborators to make and track changes to a joint codebase, which will usually be hosted on a private server or popular platforms such as *Github.com* or *Gitlab.com.*

Continuous Integration. To minimise the risk and impact of merge conflicts, when conflicting changes are introduced at the same time, updates should be incremental and frequent. Modern software development practices leverage automation to facilitate the *Continuous Integration* of changes. For example, an automation will check whether new code adheres to formatting standards before allowing it to be merged, or it will run a suite of integration tests to ensure no functionality was accidentally broken with the update.

Continuous Integration is often used alongside *Continuous Delivery* or *Continuous Deployment* which means to continuously compile code into a piece of software, and to distribute, and / or deploy it automatically. There is substantial overlap between the two practices which is why one will frequently find both simply referred to as *CI/CD*.

A *CI* system will allow to define a set of automations and allow those to be triggered at specific events, such as when new code is to be integrated. Automations will be composed of individual actions that are executed sequentially or concurrently (Fig. 1). The terminology differs between *CI* platforms, we will use the terms "workflow" and "action" from here on. *CI* platforms will provide a set of actions, but also allow the development of new ones. These can be made available to others, effectively sharing the knowledge encapsulated in them.

Workflows and triggers will usually be declared by adding configuration files to the code repository. The fact that workflows can be composed from existing actions and declared in a simple text file makes it very easy to incrementally adopt *CI/CD* practices and it is common for programmers to use them even in small personal projects.

2.2 Speckle CI

We argue that the simplicity of the trigger-action paradigm, together with its incremental adaptability make a *CI*-inspired automation platform an ideal candidate to demonstrate the potential of networked tools in *AEC*. Built on top of *Speckle*, being "polyglot" and itself incrementally adoptable, it constitutes, in our view, a sort of "Goldilocks path" towards a future of networked, composable tools.

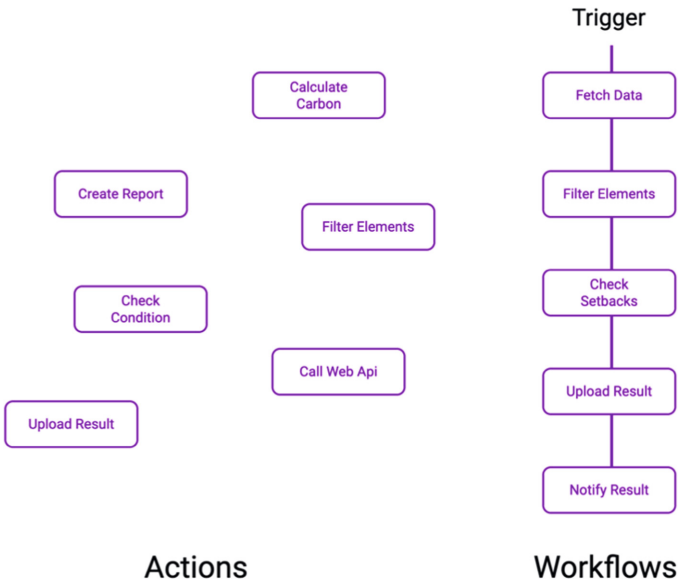

Fig. 1. *Continuous Integration* systems allow to define automations by composing actions into workflows, which are then triggered by specific events.

While *Speckle CI* draws inspiration from established *CI* platforms, key aspects were adapted to *AEC* workflows and practitioners. Based on the theory that the adoption of the automation of higher-level design tasks is often hindered by a lacking user experience (Heumann and Davis 2019), a particular emphasis was put on making the user interface as easy to use as possible. For example, the creation of workflows is done entirely via the user interface as opposed to configuration files in other solutions. While those can encode very sophisticated workflows, they might be difficult to use for practitioners unfamiliar with scripting or programming. After weighing the familiarity of visual scripting editors such as *Grasshopper* or *Dynamo*, a user interface paradigm inspired by the iOS *Shortcuts* app was selected for its simplicity (Fig. 2).

Upon login, a user will be presented with a list of the workflows they have previously set up (Fig. 3). Workflows will be sorted by the time they were last triggered and will be displayed along additional information such as whether the last run failed or succeeded, and the option to edit a workflow or create a new one.

Workflows are created or updated in the workflow editor. To keep the mental model as lean as possible, *Speckle CI* matches *Speckle's* ontology with a stream as the top hierarchy. That's why each workflow is assigned to exactly one stream, while a stream can have multiple workflows configured for it.

The editor is organised in three sections: Triggers, conditions, and the actions to be performed. After specifying a name and *Speckle* stream, users are presented with the choice to chose specific triggers on that stream, such as when a new commit is created or deleted, and conditions that will restrict a workflow from being triggered unless they are met. For example, a user could set a workflow for *Stream A* to be triggered when a new

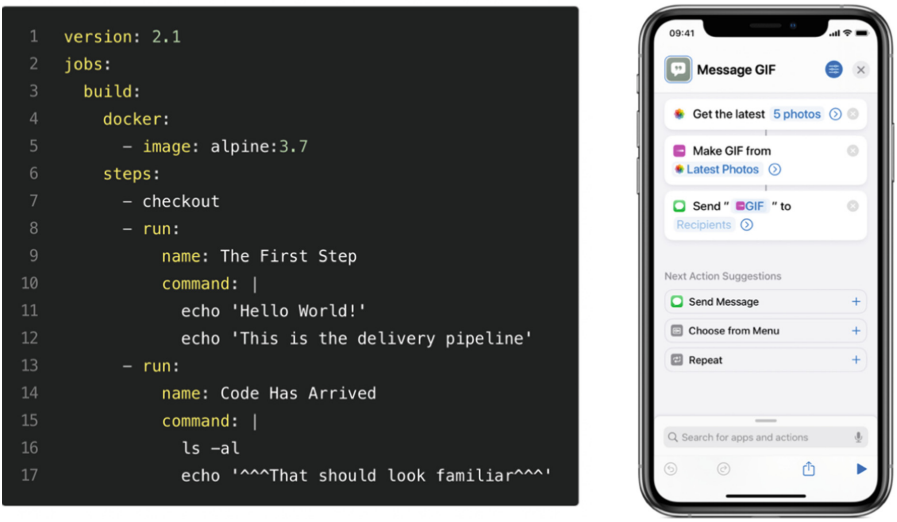

Fig. 2. Configuration of a workflow in the *CircleCI* platform. (left) compared to the simpler interface of the *Shortcuts* app on *iOS* (right) © Circle Internet Services, Inc; Apple Inc.

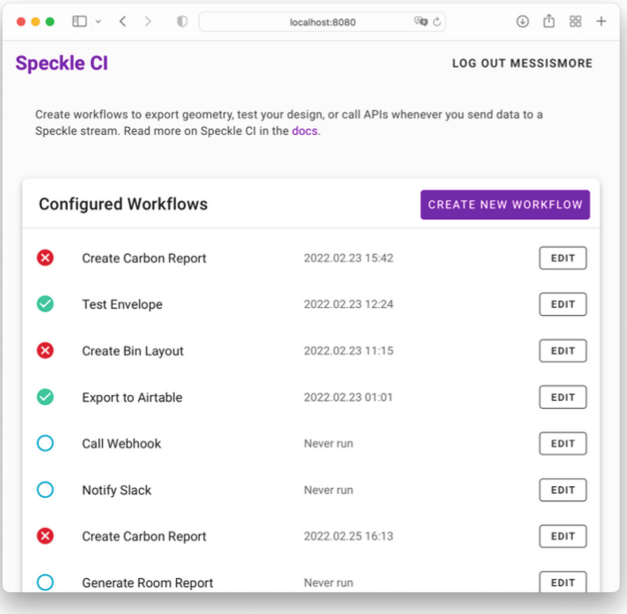

Fig. 3. The main view of the web app lists all configured workflows at a glance. Icons communicate the success or failure of the most recent run of a workflow.

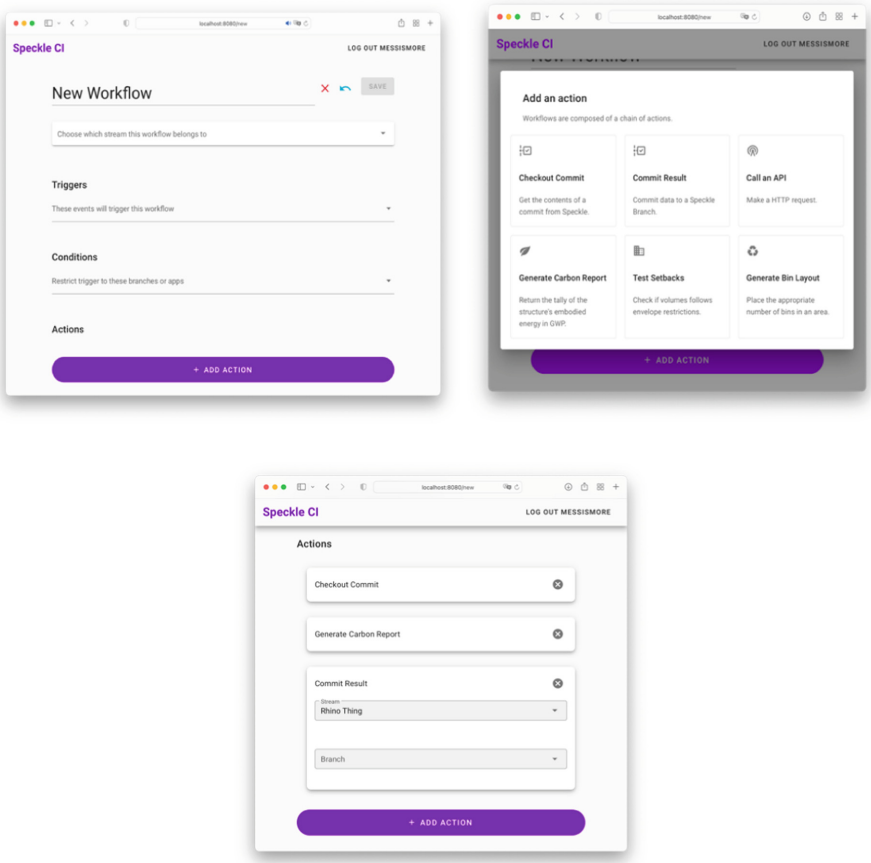

Fig. 4. Flow to create a new workflow. Actions are selected from the *Actions Store* and may offer additional settings.

commit is created, updated, or deleted, but only when the commit is made to *Branch B* and authored by *Application C*.

A prominent button opens the *Actions Store*, where the user can pick the individual actions to compose the workflow from (Fig. 4). Actions will appear in the editor as a sequence of blocks that may offer additional inputs and configuration options. For example, the action to commit data to Speckle allows the user to specify a branch.

2.3 Case Studies

Three workflows were implemented to explore the potential of this approach.

1. Carbon Calculator
2. Setback Checker
3. Bin Layout Tool

They were selected because, in the authors' view, they are representative of the kind of design challenges that could benefit from such an approach, as well as because they demonstrate the platform's capabilities to generate numerical (carbon accounting reports), spatial and geometric (setback checker), or hybrid (bin layout tool) results.

Each workflow's setup, triggering, and results were documented in videos, to serve as a reference for future user testing.

Carbon Calculator. Quantifying and optimising the environmental impact of the built environment constitutes a relatively new area of expertise for planners. A project's environmental impact should be considered from the earliest design stages (Apellániz 2021). To test this workflow, a minimal carbon calculator action was developed. Being a prototype, it applies the rather simplistic approach of matching a model's specified materials to their embodied carbon and returning the tally. While this provides only a heuristic and does not eliminate the need for more accurate modelling by domain experts, continuous, immediate feedback about how design changes impact the performance of a proposed building can help instil an intuitive understanding of how measures relate to outcomes and encourages precise modelling practices from the get-go.

Setback Checker. A big part of the morphology of the built environment is formed by building codes. To translate the rules encoded in law into software means to capture the knowledge necessary to adhere to them. Code compliance becomes thus to an extent automatically testable. While certainly not every part of the building code can be captured and tested in this way, we speculate that this can help prevent planning mistakes, direct planners' attention to critical areas, and thus increase planning precision. A simple, visual example of such an "algorithmic" legislation are the setback rules specified in Berlin's building code. We developed an action that, given a plot's perimeter, a building volume, and adjacent streets and buildings, will return whether or not it violates envelope restrictions (Fig. 5).

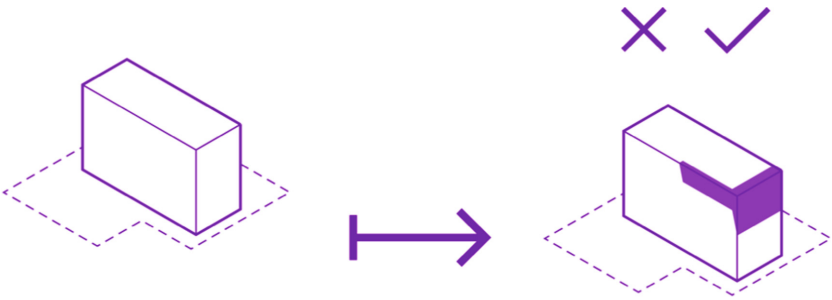

Fig. 5. The setback checker action tests whether a given volume violates envelope restrictions.

Bin Layout Tool. Planning involves many mundane, tedious, and time-consuming tasks that nevertheless require practitioners to draw from their experience. Organisations will usually develop informal best practices and conventions, and these can often be encoded

(Heumann and Davis 2019). One such example is the configuration of bins for residential buildings, because it combines hard rules, such as the required number of bins per unit, with soft, empirical rules, such as their preferred layout. As a third case study, such a tool was made available as an action in *Speckle CI*. Provided with an area and the number of residential units, the action will return whether the required number of bins will fit, and, if so, a generated layout.

3 Results

The case studies show that the abstractions and paradigms of *Continuous Integration* lend themselves well to *AEC* workflows, although further validation through user testing and the application in a real-world test-case are still to be carried out. The three case studies are, in the authors' view, representative of a whole range of domains where the application of cloud-based automation can improve planning precision.

To apply the practice of *Continuous Integration* to *AEC*, some aspects of the implementation need to be adapted. The model of a workflow was simplified and all functionality was made available via a graphical user interface. We found that the simplicity of the interface paradigm did not hinder the implementation of our case studies.

While the development of new actions requires some programming expertise and an understanding of *Speckle's* data model, the abstraction of composing actions into workflows allows for the creation and adaptation of automations by non-expert users.

Because *Speckle CI* is a web app rather than desktop software, initial setup is easy as no installation is required. This makes *Speckle CI* more approachable compared to other tools such as *Grasshopper* that are sometimes used to achieve similar goals. Leveraging *Speckle's* authentication mechanisms lowers the barrier further because users can simply start to use *Speckle CI* with their existing *Speckle* accounts.

Since *Speckle CI* is a web application, collaboration is also facilitated. Running automations locally on each staff member's computer would require their execution environment to be configured identically. This synchronisation is tedious and error prone. With *Speckle CI*, workflows are instead executed on a web server, a kind of "cloud" that contains the authoritative configuration and delivers reproducible results.

Speckle's wide range of available connectors to other software make it incrementally adoptable. Accordingly, existing workflows can be incrementally moved to *Speckle CI*. This constitutes a key advantage, as the adoption of automation tools is often hindered by their failure to integrate into existing design workflows (Heumann and Davis 2019).

4 Discussion and Next Steps

The method ultimately requires a renegotiation of the relationship between architects and their tools. Firms will usually employ teams of computational design specialists to solve specific problems for one-off projects many times over, even if these problems are very similar in nature. A platform such as *Speckle CI* allows them to shift their roles towards a new kind of toolmaker. It is positioned to herald a new culture of toolmaking as a means of capturing and developing organisational knowledge.

Developing actions, composable blocks of functionality that can be integrated into the design process by the members of other teams, allows their expertise to be more effectively disseminated in the organisation, and, through the adoption of open source methodologies, even throughout the discipline as a whole. Newly developed actions can be immediately deployed to a whole range of projects. Their effort scales.

Because their work can be more effectively reused, they can now better justify applying engineering practices to toolmaking (Davis 2013), applying lessons from design process iterations to improve functionality, thereby improving future iterations of the design process.

Tool and process share a reciprocal relationship. Actions developed to address a specific project need become available in future projects to be deployed and built upon. Their inclusion in a firm's toolkit will then represent part of its organisational design expertise and therefore make the application of this particular approach in future projects more likely, shaping the organisation's design culture.

The composition of actions into workflows can be regarded as a form of toolmaking, too. Hereby, too, is knowledge encoded, although to a lesser extent than by the development of new actions. A mechanism to share workflows between projects and users should be assessed going forward, as this will make the knowledge captured in them more easily accessible and also improve the user experience of the platform. Such a shared workflow would be scrutable, and, more importantly, adaptable by other users.

Speckle CI should be regarded as a step towards a future of networked tools and as an enabler of a new culture of toolmaking. While this work focussed on the appropriateness and applicability of *Continuous Integration* to *AEC* workflows, we suggest that the obvious next steps will be to perform user-testing, expand the set of actions, investigate a solid sharing mechanism for workflows, and apply the findings of recent research to build a more flexible and powerful backend to orchestrate the execution of workflows (Wanderley Barbosa 2022).

References

Apellániz, D.: A holistic and parametric approach for life cycle assessment in the early design stages. In: Symposium on Simulation for Architecture and Urban Design SimAUD (2021)

Davis, D.: Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture. RMIT University, Melbourne (2013)

Fano, D., Davis, D.: New Models of Building: The Business of Technology. In: Shelden, D. (ed.) Architectural Design 90(2), pp. 32–39. John Wiley & Sons Inc, Hoboken, NJ (2020)

Fowles, E.: Make low-tech our mantra. In: RIBA journal: the journal of the Royal Institute of British Architects, August 2021, pp. 36–40. RIBA, London (2021)

Heumann, A., Davis, D.: Humanizing architectural automation: a case study in office layouts. In: Gengnagel, C., et al., (eds.) Proceedings of the Design Modelling Symposium, Berlin 2019. Springer, Cham, Switzerland (2019)

Stefanescu, D.: Alternate Means of Digital Design Communication. UCL, London (2020)

Susskind, R., Susskind, D.: The Future of Professions. How Technology Will Transform the Work of Human Experts. Oxford University Press, Oxford (2015)

Thaler, R.H., Sunstein, C.R.: Nudge. Improving Decisions About Health, Wealth, and Happiness. Yale University Press, New Haven & London (2008)

Wanderley Barbosa, V.: Computational Design Workflows Orchestration Framework. DTU, Kongens Lyngby (2022)

Witt, A.J.: A machine epistemology in architecture. Encapsulated knowledge and the instrumentation of design. In: Sowa, A., et al., (eds.) Candide. Journal for Architectural Knowledge no. 03 (December), pp. 37–88. Hantje Cantz, Ostfildern (2010)