

# An interactive agent-based framework for materialization-informed architectural design

Abel Groenewolt<sup>1</sup>  · Tobias Schwinn<sup>1</sup> ·  
Long Nguyen<sup>1</sup> · Achim Menges<sup>1</sup>

Received: 18 May 2017 / Accepted: 17 November 2017 / Published online: 4 December 2017  
© Springer Science+Business Media, LLC, part of Springer Nature 2017

**Abstract** Concepts of swarm intelligence are becoming increasingly relevant in the field of architectural design. An example is the use of agent-based modeling and simulation methods, which can help manage the complexity of building designs that feature many similar, but geometrically unique elements. Apart from leading to effective solutions and expanding the architectural design space, agent-based design methods can also be employed in integrated planning processes, in which the contributions of various disciplines take place in an integrated loop instead of being executed consecutively. We propose a computational framework for architectural design, in which agents represent building elements and/or joints between building elements. Behavior parameters, behavior weighting, and the environment can be modified in real-time while the agent system is running. Additionally, the designer can interact with individual agents directly, while slowing down or pausing agent movement if so desired. In the resulting design approach, the designer can globally adjust behavior parameters, while retaining local control over details where needed. To facilitate an integrative design process, domain-specific data and the results of external analysis can be included, either directly as input for agent behaviors, or by modifying the environment. We illustrate the potential of this computational framework using the example of the design of plate structures and show how this method can lead to quantifiable results while also attaining aesthetic goals. Furthermore, we provide an outlook toward possible further extensions of agent-based design methods in architecture.

**Keywords** Building design · Agent-based design · Multi-agent systems · Emergence · Polyhedral plate structures

---

✉ Abel Groenewolt  
abel.groenewolt@icd.uni-stuttgart.de

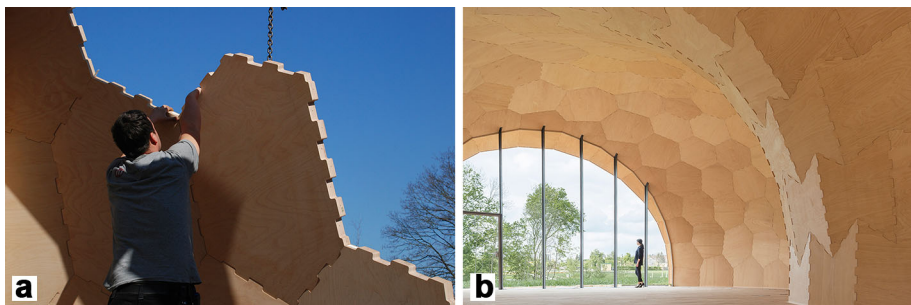
<sup>1</sup> Institute for Computational Design and Construction, Universität Stuttgart,  
Keplerstrasse 11, 70174 Stuttgart, Germany

# 1 Introduction

Understanding principles of nature has been identified as an important skill for architects since the very origins of architecture theory (Pollio 1914; Laugier 1753). The idea of studying systems occurring in nature in order to be able to apply them in architecture and engineering has been reinvented in the second half of the twentieth century (Otto et al. 1982; Wester 1989). With advances in computational design and digital fabrication (e.g., through the use of industrial robots) (Krieg and Menges 2013), the study of complex systems that exhibit emergent behavior is becoming increasingly relevant for the field of architecture (Ball 2012; Vincent 2009). Through the development of computational methods that are inspired by biological systems (such as evolutionary design methods or agent-based modeling and simulation), new approaches to architectural design open up (Menges 2012). The ability of complex adaptive systems (such as swarm systems) to adapt to varying conditions and accommodate change (Macal and North 2005) makes agent-based modeling a particularly suitable method for design integration and design interaction within a rule-based system. Properties of agent systems that are relevant in the context of architectural design include self-organization, adaptation, emergence, and the capacity to negotiate conflicting goals within one system.

Methods such as particle swarm optimization (PSO) (Kennedy and Eberhart 1995) and ant colony optimization (ACO) (Dorigo et al. 1996) aim to find an objective global minimum or maximum value within the parameter space of a function. Design tasks often involve negotiating large numbers of requirements, which suggests that PSO and ACO could be successfully employed. As demonstrated by Puusepp (2014), ACO can be effectively used for subtasks of design, such as circulation planning. However, as subjective decisions (including aesthetic judgment) often play an important role in design processes, design processes cannot entirely be carried out by optimization only (Takagi 2001). In early design stages, objective functions can usually not yet be defined and the parameter space tends to change while the design process progresses. Therefore, instead of trying to automate and optimize design tasks, we focus on the development of interactive tools that assist a designer during the design process.

In previous work, an agent-based design approach was used for the design of a building featuring a construction made of many similar but geometrically unique timber plates: the Landesgartenschau Exhibition Hall in Schwäbisch Gmünd, Germany (Fig. 1). This research showed that agent-based modeling is a viable computational method for integrating geometric constraints and fabrication constraints at the building element level (Schwinn et al. 2014).



**Fig. 1** Landesgartenschau Exhibition Hall: assembly process (a) and interior (b). The geometrically unique, robotically fabricated timber plates are designed using an agent-based design approach and form the load-bearing structure of the building. Images from Krieg et al. (2015).

In this paper, we focus on subsequent developments in our research on agent-based modeling and present a framework for agent-based architectural design and planning that integrates a large range of planning aspects (such as structural analysis and life cycle analysis, as well as prefabrication planning). It enables a design feedback loop, in such a way that steps that are normally taken sequentially are integrated with earlier phases of the planning process. Our implementation allows user interaction while the agent system is running, thus creating an interactive design environment. We demonstrate this environment with a case study of polyhedral plate structure design.

## 1.1 Motivation

The research described in this paper is part of an ongoing effort to develop design approaches, structural systems, and building systems, as well as their associated fabrication and construction procedures, that are more performative than conventional approaches. By allowing the exploration of building designs that lie outside established architectural and structural typologies, novel design approaches and building systems could help answer challenges that the building sector is facing: challenges that range from the large environmental impact of the construction and operation of buildings (McMullen and Jabbour 2009) to inefficient design and construction processes (Construction Users Roundtable 2004).

In previous research, the authors have established the viability of an agent-based design approach for material- and fabrication-informed design exploration (Baharlou and Menges 2013) and presented an agent-based design approach for robotically fabricated segmented timber shells (Schwinn et al. 2014; Schwinn and Menges 2015). Based on these findings, our premise is that agent-based design is a viable approach for the development of more integrated design processes, which in turn could lead to flexible and efficient solutions that make use of the possibilities of digital fabrication.

Our overarching hypothesis is that the use of agent-based methods has the potential not only to expand the architectural solution space, but also to profoundly impact the nature of the processes of building design and construction.

## 1.2 Aim

Our aim is to further develop and investigate interactive, integrative building design processes using agent-based methods. The integrative agent-based design process that we envision would need to fulfill the following requirements:

1. Integrate design and analysis steps in a single design phase.
2. Help manage the geometric complexity of unique and interdependent building parts.
3. Combine self-organizing systems with interactive design input.

A prerequisite for any integrated planning approach is that requirements that are normally provided by engineers and other consultants during various phases of the planning process can be abstracted early on. Our aim is to integrate such planning aspects (e.g., structural analysis, life cycle analysis, and prefabrication planning) in early design stages in a single feedback loop. We propose an agent-based design approach not only as a method to generate variation of design options and differentiation of building, but also as a method for accommodating changes and for reformulating the solution space during the sequential development process of architectural design, which potentially reduces the complexity and cost of design changes.

Instead of a designer defining the geometry of a design directly, we envision a design process in which the designer defines and controls agent behaviors (which we define as the response of an agent to internal and/or external stimuli based on internal rules), in such a way

that the resulting self-organizing and/or emergent behavior of the agent system (in which numerous agents interact) generates the design output. The definition of ‘behavior’ is based on Levitis et al. (2009), and for ‘emergent’ and ‘self-organizing,’ we follow the definitions given by De Wolf and Holvoet (2004).

Additionally, as the exact outcome of agent-based processes can be hard to predict due to nonlinearity of agent interactions, it should be possible to interfere and interact with agents in the system directly. Details of our proposed implementation are described in Sect. 3.

### 1.3 Scope

This research is focused on architectural design methods that use affordances of integrative design processes and digital fabrication. Building on concepts and methods from swarm intelligence and computer science [in particular agent-oriented programming (Wooldridge 1997)], the research addresses questions related to design exploration, design integration, and optimization.

As opposed to conventional ways to employ software in the design process, our main concern is not just an increase in speed or efficiency. While the resulting design may perform very well in metrics such as structural efficiency, material use, energy use, and construction costs, part of what we aim for is the ability to explore a larger part of the potential solution space of building design. This would result in the ability to design and build buildings that could not be realized with conventional design and construction methods.

## 2 Context

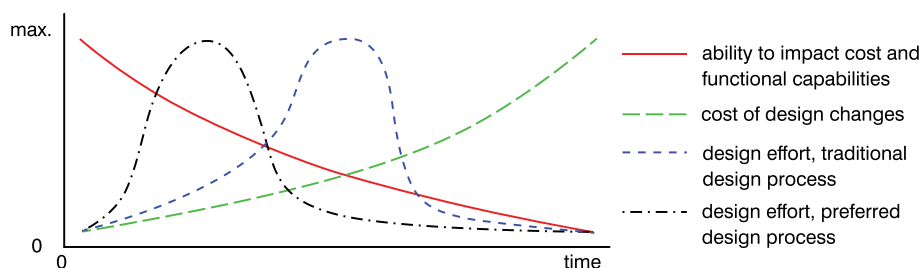
### 2.1 Construction processes

Building construction is a process that typically consists of combining many elements into a larger whole. The way elements are combined plays a vital role in how well a building performs on metrics ranging from structural performance (where the way elements are locally put together has an influence on global performance) to spatial quality and even social and cultural relevance. Deciding how to put building parts together is thus vitally important in making sure the resulting building is more than the sum of its parts.

Building elements interface with their neighbors through joints and connections, in ways that are informed by material properties and construction processes. From the point of view of agent-based design, this provides opportunities to define local rule sets, which can be utilized in a bottom-up approach for negotiating the shape and location of these elements within the larger context of an assembly according to well-defined performance criteria (Schwinn et al. 2014).

### 2.2 Building design processes

Typically, architectural design is a process in which over time, decisions are made at an increasing level of detail: In initial design stages, design solutions on a detail level may be entirely undefined. While parts of the architectural design process are iterative, the overall process is sequential and certain design tasks (such as structural design or energy performance analysis) tend to be carried out after major design decisions have already been taken. Prefabrication and construction planning take place even later in the project.



**Fig. 2** Improved building design process as proposed by MacLeamy (Construction Users Roundtable 2004): By concentrating design effort earlier in the process, design changes can be made more effectively and with lower costs. Figure redrawn after Construction Users Roundtable (2004)

Making major changes to a design in later stages of the design process (or even during the fabrication and construction phases) tends to result in increasing amounts of additional design work and associated costs (Construction Users Roundtable 2004; see Fig. 2). Consequently, there are clear economic incentives to integrate steps that normally take place later in the process in an earlier stage in the design process. However, just making decisions earlier in the process does not guarantee that these decisions do not need to be revised later on. Thus, the relevance of finding ways to reduce the complexity and associated costs of revising design decisions is independent of the phase in which design decisions are made.

Parametric design has been proposed as a way to reduce the costs of changes later in the process by maintaining a “live” model that is capable of accommodating changes (Roller 1991). While parametric design is a tried-and-tested approach for generating design variations within the solution space of a specific parametric model, changes to the parametric model itself may be difficult to implement. During changes to a parametric model (e.g., in the case of modifying the topology in a visual programming environment), the model will be dysfunctional and a significant amount of time may need to be invested to make it operational again. The limited capacity to accommodate changes that require the reformulation of the design solution space limits the suitability of parametric design for design exploration.

### 2.3 Swarm intelligence

Agent-based modeling and simulation (ABMS) is closely related to research into emergence and self-organization in natural and artificial systems consisting of social agents within the field of complex adaptive systems (CAS). Systems such as flocks of birds, schools of fish, colonies of ants, termites, bees or wasps, but also crowd stampedes, traffic, and stock markets exhibit a higher order and repeating patterns even though none of the involved entities have an overview of the global system. ABMS provides methods to study and to gain insights into such systems, which due to the number of nonlinearly interacting entities and lack of central control are difficult to study using alternative modeling and simulation methods (such as discrete event simulation, or systems of differential equations as in system dynamics).

The ability of complex adaptive systems in nature to solve problems such as effective foraging for food, predator evading, or colony re-location through cooperation of multiple individual agents has inspired a number of approaches used in engineering that involve multiple computational agents. In these approaches, which are usually grouped under the term ‘swarm intelligence,’ the aim is to harness the ability of complex adaptive systems to solve problems that are difficult to solve using alternative approaches, especially finding global

optima in non-convex optimization problems. As Glover and Kochenberger (2003) put it, such meta-heuristic approaches “orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space.” Meta-heuristic approaches include the swarm-inspired stochastic optimization method PSO as well as ACO and other population-based search algorithms, such as genetic algorithms (GA), all of which utilize metaphors from natural systems. In PSO, each particle represents a candidate solution to the optimization problem and, using the flocking metaphor, its speed and direction of travel across the continuous search space are influenced by the best solution of its immediate neighbors as well as its own personal best solution that was previously achieved (Brownlee 2011). ACO builds on the metaphor of ants using pheromone trails to find the shortest or ‘minimum cost’ path between a food source and the nest and is used when the optimization problem can be transformed into the problem of finding the best path on a weighted graph (Dorigo 2007).

Swarm-based optimization is based on the concept of agents, that is individual entities that have a degree of autonomy, but compared to ABMS, there are important differences in the underlying assumptions and consequently in the use cases. In swarm-based optimization, the practical focus is on harnessing the “power of the many” in order to converge on a solution, whereas in ABMS the focus is on the investigation of the dynamics of a complex system in order to provide insight into the mechanisms that lead to observed macro-level, emergent behaviors. In applications of swarm optimization, systems are usually composed of homogeneous agents with static rules, whereas in ABMS the systems are usually heterogeneous with agents that are able to have different and evolving attributes and behaviors at different times and in different contexts (Brownlee 2011; Bonabeau 2002).

The proposed application of agents in our system essentially draws from ABMS as well as swarm intelligence: On the one hand, we are interested in the dynamics of the system of interacting agents and in exploring the micro-level behaviors that lead to a meaningful macro-level configuration; on the other hand, we are searching for steady states in the system. This strongly contrasts with natural CAS, which do not achieve a steady state unless they are dead (Forrest and Mitchell 2016). In effect, we exploit the dynamics of the system to be able to converge to different states, not focusing on optimization but rather on the ability of complex systems to adapt and self-organize, taking into account changing constraints and user input. This mode of operation engenders a novel ‘exploratory modelling’ approach, which is highly relevant in architectural design: In this approach, hypotheses can be systematically tested by designing micro-level rules and by implementing functional principles, in order to gain insights into macro-level effects.

In contrast to social or socio-technical systems, the proposed approach is not limited by the challenge posed by homomorphisms (that is how well a model describes the real system that is being modeled) and the corresponding questions concerning the validity of the conceptual model, the validity of the results of the agent-based simulations, and the usability of the model [generator—mediator—predictor; see Heath et al. (2009)]. Instead of validating results by matching real-world data gathered in the field (as, e.g., in computational social sciences or in ecology), in our case the conceptual model and agent system output are validated through expert opinion.

## 2.4 Agent systems in architecture

As the roots of some core concepts of swarm intelligence stem from insects carrying out construction processes (Garnier et al. 2007), it can come as no surprise that this topic has been

explored by architects in various projects. For example, Scheurer (2007) discusses the use of agent systems for structural optimization, Tamke et al. (2010) use self-organization principles for the design of timber structures, and Vasey et al. (2015) show an application of agent-based methods for design and fabrication of fiber composite structures. Helbing et al. (2005) use simulation of human movement to simulate pedestrian flow and building egress, and Leach (2009) promotes the use of agent-based methods for urban design exploration. Baharlou and Menges (2013, 2015) discuss the integration of aspects such as material and fabrication in behavior-based design systems. An overview of further examples of the application of agent-based methods in the design and construction context can be found in Gerber et al. (2017).

The role that agents take in various agent-based architectural design approaches varies. In Zeng et al. (2007); von Mammen and Jacob (2008), agents represent builders that deposit material. In Taron and Parker (2014) and in some of the work of Snooks (2012a, b), the interaction between agents over time is expressed in the materialized shape. In Anumba et al. (2002), agents represent the roles of various actors in a multi-disciplinary design team, whereas in one of the projects of Puusepp (2014), agents represent buildings. In the work presented in this paper, agents represent building components and joints.

As discussed by Macal (2016), agent-based models of this kind fall within the category of interactive agent-based modeling and simulation, despite technically not being simulations as they do not model phenomena occurring in the real world. Instead of using the term “running a simulation,” we will describe the process of agents updating their state as a result of behaviors with the term “running the agent system.”

In early design phases, the agent-based model (ABM) that we propose serves as an exploratory generator that facilitates testing different design hypotheses, potentially generating unforeseen and novel outcomes. Throughout the design process, the conceptual model underlying the ABM is iteratively refined and becomes an increasingly accurate abstraction of design intent and building system, with the outcome of the ABM continuously adapting to constraints and design decisions. Finally, the building design (as the outcome of the ABM) is collaboratively validated using expert opinion. Even in this late stage in the building design, constraints can be updated, for example, to evaluate the impact of choosing alternative fabrication strategies.

## 2.5 Interactive agent systems

A variety of multi-agent system environments exists (Nikolai and Madey 2009; Eiter and Mascardi 2002), but such systems are often not easily usable in the field of architecture (Beetz et al. 2004). However, as successfully demonstrated by Gerber et al. (2017), external multi-agent systems can be linked to design software, in such a way that core geometric functions of the design software can be used. In our implementation, the core agent functionality is implemented as a plug-in that runs within design software; details about this implementation can be found in Sect. 3.2.

Interactivity and subjective user evaluation are a vital component in many applications of design-oriented evolutionary computation (Takagi 2001). Most agent systems are designed to be autonomous, and many frameworks do not allow for user interaction (Eiter and Mascardi 2002), other than allowing the user or programmer to set certain parameters, as in Gerber et al. (2017). Interactive agent systems have been proposed for game design (Reynolds 2000) and have been used in art installations (Arndt et al. 2000; Boyd et al. 2004). In the latter, agents react to the physical presence of humans, whereas in the former, the user can interact with a group of agents by controlling an avatar to which the agents in the system react.

### 3 Methods

#### 3.1 Agent-based framework

The consolidation of ABM into an accepted modeling and simulation (M&S) approach (in addition to DES and SD) that happened in various fields in the late 1990s and 2000s was not only facilitated by but also predicated upon the development of commonly used software frameworks (Heath et al. 2009). Instead of scientists developing idiosyncratic software and duplicating efforts, the goal of software frameworks is to allow comparison and repeatability of results by separating the development of the framework core from the modeling of the experiment (Minar et al. 1996). As demonstrated by the variety of implementations listed in Sect. 2.4, such a consolidation (which could make ABM a generally accepted modeling approach next to modeling approaches such as parametric modeling) has so far not happened in the field of architectural design.

The purpose of the proposed framework is therefore to provide a structured environment for experimentation with agents and behaviors, a ‘scientific apparatus’ that facilitates comparison and replication of results. In our system, agents represent building parts or connections between building parts; together, the agents form an agent system that represents a building design. The development of this framework is an ongoing interdisciplinary collaboration between architects, engineers, and computer scientists. The framework is based on the agent system that was developed for the Landesgartenschau Exhibition Hall (Schwinn et al. 2014), which in turn is derived from systems described by Shiffman et al. (2012) and Reynolds (1987).

##### 3.1.1 Scope of application

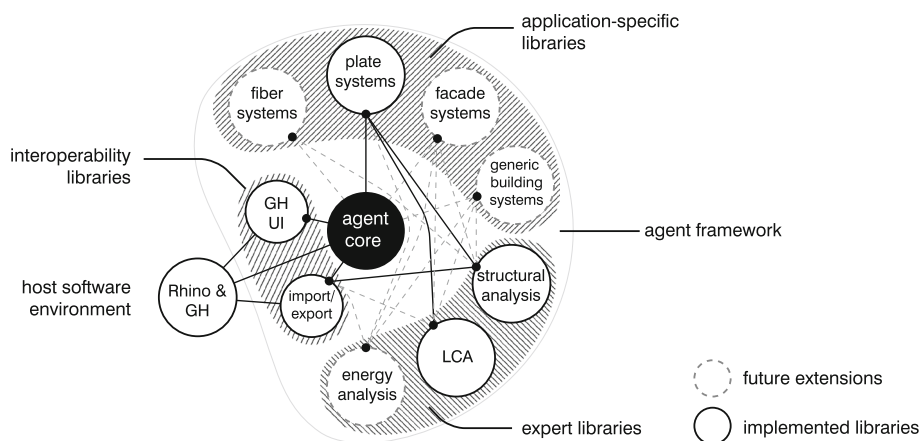
The proposed framework is field-specific insofar as it relies on methods of Computer Aided Geometric Design (CAGD), but generic enough to be able to be used by a variety of fields that involve geometric modeling in design (such as architecture, product design, and automotive design) or engineering (such as mechanical, civil, and aerospace engineering). In order to allow for extensibility and domain-specific applications, the framework is composed of a set of class libraries organized around a core agent library. These consist of application-specific libraries, expert libraries, and support libraries; the latter category includes user interface, import and export functionality as well as domain-specific functionality provided by the host software environment (see Fig. 3).

##### 3.1.2 Agent core library

The agent core library implements the functionality specific to agent-based modeling and simulation. It consists of five core classes: solver, agent system, environment, agent, and behavior. These classes are described in detail in Sect. 3.2.

##### 3.1.3 Application-specific libraries

The application-specific libraries contain classes that allow running experiments related to specific application domains. An example of such an application-specific implementation is the plate system case study described in Sect. 4; other applications include, for example, systems based on fiber-reinforced polymers (FRPs) or conventional building systems (using



**Fig. 3** Schematic overview of the agent framework structure

elements such as walls, floors, columns, or facade panels). The framework then facilitates running experiments of such systems consisting of multiple interacting entities, which in turn facilitates studying macro-level behaviors of micro-level, domain-specific rules.

### 3.1.4 Interoperability libraries

The interoperability libraries contain functionality related to the user interface, data import, and data export as well as access to libraries provided by the host software environment. Data visualization and display support are implemented through visualization components and the model view of the host environment, so various properties of agents and the agent system can be observed by the user. For more involving analysis, numeric data can be exported in various file formats and (through custom scripts) in specific file formats that might be required by external analysis software packages. In addition to exporting files for specific software, the export class can also be used to store data at each iteration (e.g., the agent positions or behavior settings) in order to allow for checkpointing of the agent system.

### 3.1.5 Workflow

In the framework, the designer defines how agents should react to external (e.g., surface curvature of the environment, or proximity to other agents) or internal conditions (e.g., its own size or shape) by defining agent behaviors. By acting in accordance with the behaviors assigned to them, agents engage in self-organization, which on the level of the agent system may lead to emergent behavior (De Wolf and Holvoet 2004). The difference between the user-defined behavior of agents and the emerging behavior of the agent system as a whole is important to note, as the designer can only influence the former but is ultimately interested in the latter.

### 3.1.6 Contributions

The framework we propose extends the functionality used for the Landesgartenschau Exhibition Hall in the following ways:

1. Multiple agent types (e.g., some that form planar elements and others that consist of linear elements) can coexist and interact, resulting in a heterogeneous system.
2. A more explicit way to assign different behaviors to different groups of agents is provided.
3. Additional agent types can be defined by the designer using inheritance.
4. Behaviors can use and modify information that is locally stored in the environment.
5. Behaviors can act on non-geometric properties.
6. Additional behaviors can be defined by the designer without modifying the framework code.
7. Import and export of data allow results from external programs to be included while the agent system is running.
8. The designer can directly influence individual agents while the agent system is running, for example changing an agent's position, or marking an agent as being unmovable.

## 3.2 Software implementation

### 3.2.1 Software and language

Our object-oriented software framework is implemented in C# and is integrated as a plug-in in Grasshopper, a node-based visual scripting tool for Rhinoceros.<sup>1</sup> This integration enables us to take advantage of the free-form curve and surface modeling capabilities of Rhinoceros' Non-Uniform Rational B-Spline (NURBS) modeling engine and also allows the use of Rhinoceros' 3D viewport. Furthermore, this integration allows designers to use our framework in combination with functionality offered by Grasshopper itself and other Grasshopper plugins (e.g., physics simulation, environmental analysis, genetic algorithms, and machine learning), in a modeling and scripting environment that is familiar to many designers.

Grasshopper includes the ability to add scripting nodes to its canvas. The classes that we implemented (described below) can be programmatically referenced (and therefore inherited) in these components, which gives users the option to create custom variants of these predefined classes.

### 3.2.2 Environment class

The environment class defines the environment that the agents operate in. It can contain either a NURBS surface or a volumetric region. We subdivide the surface domain or volumetric domain by a user-defined resolution in order to be able to store information in particular areas of the environment, so that agents can react to locally stored environment properties. Agents can access Environment information through the agent system they are part of.

### 3.2.3 Agent system class

The agent system class aggregates instances of agents of one or more types, as well as an instance of the environment class. Its main functions are to pass on update calls from the solver to the agents, and to pass data from the agents through the solver to the display and export classes.

---

<sup>1</sup> McNeel: Rhinoceros 3D, [www.rhino3d.com](http://www.rhino3d.com)

### 3.2.4 Agent class

Agents are instances of the agent class. The base agent class contains a list of behaviors that are assigned to the agent, an execute method and a display method. Specific agent types derive from this class, so that additional parameters (such as a position) and specific methods can be added. In our case study (see Sect. 4), we define two types of agent classes: one representing plates and the other one representing the joints where edges of plates meet.

Agents have an execute method that calls the execute methods of all behavior instances that are assigned to the agent. Movement vectors that are computed by behavior instances are first stored individually and then combined into a single movement vector (see Sect. 3.2.5). This movement vector is used to move the agent to the new position after all behaviors for all agents have been calculated at the current iteration, except if the agent is currently being moved by the user; in that case, the agent is moved to the target position that is indicated by the current mouse position (see Fig. 10).

### 3.2.5 Behavior class

The behavior class defines how the state of an agent should change at each iteration. All behavior instances contain a weight parameter and a virtual execute method. This class acts as a base class from which all specific behaviors are derived. Apart from implementing the execute method, behaviors typically feature one or more user-adjustable parameters. These parameters can be adjusted by the user while the agent system is running.

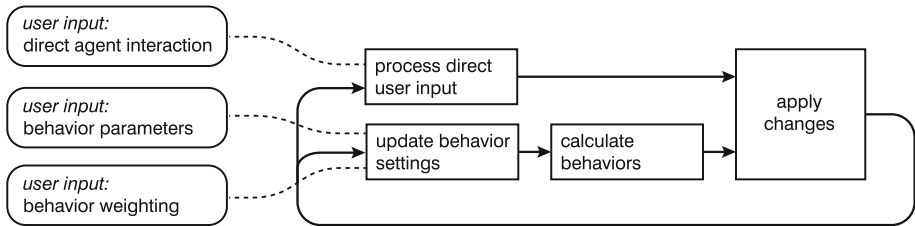
For geometric behaviors, the output of a behavior consists of one or more vectors that affect the position of the agent itself and/or its neighbors. Reynolds (1999) describes various ways to combine the influence of multiple behaviors, of which the most straightforward is a summation of vectors that act on an agent. This is the method we use for the case study described in Sect. 4. By applying a global scaling factor to the movement vector, the user can control the speed of agents. In case agents are linked to a base surface, the agent is pulled back to the surface after its position has been updated.

### 3.2.6 Solver class

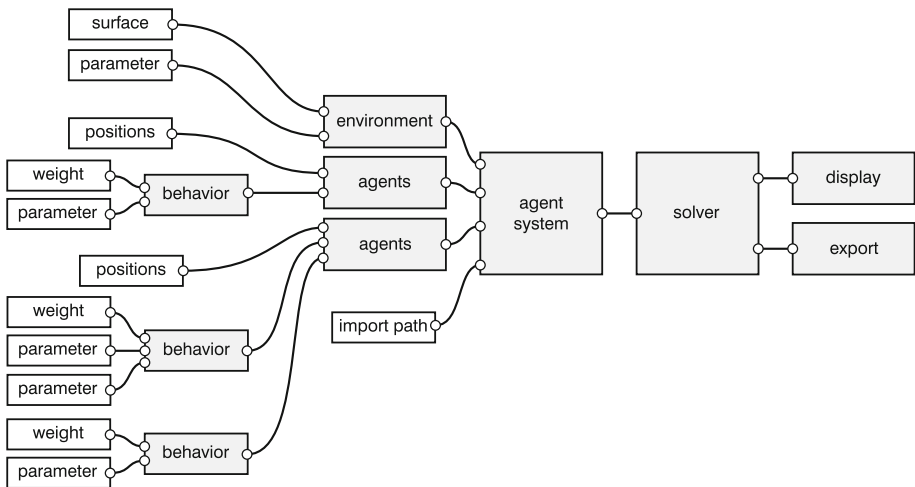
The solver class manages one or more agent systems, coordinates the updating process, manages import and export of data, and provides data to the display components.

The solver updates the agent system in a synchronous, iterative manner. The current state of the overall system can be visualized either at each iteration, after a certain number of iterations have passed (which is computationally more efficient, but visually less fluent), or only when a final state has been reached (either after a certain threshold has been reached, or after a fixed maximum number of iterations). This latter method (similar to a batch mode in other frameworks) can be used for parameter optimization and sensitivity analysis of the model. Before executing any behaviors, the user-defined behavior parameters (including the weighting factor) are checked, to allow the user to adjust behaviors while the agent system is running (see Fig. 4).

Certain agent behaviors depend on agents that are in close proximity. In agent systems with large numbers of agents, computation of behaviors that rely on neighborhood searches can be computationally expensive. Therefore, we accelerated the neighbor lookup step by using the R-Tree algorithm (Guttman 1984), which is particularly beneficial for systems that contain more than 1000 agents.



**Fig. 4** Illustration of program execution logic. Solid lines indicate the order in which processes are executed. Dashed lines indicate input that does not trigger an update, but will be taken into account the next time a process is executed. When no changes are made to any behavior settings, behaviors will be calculated using the same settings as in the previous iteration



**Fig. 5** Schematic representation of the graphical programming interface in Grasshopper. Shaded boxes represent classes of the agent framework, and white boxes represent data that needs to be provided by the user. Circles on the left side of boxes represent inputs, and circles on the right side of boxes represent outputs.

### 3.2.7 Visual programming interface

For all classes described above, Grasshopper components have been created. By connecting these components and providing additional inputs, a user can set up an agent system without any programming being involved. A typical setup is shown in Fig. 5.

## 3.3 User control

Reynolds (1987) suggests that using his flocking system, animators become meta-animators. We envision a similar role for designers, who control how an agent system behaves while adjusting behavior parameters in real time and occasionally nudging agents in a desired direction.

While the proposed agent system is being executed, a designer can interact in the following ways:



**Fig. 6** A midi interface that can be used to interactively control agent behavior

1. The strength of behaviors can be adjusted by a global multiplier, which has the effect that movements within the system can be sped up, slowed down or temporarily stopped.
2. The strength of individual behaviors can be modified, so that based on visual feedback, the designer can control which behaviors should exert a stronger influence.
3. Behavior parameters for each individual behavior can be adjusted.
4. Individual agents can be moved by the designer.

The first three modes of interaction can be controlled using number sliders on the Grasshopper canvas (“behavior parameters” and “behavior weighting” in Fig. 4). The last interaction method happens in Rhino’s 3D view, where mouse movements in screen space can be translated into agent movements in 3D space (“direct agent interaction” in Fig. 4).

### 3.4 Behavior control interface

While the Grasshopper user interface provides various types of user interface elements, the user may prefer additional or alternative input methods. To facilitate such input methods, behavior parameters can be set to check the contents of a file that stores parameter values, once per iteration. Any kind of user interface software or hardware component can thus update parameter settings by writing to this file. Additionally, with this mechanism, external data sources (e.g., sensor data or live data from sources on the Internet) can be used to inform behaviors.

An example of an alternative interface for behavior control is an external midi controller (see Fig. 6) with hardware sliders and knobs. Such a device has certain advantages, such as the ability to change multiple sliders at once. Additionally, thanks to haptic feedback, one can keep observing changes in agent behavior while modifying parameters.

### 3.5 Data integration and external computation

For various types of analysis, external data sources or calculations may need to be linked to the agent framework. This section discusses various ways in which such integration can be accomplished.

### 3.5.1 Geometric data

Geometric data can be generated and displayed directly, using geometric properties of the agents in the agent system. Examples of such data are the number of elements in a design, the total length of joints between elements, the total floor area, and the volume of the design.

### 3.5.2 Integrated data sources

By using data from specific data sources in the framework (either as text files, in a separate database or provided by the user in the Grasshopper environment), feedback that depends on not just the geometry but also the material of the design can be generated. For example, by specifying the material buildup of a construction and the nature of machining operations that are necessary to fabricate each joint, the total material use, prefabrication time, and costs can be estimated. By including life cycle analysis (LCA) data for the materials and processes in question, the environmental impact of a design can be compared with alternative designs.

### 3.5.3 External calculation behavior loop

For certain types of analysis, the course of action to take in response to the results of particular calculations can be formulated in explicit rules. In this case, data can be exported from the agent framework to an external program and the result can be read in. An example of this is the adjustment of joint properties that affect the capacity of the joint to transfer loads (such as number and spacing of structural connections) based on the calculated forces that occur in the joints.

External calculations can be orders of magnitude slower than purely geometric behaviors. Therefore, we implemented the option to activate behaviors of this kind only after the system has reached dynamic equilibrium, which is measured by comparing the displacement of the fastest moving agent with a threshold value.

### 3.5.4 Export

A model for structural analysis can be exported using a custom script that writes geometric data, material properties, and joint properties to a file, in the input data format defined by the FEM analysis program that is being used (e.g., Sofistik).<sup>2</sup> In order to facilitate this scenario, structural engineering input would need to be provided right at the start of the design process. Similar scenarios could be set up for other types of analysis, such as daylight, heating, or acoustics.

All agents that represent building elements feature methods that generate the fabrication data that are required for their fabrication. Thus, instead of needing a subsequent Computer Aided Manufacturing (CAM) stage as is typically the case in CAGD, design and fabrication planning are tightly integrated (Krieg et al. 2015).

### 3.5.5 Environment-informing external analysis

Certain external calculations yield results that directly relate to the geometric environment instead of to particular agents. For example, solar insolation calculations for external surfaces could be executed once, after which the results are stored in the agent system's environment. An agent behavior can subsequently query the environment for this insolation information.

<sup>2</sup> SOFiSTiK AG: SOFiSTiK—FEM, BIM, and CAD Software for Structural Engineers, [www.sofistik.de](http://www.sofistik.de)

### 3.5.6 Environment-modifying behaviors

In the biological example of ant colonies, ants have the ability to modify their environment. In the proposed framework, this can be done in an equivalent way through behaviors that modify the environment, for example by locally changing a NURBS surface that agents move around on, or by locally storing data that influence behaviors in the environment.

### 3.5.7 Performance-encoded behavior

Instead of executing external analysis as part of a software loop, specific agent behaviors can be created that directly try to ensure particular performance. For example, near-planar areas in polyhedral shell structures are known to be structurally weak; the occurrence of such areas can be countered by using a behavior that moves plates in directions that result in larger plate angles.

While this way of integrating domain knowledge fits the proposed agent-based approach best, formulating behaviors that ensure particular performance on a certain metric is difficult and may in some cases not even be possible. However, the framework we propose provides an environment that can be used to develop and test such behaviors.

## 4 Case study: design of plate structures

We are using the proposed agent framework in ongoing research on plate structures, as well as for the architectural design of plate structures. This section describes the aim of this case study and the features that have been implemented for this specific application.

### 4.1 Case study aim

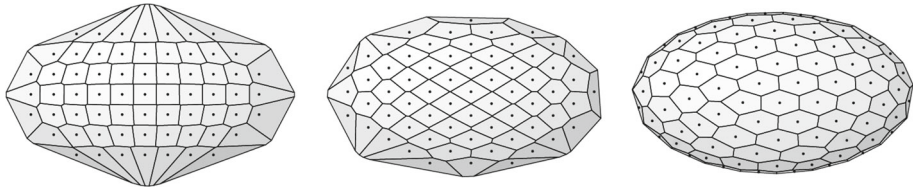
The aims of this case study are as follows:

1. Test the proposed interactive agent-based workflow.
2. Test the proposed integration of domain-specific data sources and calculations.
3. Create tools for the architectural design of polyhedral plate structures.
4. Create an experimental setup for the further study of plate structures, including studies of geometry, structural performance, and environmental impact.

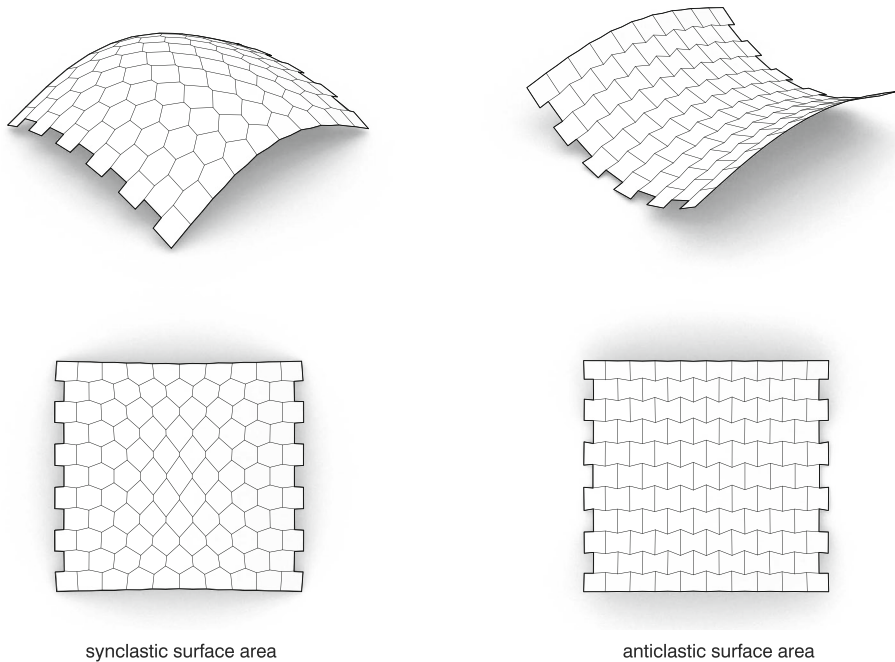
### 4.2 Background

Modeling polyhedral geometry in conventional CAGD software is a labor intensive process, as changes in position or orientation of a polyhedron (to which we will refer as *plate*) affect the neighboring polyhedrons. Additionally, the interrelation between a base surface, the positions where plates are created, and the resulting plate shapes is difficult to intuitively predict: As can be seen in Fig. 7, creating the plates at positions defined by a regular grid may result in a somewhat regular appearance in certain areas, while leading to irregular shapes in areas with different curvature conditions.

Strategies for approximating double-curved surfaces with planar elements are actively being studied within the fields of architectural geometry as well as computer graphics (Cohen-Steiner et al. 2004; Wang et al. 2008; Zadavec et al. 2010; Li et al. 2015). Structural behavior of polyhedral structures is another topic that is only partially explored (La Magna et al. 2012;



**Fig. 7** Plate patterns resulting from various distributions of points on a base surface: When distributing plates on a surface using a regular square grid (left) or hexagonal grid (middle), the resulting polygonal shapes are irregular and structurally unfavorable: The continuous joint lines would constitute folding lines. The plates in the right image have been distributed using our agent-based approach and show a much more regular and structurally favorable distribution of plate sizes and edge lengths



**Fig. 8** Typical plate shapes on synclastic surfaces (left) and anticlastic surfaces (right)

Li and Knippers 2015). The development of a computational environment that facilitates the systematic study of plate structures would enable further research in this area.

### 4.3 Plate geometry generation

We generate polyhedral plate geometry by defining agent positions on a NURBS surface and then intersecting the surface tangent planes at these locations. The resulting plate geometry reflects the underlying geometry: Synclastic areas result in convex polygons, whereas anticlastic areas result in concave polygons such as bow tie-shaped polygons (see Fig. 8). This relation between surface curvature and polygon shape is described in more detail in Wang et al. (2008).

In the software framework, we implemented two intersection algorithms. The first algorithm, tangent plane intersection (TPI) (Troche 2008), works on synclastic as well as anticlastic surfaces and consists of two steps: First, a network interaction topology is computed using Cartesian distance; then plate contours are defined based on this interaction topology. This method works reliably for certain distributions of points, but in particular in areas with low Gaussian curvature (including transition zones between synclastic and anticlastic surface areas), self-intersecting (and thus invalid) plate contours may occur.

The second intersection algorithm consists of an iterative process, in which for each plate, geometry is sliced off from an initial bounding box. This method does not explicitly generate topology diagrams, but instead stores connectivity information in the edges between the resulting polygons. This method only works on synclastic surfaces, but works for any distribution of points and thus is more robust than TPI.

In order to study agent behaviors in a controlled way and to simplify the development of curvature-dependent behaviors, we use the latter method for the case study presented here.

#### 4.4 Agent types

For the design of plate structures, we are using two agent types: One type represents the plates in the structure, and the other type represents the edges. The rationale behind using the additional edge agent type is that in plate structures, edge forces are of particular relevance. By using edge agent objects, the results of structural finite element analysis (FEA) can be easily stored in the agent system. Additionally, certain geometric goals (such as ensuring that all edges are longer than a certain minimum length) can be more easily approached by assigning behaviors to edges instead of plates. However, it should be noted that the geometric effects of edge behaviors act on plate agents and not on edge agents.

Plate agent positions are used to recreate geometry at every iteration. Edge agents are subsequently created for every edge in the resulting geometry, inheriting properties from edge agents of the previous generation that share the same neighboring plate agents. These two agent types are illustrated in Fig. 9.

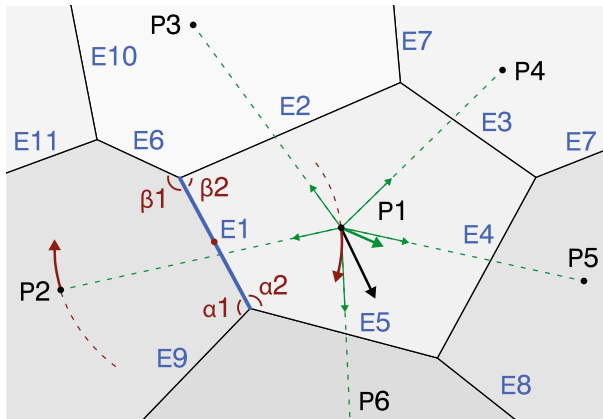
#### 4.5 Agent behaviors

In the design approach we propose, behaviors are tools that a designer employs to reach a particular goal. For example, behaviors could be devised with the goal of ensuring that plates can be produced (by limiting the size of plates to the size of available stock material), with the goal of creating a visually coherent plate pattern (e.g., by reacting to edge lengths and edge angles), or with the goal of creating structurally favorable plate configurations (e.g., by reacting to near-collinear edges). Within the framework, a designer can define any number of custom behaviors.

In this section, we describe behaviors that we developed for this study. Additional descriptions of behaviors can be found in Reynolds (1999) and Schwinn et al. (2014).

##### 4.5.1 Plate behavior: locally gradated cohesion and separation (LGCS)

Behaviors that Reynolds (1999) describes include cohesion and separation behaviors. Together, these tend to create an even distribution of agents, which in our application leads to plates that are all roughly the same size. In order to have more local control over plate size, we use a behavior in which the cohesion and separation distances are not constant, but



**Fig. 9** Interaction of edge agents and plate agents using geometric behaviors: Edge agent E1 has a behavior (EAE, described in Sect. 4.5.5) that acts on neighboring plate agents P1 and P2 and tries to equalize angles  $\alpha 1$  and  $\alpha 2$ , as well as  $\beta 1$  and  $\beta 2$  (drawn in red). Plate agent P1 has a behavior that acts on itself (LPSE, described in Sect. 4.5.2), based on the proportion between its own surface area and the surface area of neighboring plate agents P2–P6. The vectors that act on plate agent P1 (drawn in green) are summed (drawn in black) and then used to move P1 before the next iteration starts

instead depend on either the distance of an agent to a guide curve or parameters that are locally defined in the environment.

In the *locally graded cohesion and separation* behavior, the user sets the ideal neighbor distance at a guide curve ( $d_c$ ), the ideal neighbor distance far away from the guide curve ( $d_f$ ) and the size of the area in which the neighbor distance should gradually change ( $2 \cdot d_{max}$ ). Additionally, a distance within which neighbors are taken into account should be defined. The resulting translation vector  $v$  is then calculated as follows:

$$v = w \cdot \sum_{i=1}^n \left( 1 - \frac{d_i}{\kappa \cdot d_c + (1 - \kappa) \cdot d_f} \right) \cdot \hat{v}_i$$

Here,  $w$  is the weighting factor,  $n$  is the number of agents within range,  $d_i$  is the distance between the agent this behavior is assigned to and neighbor agent  $i$ ,  $\kappa$  the distance from the agent to the curve divided by  $d_{max}$ , and  $\hat{v}_i$  is the normalized vector to neighbor agent  $i$ .

#### 4.5.2 Plate behavior: local plate size equalization (LPSE)

The *local plate size equalization* behavior aims to achieve a locally consistent plate size, instead of defining a fixed size for all plate agents. Plate agents compare their own surface area with that of their direct neighbors and then calculate vectors toward the neighbors that are larger and away from the neighbors that are smaller. In order to prevent oscillating behavior that can occur when changes in plate connectivity occur, the resulting vector is multiplied by the length of the edge that connects the plate with its neighbor. The resulting translation vector  $v$  is calculated as follows:

$$v = w \cdot \sum_{i=1}^n \left( 1 - \frac{A_i}{A_a} \right) \cdot l_e \cdot \hat{v}_i$$

Here,  $w$  is the weighting factor,  $n$  is the number of connected agents,  $A_i$  is the surface area of neighbor agent  $i$ ,  $A_a$  is the surface area of the agent that this behavior is assigned to,  $l_e$  is the length of the edge that is shared with neighbor agent  $i$ , and  $\hat{v}_i$  is the unit vector to neighbor agent  $i$ . The translation vector calculation of this behavior is illustrated in Fig. 9, and a plate configuration using this behavior is shown in Fig. 14.

#### 4.5.3 Plate behavior: centroid (PC)

The *centroid* behavior moves a plate agent closer to the centroid of its corner points, or (at the choice of the user) the centroid of the mid points of its edges. This typically leads to plate shapes with an increasingly regular appearance. This behavior is calculated as follows:

$$v = \frac{w}{n} \cdot \sum_{i=1}^n p_i - p_a$$

Here,  $v$  is the translation vector,  $w$  is the weighting factor,  $n$  is the number of corners of the plate contour,  $p_i$  the position of either corner point  $i$  or the midpoint of edge  $i$  (at the user's choice), and  $p_a$  is the position of the agent that this behavior is assigned to.

#### 4.5.4 Edge behavior: plate target angle (PTA)

The *plate target angle* behavior compares the angle between neighboring plates and compares this to a user-defined target angle. Based on the result, the plates are either moved closer to the edge, or pushed further away. While this behavior is very simple, it has the effect that small plates form in areas with strong curvature and large plates in areas with less curvature. Each of the two plates that border an edge is moved by the following vector:

$$v_{b,i} = w \cdot \left(1 - \frac{\alpha_c}{\alpha_t}\right) \cdot v_i$$

Here,  $w$  is the weighting factor,  $\alpha_c$  is the current angle between the normals of the two plates that border this edge,  $\alpha_t$  is the target angle between these normals, and  $v_i$  is the vector that is perpendicular to the edge and points to neighboring plate  $i$ . A plate configuration using this behavior is shown in Fig. 14.

#### 4.5.5 Edge behavior: edge angle equalization (EAE)

In the *edge angle equalization* behavior, an edge agent checks the angles with the two pairs of edge agents it connects to. It then tries to orient itself in such a way that these angles become more equal. This is accomplished by rotating its neighboring plates around its own midpoint in either clockwise or counterclockwise direction, as illustrated in Fig. 9. This behavior tends to result in a plate structure with a quite regular appearance, in particular when combined with a behavior that enforces a certain minimum edge length (such as MNEL). The rotation angle of the neighboring plates is calculated as follows:

$$\alpha = w \cdot ((\alpha_1 - \alpha_2) + (\beta_2 - \beta_1)) \cdot l_e^n \cdot l_{a1} \cdot l_{a2} \cdot l_{\beta 1} \cdot l_{\beta 2}$$

Here,  $w$  is a weighting factor,  $\alpha$  is the angle by which the neighboring plates should be rotated, in the plane that is defined by the edge midpoint and edge normal (defined as the average of the normals of the neighboring plates). Angles  $\alpha_1$ ,  $\alpha_2$ ,  $\beta_1$ , and  $\beta_2$  are the angles between the edge that this behavior is assigned to and its connected edges, as illustrated in Fig. 9.

Lengths  $l_{\alpha 1}$ ,  $l_{\alpha 2}$ ,  $l_{\beta 1}$ , and  $l_{\beta 2}$  are the lengths of the connected edges, and  $n$  is a user-defined exponent value that controls to what extent the behavior strength depends on the edge length. The translation vectors of the affected plates are defined as the position changes that would occur from rotating their positions by angle  $\alpha$ .

#### 4.5.6 Edge behavior: edge length equalization (ELE)

The *edge length equalization* behavior aims to minimize the differences in edge lengths between connected edges, by moving plates in such a way that the length of an edge gets closer to the average length of the edges it is connected to. To lengthen the edge, the plates that border the edge will be moved closer, while the plates that the edge connects to at its end points (which we will refer to as the edge's 'end plates') will be moved further away; to shorten the edge, the inverse would take place. This behavior is calculated as follows:

$$v_{b,s,i} = w_s \cdot \left( \frac{l_a + c}{l_e + c} - 1 \right) \cdot v_{s,i}$$

$$v_{b,e,i} = w_e \cdot \left( \frac{l_a + c}{l_e + c} - 1 \right) \cdot (-v_{e,i})$$

Here,  $v_{b,s,i}$  is the translation vector that is applied to side plate  $i$ ,  $v_{b,e,i}$  is the translation vector that is applied to end plate  $i$ ,  $w_s$  and  $w_e$  are weighting factors,  $l_a$  is the average length of connected edges,  $l_e$  is the length of the edge this behavior is applied to,  $c$  is a constant value that prevents the equations from reaching near-infinite values for short edges, and  $v_{s,i}$  and  $v_{e,i}$  are vectors from the midpoint of the edge to the base points of affected plates. A plate configuration using this behavior is shown in Fig. 14.

#### 4.5.7 Edge behavior: minimal edge length (MNEL)

The *minimal edge length* behavior is intended to ensure that a generated plate configuration does not contain any edges that are shorter than a user-defined value. Just like the ELE behavior, to lengthen an edge, the plates that border the edge will be moved closer, while the plates that the edge connects to at its end points will be moved further away. This behavior is calculated as follows:

$$v_{b,s,i} = w_s \cdot \left( 1 - \frac{|0.5 \cdot l_{min} - l_e|}{0.5 \cdot l_{min}} \right) \cdot (-v_{s,i})$$

$$v_{b,e,i} = w_e \cdot \left( 1 - \frac{|0.5 \cdot l_{min} - l_e|}{0.5 \cdot l_{min}} \right) \cdot v_{e,i}$$

Here,  $v_{b,s,i}$  is the translation vector that is applied to side plate  $i$ ,  $v_{b,e,i}$  is the behavior vector that is applied to end plate  $i$ ,  $w_s$  and  $w_e$  are weighting factors,  $l_{min}$  is the user-defined minimum edge length,  $l_e$  is the current length of the edge this behavior is applied to,  $v_{s,i}$  is the vector pointing to the base point of one of the side plates from the closest point on this edge, and  $v_{e,i}$  is the normalized vector pointing from the midpoint of the edge to the edge's end point that touches end plate  $i$ .

#### 4.5.8 Non-geometric behavior: structural joint properties

After an initial plate structure has been found, structural FEA is used to estimate the expected edge forces. This information is then stored in edge agents and can be used to adjust structural properties of the joints, which will be used in the next FEA export iteration.

## 4.6 Data integration

As part of an ongoing research project into polyhedral timber structures, we included domain-specific data and integrated external structural calculations in the agent system's execution loop, so that various plate configurations and joint designs can be evaluated. This section describes how these domain-specific sources and analysis methods are integrated.

### 4.6.1 Finite element analysis loop

Structural analysis is carried out using the FEA (Bathe 1996) software Sofistik. In our case, we are particularly interested in forces and bending moments that occur in joints, so that the joint properties (in particular stiffness) can be adapted to these forces and moments. As this calculation takes multiple minutes per iteration, interactive agent control is no longer practical; therefore, we only run the structural analysis loop after other behaviors have reached equilibrium. The joint properties are iteratively adapted by automatically exporting in Sofistik's file format and by using the resulting joint forces as inputs for the edge agents in the agent system.

In principle, behaviors could be created that react to joint forces, with the goal to limit these forces. However, as local forces depend on global geometry, developing such behaviors would require further research, either by making micro-level behaviors adapt in response to global behavior (downward causation), or by explicitly developing micro-level behaviors that lead to structurally beneficial global behavior.

### 4.6.2 Material use, machining time, and costs

When assigning values for material use and machining times to plate agents as well as edge agents, the total material use and machining time of a plate structure can be readily assessed. By specifying costs for both materials and machining time, prefabrication costs of timber plates can be evaluated.

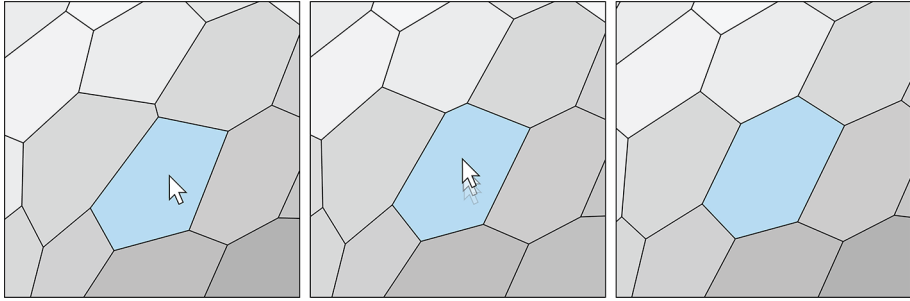
### 4.6.3 Life cycle analysis

LCA is typically based on environmental impact data of materials and processes (Consoli 1993); for each material and process, the environmental impact on a range of categories is established. While such data need to be provided by experts in the field of LCA, integration in the agent framework is straightforward as for each material or machining operation and for each environmental impact category, the impact can be calculated by multiplying the given LCA values by the material amounts or machine time (see Sect. 4.6.2).

## 4.7 Direct agent control

As described in Sect. 3.3, our software framework makes it possible to interact with individual agents while the agent system is running. For example, one can control the position of one particular agent by selecting it and moving it with the mouse. During the movement, the agent keeps influencing its neighbors, but its position will temporarily not be affected by its own neighbors. This process is illustrated in Fig. 10.

With certain behaviors, neighbors act as if they are making space for the agent that is being controlled by the designer. Once the agent that is being moved is released, the agent



**Fig. 10** User interaction with an individual agent: A designer can select a plate agent (left) and move it to a new position (middle), which may result in changes in plate connectivity. After releasing the mouse button, behaviors assigned to the agent and its neighbors will act until equilibrium is reached (right)

is again influenced by its neighbors, typically resulting in a quick move to a position where the agent is in dynamic equilibrium with its neighbors.

In addition to moving agents, we implemented functionality to permanently fix agents in a particular position. This can be used not only to organize agents at the edges of the environment, but also to fix areas with agents that have desirable characteristics while continuing to adjust behaviors for the remaining agents.

## 5 Results

### 5.1 Computation speed

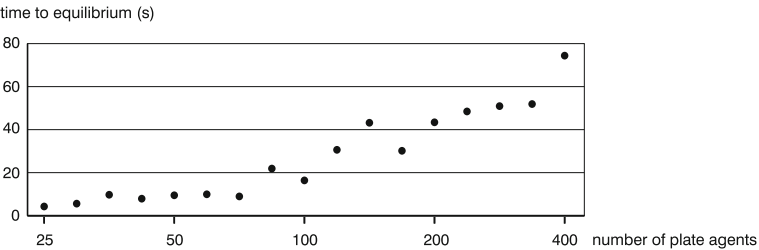
In scenarios where no external calculations are required, the proposed agent system typically runs at an interactive frame rate, with each iteration of an agent system with 100 plate agents being calculated in around 100ms on a laptop computer running on a single thread of a quad-core CPU at a clock speed of 2.7 GHz. This provides near real-time visual feedback and allows the designer to interact with the system. We implemented and tested the framework on personal computers, as this reflects the hardware that is typically used in architectural design; targeting such hardware lowers the barrier for adoption of ABM in the field.

Convergence time (the time it takes until all agents move less than a certain threshold distance per iteration) increases with the number of agents and also strongly depends on the number and kind of agent behaviors: Certain agent behaviors take very little computation time and reach equilibrium quickly, whereas other behaviors may either be computationally intensive or never converge at all. However, for combinations of behaviors (such as the ones described in Sect. 4.5), convergence time is typically well under one minute, as illustrated in Fig. 11.

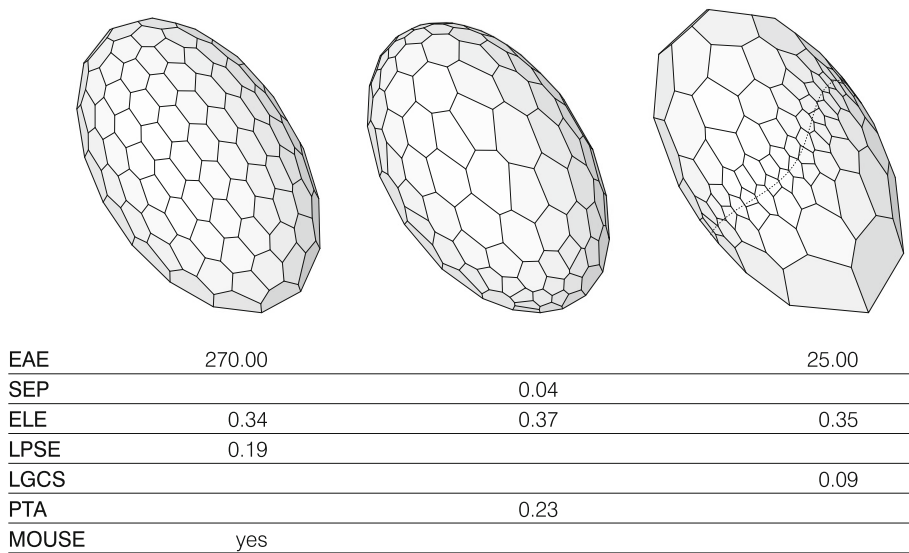
### 5.2 Design freedom

The effectivity of using agent behaviors in a design tool was tested by comparing the geometric output of different combinations of behaviors in a single environment. As illustrated in Fig. 12, the behaviors do indeed lead to a variety of results.

As can be seen in the table with behavior weighting settings (Fig. 12), the shown geometry resulted from using combinations of behaviors. It is worth noting that the timing and the order in which the designer makes changes to weighting settings during the design process affect



**Fig. 11** Convergence time as a function of the number of plate agents in an agent system with four active behaviors (LPSE, PC, PTA, EAE; see Sect. 4.5) and without direct user interaction. Convergence was defined as the moment when none of the agents moves more than a user-defined threshold distance per iteration

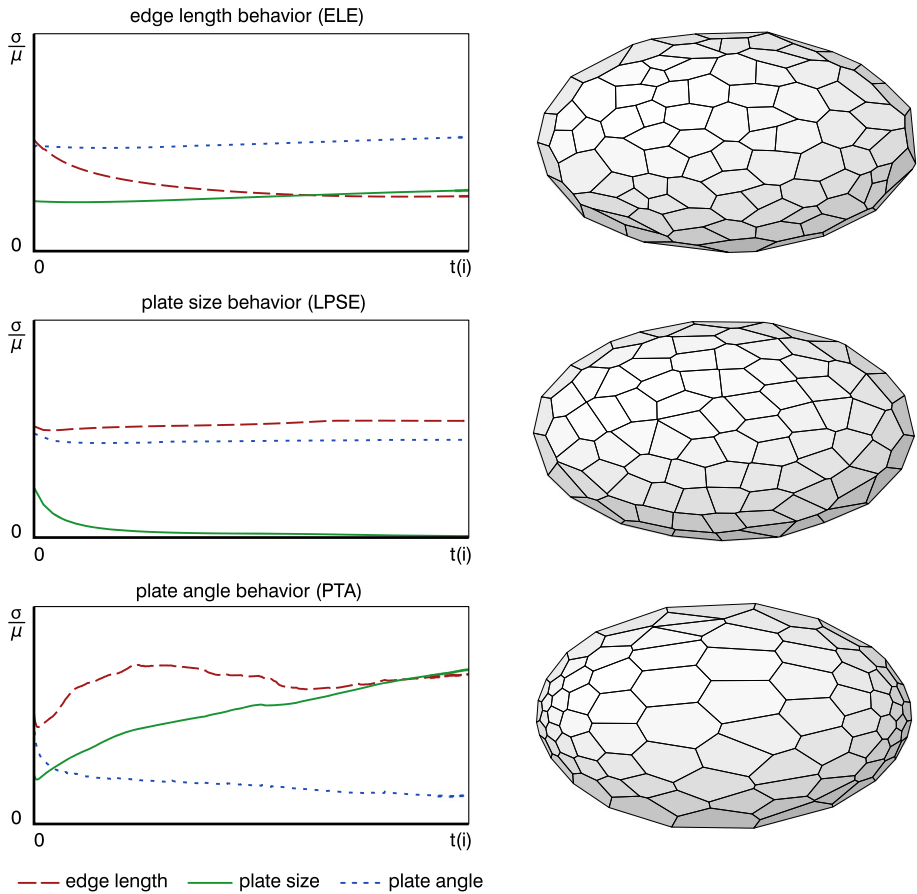


**Fig. 12** Three different plate patterns based on a single base surface, and the behavior weighting settings at the moment the agent system was stopped. The separation behavior (SEP) is calculated according to Reynolds (1999), and the other behaviors are described in Sects. 4.5 (EAE, ELE, LPSE, LGCS, PTA) and 4.7 (MOUSE)



**Fig. 13** Architectural design study, created using the tools described in Sect. 4

the final plate configuration. Fig. 13 shows an example of an architectural design study that was carried out using the behaviors described in Sect. 4.

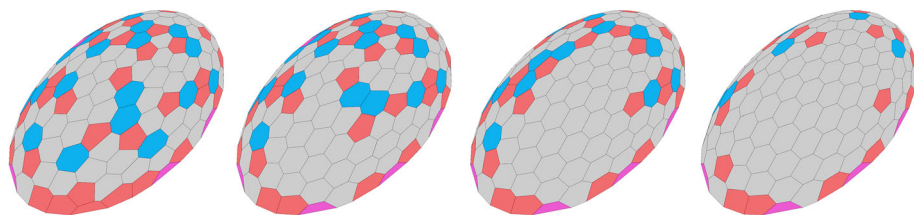


**Fig. 14** Three examples of the effect that specific behaviors have on the coefficients of variation ( $\sigma/\mu$ ) on various metrics. The top graph shows the ELE behavior, which over the course of multiple iterations reduces variation in edge length. The middle graph shows the LPSE behavior, which reduces variation in plate size. The bottom graph shows the PTA behavior, which reduces variation in angles between plates. The images on the right show the resulting plate configurations at the moment when execution of the agent system was stopped. In addition to the listed behaviors, a separation behavior with a small separation distance was active

### 5.3 Quantifiable geometric properties

Agent behaviors can be created with a particular goal in mind, in which case their effectivity in reaching this particular goal can be measured. Such goals can be particular performance on a metric calculated by external software, but goals can also be purely geometric (e.g., minimizing variance in plate size or making sure that plates are producible using given stock material). In this section, we compare the effects of agent behaviors that act on plate size (see Sect. 4.5.2), angles between plates (see Sect. 4.5.4), and edge length (see Sect. 4.5.6).

As can be seen in Fig. 14, behaviors can significantly reduce variance of the metric they act upon, but the variance is often not reduced to zero. In certain cases, reaching an absolute goal may not be geometrically possible. However, even if a geometric solution can be proven to exist, there is no guarantee that this solution will be found by a particular behavior, as



**Fig. 15** Four steps in a process in which the designer interactively adjusts plate agent positions executed while multiple agent behaviors (EAE, LPSE, ELE, PC) were active. The colors indicate the number of sides of the polygons

this might require a more radical reorganization of agents than the behavior will effectuate. Additionally, in practice multiple behaviors will be applied concurrently and they may act on agents in irreconcilable ways. This can be observed by comparing the effect that the uniform angle behavior exhibits in the bottom graph in Fig. 14: In the process of limiting the difference between plate angles, the variance in plate size sharply increases. Such effects can, however, be monitored easily, as geometric properties of the agent system can be evaluated in real time.

#### 5.4 Qualitative characteristics

As aesthetic considerations play a significant role in architectural design, design tools should enable the designer to reach visually compelling designs. Although there may be no universal way to judge aesthetics, some factors that contribute to visual coherence of polyhedral structures can be identified, such as consistency in plate size, in edge length, and in angles between edges. Geometric agent behaviors that affect such properties can be used for formal design exploration.

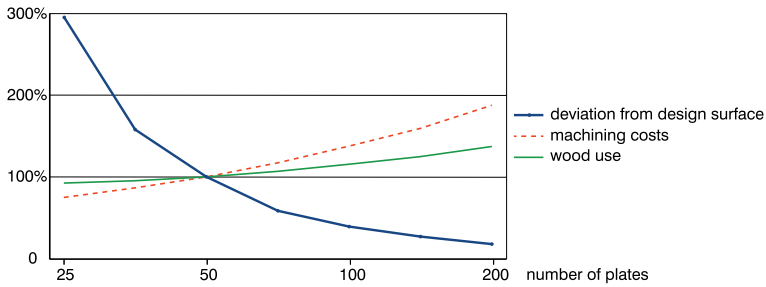
Direct control of individual agent positions turned out to be an effective means to locally improve the visual coherence of plate structures. As illustrated in Fig. 15, a designer can increase the regularity of a plate structure significantly by interactively adjusting the positions of a small number of agents while the agent system is running.

#### 5.5 Data integration

As we integrated material use and machining time data in the framework (defined per element, per running meter, per square meter or per cubic meter), creating output of this kind of data is straightforward. By linking this information to known costs per material and machining operation, cost estimations can be followed in near real time and therefore, the implications of design decisions can be readily assessed. An example of this is shown in Fig. 16.

Using LCA data provided by an expert in this field, environmental impact of the polyhedral structure has been assessed. As LCA results strongly depend on machining time and material use, the impact of design decisions on environmental impact can be readily calculated, in a similar manner as is used to estimate costs.

Using the syntax and parameters of structural analysis export that were provided by a structural engineer, structural analysis has been integrated in a loop. Using this integrated structural analysis, we adjusted joint properties to occurring forces. By combining the output of structural analysis, LCA analysis, and cost estimation, the impact of such adjustments on costs and environmental impact can be assessed.



**Fig. 16** Relative material use and machining costs for the fabrication of a timber plate structure with varying numbers of plates, where a design with 50 plates is used as a reference. These data are generated by running the agent system multiple times with varying numbers of plate agents. Using more plates allows a closer approximation of the input surface, but results in more material use and machining time

## 5.6 Behavior effectivity

Establishing relationships between agent behaviors and the emerging agent system behavior can be a challenging and time-consuming task, in particular when behaviors depend on external calculations. While the effectivity of behaviors that we implemented to attain geometric goals has been demonstrated in Sects. 5.2–5.4, only tentative progress has been made on developing behaviors that improve structural behavior. Initial experiments suggest that the number of joints that are traversed in the main spanning directions should be minimized, but further research is needed in order to be able to develop effective agent behavior strategies.

Results of LCA are based on material use and machining time. As these are both dependent on joint length, geometric behaviors that lead to a shorter total edge length also lead to a lower environmental impact. Therefore, with a given number of plates, geometric behaviors that shorten edge length (by moving plates toward long edges) or change the area-to-edge length proportion of plates (by making plates more regularly shaped) can be effectively employed to reach targets that are set based on LCA.

## 5.7 Design approach effectivity

Implementing the agent framework and the tools that have been used for this research has taken more time than would normally be available in a design project. Therefore, reusability is of major importance for the viability of this approach in industry. The function of the framework as a development environment (including its object-oriented nature) addresses this issue of reusability. While the amount of time needed to develop effective behaviors and agent types remains an unpredictable factor, successful behaviors and agent types can be reused in ongoing and future design and research projects. At the same time, the tools and framework can be refined and developed further, so the time invested per project will be reduced. Furthermore, the tools we developed can be used for design tasks that are otherwise practically impossible to realize.

Getting familiar with new tools takes time, but as the influence of adjusting behaviors is shown in near real-time, a designer can develop an intuitive understanding of the proposed design approach in a matter of minutes, even without prior experience in ABM. The way agents interact while adjusting behavior weighting or while the designer directly controls a single agent can help in gaining an understanding of how agent behaviors affect macro-level system behavior. It also helps in understanding the nature of combinations of behaviors.

## 6 Discussion

In this paper, we introduced a software framework that extends the agent-based design approach described in Schwinn et al. (2014) in various ways: Multiple agent types can coexist and interact, subgroups of agents can be defined by assigning different behaviors, custom agents and behaviors can be created through inheritance, behaviors can use and modify information stored in the environment, behaviors can act on non-geometric properties, calculations using external software can be integrated with the agent loop, and the designer can directly interact with specific agents (e.g., changing their position).

### 6.1 Data integration and external calculation

We implemented the possibility to include external data sources, as well as the ability to integrate the results of computationally expensive analyses by exporting to external analysis programs and reading back the results in a feedback loop. By storing the results of external analysis in agent parameters or in the environment, behaviors can access this information.

Structural evaluation on the system level using FEA typically takes multiple minutes (and sometimes even hours); however, in comparison with common building design processes in which files are sent back and forth between different designers and engineers, the feedback is still very fast. Apart from FEA, we integrated life cycle analysis data for the estimation of environmental impact, so that the designer can get feedback on the environmental impact of a design early in the design process.

### 6.2 Interactive workflow

Our software framework implementation provides an interactive design environment, which is enabled by near real-time computation speed. In the agent-based design approach we propose, the designer no longer draws elements, but instead controls the design by creating and interactively controlling agent behaviors. The result is a design process that is of a qualitatively different nature than conventional architectural design processes. Near-instant visual feedback of the influence of different combinations of behaviors helps adjust to this approach quickly. The additional functionality to directly interact with individual agents while the agent system is running turned out to be very effective in locally improving plate configurations.

In addition to using visual feedback, the designer can interactively evaluate a design on a range of criteria. Beyond geometric information such as the size of elements, information on, for example, material use, costs, and environmental impact can be displayed in near real-time as well, based on integrated external data sources.

### 6.3 Behavior effectivity

We showed how various behaviors that we implemented can lead to quantifiable results, including consistency in geometric properties such as element dimensions or angles between plates. Qualitative results (evaluated by the designer) can be achieved as well, in particular when making use of the possibility to directly interact with individual agents while the agent system is running.

## 6.4 Applicability

Due to its unique ability to accommodate change, the proposed agent-based design approach can increase the effectiveness of design processes. The ability to get feedback data on various performance criteria of a design early on (enabled by the integration of analysis tools within the design loop) is not only useful to create designs more rapidly; equally importantly, it can lead to better informed and thus potentially more effective design solutions. Thus, agent-based architectural design approaches can lead to the design of buildings that in many respects perform better than buildings that are designed using conventional methods.

### 6.4.1 *Applicability to conventional design tasks*

While the focus of our research is on design approaches for novel building systems, a similar design approach could be applied to the design of conventional buildings. As discussed in Sect. 2.4, there are already various subtasks of architectural design for which agent-based methods have been applied. However, due to the challenging process of translating design objectives into agent behaviors (see Sect. 5.6), we expect agent-based design approaches to be mostly beneficial to geometrically complex design tasks, to design tasks with a clearly measurable objective, and to highly complex design tasks in which many requirements need to be resolved in parallel.

### 6.4.2 *Applicability in architectural practice*

This research has been carried out in an academic context, and while our design studies (such as the one illustrated in Fig. 13) are similar to design tasks that may occur in architectural practice, the conditions under which these design studies are executed differ. In particular, integrating LCA and structural analysis could only be done using data and expertise provided by academic experts in these fields; the proposed approach thus depends on the willingness of stakeholders in the design process to collaborate. In architectural practice, external experts typically become involved in later phases of the design process; before getting their input, agent behaviors that relate to a specific domain would need to be based on design guidelines or preexisting knowledge in this domain. If necessary, these behaviors could be updated later in the design process. For example, if calculations of a structural engineer indicate that the proposed dimensions of building parts are too small, an updated version of the design can be generated rapidly by adjusting the input parameter of a behavior. As proposed in Sect. 2.2, this could reduce the cost of design changes.

## 6.5 Validation

As discussed in Sect. 2.3, interactive agent-based design approaches cannot be validated statistically in the same way as PSO or ACO can be validated. In principle, the effectivity of agent-based design approaches could be established by comparing the speed and outcome of an agent-based design approach to more conventional design approaches. However, as we focus on design tasks for which conventional design methods are not suitable, such a comparison cannot be made. Consequently, the only way we have to evaluate the proposed agent-based design approach is through our own experience using the approach in design studies.

## 7 Outlook

As developing and testing agent behaviors is an open-ended task that can be time consuming, we are currently investigating the use of statistical methods to explore the relation between user-defined agent behavior and the emerging agent system behavior. Of specific interest is the performance of emerging agent configurations on various metrics, including (but not limited to) structural performance.

In its current state, we are already using the agent framework for research and development of building systems, including structural explorations of novel building systems (such as timber plate structures and fiber composite systems). Additionally, the framework is being used to investigate and develop strategies to generate polyhedral approximations of double-curved surfaces.

Further development of interactive agent-based design approaches (including further development of behaviors as well as of user interaction methods) is ongoing, and we are developing and evaluating the software framework we introduced in this paper in various ongoing and upcoming design and construction projects. Investigating the applicability of the proposed design approach to more conventional building systems is a further potential avenue of future research.

The increasing ubiquity of the integration of sensors in robotic fabrication processes opens up the possibility of expanding the agent-based modeling approach from digital design to processes that act on the physical environment. For example, control approaches such as path correction and sensor-guided motion can be conceptualized in a similar behavioral model as digital design agents and behaviors. Given the inherent open-endedness of the agent-based software framework, a potential avenue of future research is the modeling of specific robotic agents and behaviors with the aim of integrating real-time robotic fabrication control.

## 8 Conclusion

This paper introduces a software framework for agent-based design that extends previous work (Schwinn et al. 2014) and is well-suited for the design of building systems that consist of families of geometrically unique elements. Improvements on previous work include a more interactive workflow, more flexible definitions of agents and behaviors, support for heterogeneous agent populations, and more direct integration of data sources and external analysis software.

The agent-based design approach we propose differs qualitatively from conventional design approaches: Instead of drawing geometry directly, the designer creates and controls behaviors that generate geometry. Because of near-instant visual feedback, the designer can quickly develop an intuitive understanding of the behaviors. The designer can also interact with agents directly, for example by manually adjusting their position. In addition to visual feedback of the agent configuration, the agent system provides data that can be displayed graphically. Thanks to the integration of domain-specific data (such as life cycle analysis) and external analysis software (such as FEA), designers can evaluate designs rapidly.

Through a case study that focuses on the design of polyhedral plate structures, we show how low-level agent behaviors affect the global design output. We developed various behaviors and show how these can be used to achieve quantitative as well as qualitative geometric properties.

Due to its unique ability to accommodate change, the proposed agent-based framework can be effectively employed in architectural design. Additionally, it can be used for the

prototyping of behaviors and for studying the relationship between low-level agent behaviors and emerging high-level system behaviors. Finally, it provides an environment that can be used in further research, including development of novel building systems and investigations of their structural behavior. As such, the framework contributes to the wide field of behavior-oriented design for architecture.

**Acknowledgements** This research has been partially funded by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center (SFB/Transregio) 141 ‘Biological Design and Integrative Structures.’ The authors would like to thank their colleague Ehsan Baharlou for the inspiration provided by his doctoral research on the theory of agent systems in architectural design.

## References

- Anumba, C., Ugwu, O. O., Newnham, L., & Thorpe, A. (2002). Collaborative design of structures using intelligent agents. *Automation in Construction*, 11(1), 89–103.
- Arndt, O., Peter, S., & Wünnenberg, D. (2000). *Hyperorganismen. Essays, fotos, sounds der ausstellung “Wissen” des ZKM im themenpark der expo 2000*. Hamburg: Internationalismus Verlag.
- Baharlou, E., & Menges, A. (2013). Generative agent-based design computation. In *Computation and performance: Proceedings of the 31st international conference on education and research in computer aided architectural design in Europe* (pp. 165–174). Delft: eCAADe
- Baharlou, E., & Menges, A. (2015). Toward a behavioral design system: An agent-based approach for polygonal surface structures. In *Computational ecologies: Design in the anthropocene, proceedings of the 35th annual conference of the association for computer aided design in architecture (ACADIA)* (pp. 161–172). Cincinnati, OH: University of Cincinnati.
- Ball, P. (2012). Pattern formation in nature: Physical constraints and self-organising characteristics. *Architectural Design*, 82(2), 22–27.
- Bathe, K.-J. (1996). *Finite element procedures*. Englewood Cliffs, NJ: Prentice Hall.
- Beetz, J., van Leeuwen, J., & de Vries, B. (2004). Towards a multi agent system for the support of collaborative design. In *Developments in design & decision support systems in architecture and urban planning* (pp. 269–280). Eindhoven, NL: Eindhoven University of Technology.
- Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl. 3), 7280–7287.
- Boyd, J. E., Hushlak, G., & Jacob, C. J. (2004). SwarmArt: interactive art from swarm intelligence. In *Proceedings of the 12th annual ACM international conference on Multimedia* (pp. 628–635). New York: ACM.
- Brownlee, J. (2011). *Clever algorithms: Nature-inspired programming recipes*. Morrisville, NC: Lulu.
- Cohen-Steiner, D., Alliez, P., & Desbrun, M. (2004). Variational shape approximation. *ACM Transactions on Graphics (TOG)*, 23(3), 905–914.
- Consoli, F. (Ed.). (1993). *Guidelines for life-cycle assessment: A “code of practice”; from the SETAC Workshop held at Sesimbra, Portugal, 31 March–3 April 1993*. Pensacola, FL: Society of Environmental Toxicology and Chemistry.
- Construction Users Roundtable. (2004). *Collaboration, integrated information and the project lifecycle in building design, construction and operation*. Cincinnati, OH: Construction Users Roundtable.
- De Wolf, T., & Holvoet, T. (2004). Emergence versus self-organisation: Different concepts but promising when combined. In *International workshop on engineering self-organising applications* (pp. 1–15). Heidelberg: Springer.
- Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, 2(3), 1461.
- Dorigo, M., Maniezzo, V., & Colnari, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1), 29–41.
- Eiter, T., & Mascardi, V. (2002). Comparing environments for developing software agents. *AI Communications*, 15(4), 169–197.
- Forrest, S., & Mitchell, M. (2016). Adaptive computation: The multidisciplinary legacy of John H. Holland. *Communications of the ACM*, 59(8), 58–63.
- Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, 1(1), 3–31.

- Gerber, D. J., Pantazis, E., & Wang, A. (2017). A multi-agent approach for performance based architecture: Design exploring geometry, user, and environmental agencies in facades. *Automation in Construction*, 76, 45–58.
- Glover, F., & Kochenberger, G. A. (2003). *Handbook of metaheuristics*. Dordrecht, NL: Kluwer.
- Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data* (Vol. 14, pp. 47–57).
- Heath, B., Hill, R., & Ciarallo, F. (2009). A survey of agent-based modeling practices (January 1998 to July 2008). *Journal of Artificial Societies and Social Simulation*, 12(4):9.
- Helbing, D., Buzna, L., Johansson, A., & Werner, T. (2005). Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Science*, 39(1), 1–24.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks, IV*, pp. 1942–1948.
- Krieg, O. D., & Menges, A. (2013). Prototyping robotic production: development of elastically bent wood plate morphologies with curved finger joint seams. *Rethinking prototyping: Proceedings of the design modelling symposium* (pp. 479–490). Berlin: epubli GmbH.
- Krieg, O. D., Schwinn, T., Menges, A., Li, J.-M., Knippers, J., Schmitt, A., & Schwieger, V. (2015). Biomimetic lightweight timber plate shells: Computational integration of robotic fabrication, architectural geometry and structural design. In *Advances in architectural geometry 2014*, (pp. 209–125). Cham, CH: Springer.
- La Magna, R., Waimer, F., & Knippers, J. (2012). Nature-inspired generation scheme for shell structures. In *Proceedings of IAASS-APCS 2012*. Seoul: Korean Association for Spatial Structures.
- Laugier, M. A. (1753). *Essai sur l'architecture*. Paris: Duchesne.
- Leach, N. (2009). Swarm urbanism. *Architectural Design*, 79(4), 56–63.
- Levitin, D. A., Lidicker, W. Z., & Freund, G. (2009). Behavioural biologists do not agree on what constitutes behavior. *Animal Behavior*, 78(1), 103–110.
- Li, J.-M., & Knippers, J. (2015). Pattern and form—their influence on segmental plate shells. In *Future visions: proceedings of the international association for shell and spatial structures symposium 2015*. IAASS.
- Li, Y., Liu, Y., & Wang, W. (2015). Planar hexagonal meshing for architecture. *IEEE Transactions on Visualization and Computer Graphics*, 21(1), 95–106.
- Macal, C. M., & North, M. J. (2005). Tutorial on agent-based modeling and simulation. In *Proceedings of IEEE the winter simulation conference*, pp. 2–15.
- Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2), 144–156.
- McMullen, C. P., & Jabbour, J. R. (Eds.). (2009). *UNEP Climate change science compendium 2009*. Nairobi: Earthprint.
- Menges, A. (2012). Biomimetic design processes in architecture: Morphogenetic and evolutionary computational design. *Bioinspiration & Biomimetics*, 7(1), 015003. <https://doi.org/10.1088/1748-3182/7/1/015003>.
- Minar, N., Burkhart, R., Langton, C., & Askenazi, M. (1996). *The swarm simulation system: A toolkit for building multi-agent simulations*. SFI Working Paper 1996-06-042. Santa Fe, NM: Santa Fe Institute.
- Nikolai, C., & Madey, G. (2009). Tools of the trade: A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12(2:2).
- Otto, F., Barthel, R., & Burkhardt, B. (1982). *Natürliche Konstruktionen: Formen und Strukturen in Natur und Technik und Prozesse ihrer Entstehung*. Stuttgart: DVA.
- Pollio, V. (1914). *The ten books on architecture*. Cambridge, MA: Harvard University Press.
- Puusepp, R. (2014). Agent-based models for computing circulation. *Design agency: Proceedings of the 34th annual conference of the association for computer aided design in architecture (ACADIA)* (pp. 43–52). Toronto: Riverside Architectural Press.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4), 25–34.
- Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game developers conference* (Vol. 1999, pp. 763–782). San Francisco, CA: Miller Freeman.
- Reynolds, C. W. (2000). Interaction with groups of autonomous characters. In *Game developers conference* (Vol. 2000, pp. 449–460). San Francisco, CA: CMP Media.
- Roller, D. (1991). An approach to computer-aided parametric design. *Computer-Aided Design*, 23(5), 385–391.
- Scheurer, F. (2007). Getting complexity organised: Using self-organisation in architectural construction. *Automation in Construction*, 16(1), 78–85.
- Schwinn, T., Krieg, O. D., & Menges, A. (2014). Behavioral strategies: synthesizing design computation and robotic fabrication of lightweight timber plate structures. In *Design agency—proceedings of the 34th annual conference of the association for computer aided design in architecture (ACADIA)* (pp. 177–188). Toronto: Riverside Architectural Press.

- Schwinn, T., & Menges, A. (2015). Fabrication agency: Landesgartenschau exhibition hall. *Architectural Design*, 85(5), 92–99.
- Shiffman, D., Fry, S., & Marsh, Z. (2012). *The nature of code*, D. Shiffman.
- Snooks, R. (2012). Volatile formation. *Log*, 25, 55–62.
- Snooks, R. (2012). Behavioral matter: Pulsations of the swarm. In Goldemberg, E. (Ed.), *Pulsation in Architecture*. Ft. Lauderdale, FL: J Ross Publishing.
- Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9), 1275–1296.
- Tamke, M., Riiber, J., Jungjohann, H., & Thomsen, M. R. (2010). *Lamella flock, advances in architectural geometry 2010* (pp. 37–48). Vienna: Springer.
- Taron, J. M., & Parker, M. (2014). Bounded Agency, *Design agency—proceedings of the 34th annual conference of the association for computer aided design in architecture (ACADIA)* (pp. 33–42). Toronto: Riverside Architectural Press.
- Troche, C. (2008). *Planar hexagonal meshes by tangent plane intersection* (pp. 57–60). Vienna: AAG: Advances in Architectural Geometry.
- Vasey, L., Baharlou, E., Dörstelmann, M., Koslowski, V., Prado, M., Schieber, G., Menges, A., & Knippers, J. (2015). Behavioral design and adaptive robotic fabrication of a fiber composite compression shell with pneumatic formwork. In *Computational ecologies: Design in the anthropocene, proceedings of the 35th annual conference of the association for computer aided design in architecture (ACADIA)* (pp. 297–309). Cincinnati, OH: University of Cincinnati.
- Vincent, J. (2009). Biomimetic patterns in architectural design. *Architectural Design*, 79(6), 74–81.
- von Mammen, S., & Jacob, C. (2008). Evolutionary swarm design of architectural idea models. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 143–150). New York: ACM.
- Wang, W., Liu, Y., Yan, D., Chan, B., Ling, R., & Sun, F. (2008). Hexagonal meshes with planar faces. In *Technical report TR-2008-13, Department of Computer Science*. Pokfulam, Hong Kong: The University of Hong Kong.
- Wester, T. (1989). Structures of nature in modern buildings. *Monthly Journal of Institute of Industrial Science, University of Tokyo*.
- Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings: Software Engineering*, 144(1), 26–37.
- Zadavec, M., Schiftner, A., & Wallner, J. (2010). Designing quad-dominant meshes with planar faces. *Computer Graphics Forum*, 29, 1671–1679.
- Zeng, Y., Buus, D. P., & Cordero, H. J. (2007). Multiagent based construction for human-like architecture. In *Proceedings of the 6th international joint conference on autonomous agents and multiagent systems (AAMAS 07)* (pp. 409–411). New York, NY: ACM.

## Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval, sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

[onlineservice@springernature.com](mailto:onlineservice@springernature.com)