AGENT-BASED SIMULATION FOR EARLY-STAGE PLANNING OF COMPLEX BUILDINGS
Contents

Acknowledgements 5

Foreword 7

Notes on style and structure of this Habilitation 11

I Scientific Excellence 13

Background: Early Planning of Complex Buildings 15
  Early Planning Is More Than Conceptual Design 16
  Occupational Media in Early-Stage Planning 20

Benefits of Agent-Based Simulation for Early Planning 49
  Paper: Meeting Simulation Needs of Early-Stage Design Through ABS 53

Data Sources for Early-Stage Agent-Based Simulation 63
  Paper: Pre-Tender Hospital Simulation using Naive Diagrams as Models 65
  Article: In-Process Agent Simulation for Early Stages of Hospital Planning 73
  Paper: Function & Action 89
  Article: Towards Use Cases in Sparse Architectural Data Exchange 97

Using Early-Stage ABS as Design Tool 107
  Article: Schematic Systems - Constraining Functions through Processes 109
  Paper: Reverse Engineering Hospital Processes Out of Visited Nodes 129
  Paper: From Quantities to Qualities in Early-Stage Hospital Simulation 137
Paper: Causality in Hospital Simulation Based on Utilization Chains 145
Paper: Sensitivity Visualization of Circulation under Congestion and Blockage 150

Additional Scientific Works 157
Paper: ProceeDings - A web-based word processor 159
Book: Agent-based Modeling and Simulation in Archaeology 171

II Didactic Qualification 173

Teaching Experience 175
Paper: How to Teach Architects (Computer) Programming - A Case Study 177

III Appendix 185

Projects 187
Early-Stage Simulation Landesklinikum Wiener Neustadt 187
STABLE AIR 187
STABLE HBO 188
MONARCA 188
MODYPLAN (ongoing) 189
Paper: MODYPLAN: Early-Stage Hospital Simulation 191

Glossary 199

Annotated list of figures and fonts used 203

Curriculum Vitae 211
Acknowledgements

This work would not have been possible without intensive collaboration, both in terms of research as well as concerning lectures. In this context, I would like to acknowledge the role of Wolfgang E. Lorenz, who has been a vital partner during the past ten years. The same goes for the whole team at the Department of Digital Architecture and Planning at the Vienna University of Technology, specifically Sigrun Swoboda, Arnold Faller, Georg Franck and Andreas Jonas. From the simulation side, I am glad I was introduced to Felix Breitenecker, Niki Popper (both are now entangled in hospital planning, I fear), Katharina Schigutt, Barbara Glock Irene Hafner, Matthias Rössler, Patrick Einzinger, Florian Miksch and Martin Bruckner. There are many more people whom I owe my gratitude. Richard Schaffranek, Christian Kühn, Bob Martens, Ardeshir Mahdavi, Henri Achten, Johan Verbeke, Emine Thompson, Burak Pak and Christa Illera gave me the impression that my research is important, and consequently motivated me to publish often, ambitiously and sometimes also exhaustively. I also want to thank Michael Bacher for introducing me to the world of complex planning tasks, which (as a computer scientist) immediately caught my interest. Hail also to the team around the Hallstatt, to Kerstin Kowarik, Hans Reschreiter, Andreas Rausch, Ralf Totschnig, Philipp Pichler, Bernhard Heinzl, Johannes Tanzler and Martin Bicher - research in the field of digital archaeology feels a lot like hospital design, but reversed: The structures are there but the people are not. Last but not least I wish to thank my wife, Anke Bacher, who has endured my frantic adventures in science since for more than a decade and hopefully will continue to do so ever after.
What is Agent-Based Simulation (ABS) and what has it got to do with early-stage planning? Over the past years, I have contributed a number of approaches that seek to shed light on these questions, all based upon functional planning: Given a set of functions that declare the purpose of each building space (e.g. dining, cooking), a preliminary floor plan (also called schema, see Figure 1) can be defined. There is a plethora of meanings to be found in such early sketches - from abstract depiction of spatial proximity between functions in two dimensions to definitions of space locations in three dimensions - which is why I have dedicated an exhaustive elaboration on the subject under "Background: Early Planning of Complex Buildings" (page 15, see especially Functional Planning from page 26 onwards). My goal, however, was not to perform a comparative study on schemata, but to help design them. There are two major problems that come up in that context: First, there are many possible options for decomposing spaces which all might seem viable. Second, one cannot compare these options with more formal methods such as Space Syntax\(^1\), since spaces only represent approximate locations, proportions and relationships rather than the “final form” (see Figure 2 for a further comparison). It is nevertheless worthwhile to stay at an early stage of a planning project and to work on these problems, since changes are easy to perform, as opposed to later (i.e. when form has already been detailed): In that case, interventions might prove next to impossible, due to the amount of dependencies between building components resulting in additional costs and delays.

In order to be able to work past the mentioned problems, I have been analysing schemata not by their static arrangement of spaces, but by the temporal order in which these are utilized: As means, I inject agents into the schema and simulate their processes (see Figure 3). The result of the simulation is the overall spatial occupancy, resource utilization (if spaces are considered resources, e.g. examination rooms) and aggregate pathways taken, from which further measures can be derived (e.g. adjacency dictated by the process and causality of activities). Acting on the basis of processes makes the analysis dynamic, where the planner is free to observe building operation at different times and from the viewpoint different building users, spaces crossed and circulations utilized. Special care must be taken as to the interpretation of the results: Some values are quanti-
fyable (e.g. resource utilization) while others are based on properties of the schema that are inherently uncertain and must therefore be understood qualitatively. An example for the latter are the distances covered by each agent, which need to be mapped by methods such as fuzzy logic to linguistic terms (“near”, “average”, “far”) instead of being given as a number, which would be misleading. With this distinction in mind, one can compare different schemata from a performance perspective, meaning the operation of the building as a whole as well as individually, from the perspective of every occupant. If the set of simulated building users is kept identical and only the schemata change, it is even possible to take the comparison as a “metric” (although I would not go as far, since such a comparison would only be based on process performance, and there are far more factors to consider even when given only schemata).

The application of simulation to early (or indeed: earliest) stages of planning will help establish it as design tool in its own right, in addition to the now-common scope of optimization and verification. However, there are some types of buildings that might be better-suited than others when applying the concepts presented herein: Complex buildings - i.e. building types that require constant operation and are planned with processes of building users as prime requirement (e.g. hospitals, airports, industrial facilities and prisons) - form the main area of application for this work. By contrast, the planning of residential buildings is not process-based but tailored around the expected behaviors. This is not the end of the road though, as there are other researcher groups occupied with the latter topic. In sum, current efforts that are underway in the field of agent-based design slowly begin to enter the architecture, engineering and construction (AEC) sector. Referring to the distribution of research teams shown in Figure 4, one can already see a partial exploitation by architecture bureaus (depicted as gradient background of the respective label) which will likely increase in the future. One of the reasons why that is the case might, in my view, be that programming - the base requirement for working with simulation - is becoming a mainstream technology. I am happy

---

2 This distinction is subtle but important: A behavior is a set of activities that users are likely to inhibit, while a process is a sequence. This means that, in principle, a behavior-based user will choose freely between the available activities, while process-based users will follow predefined sequences.

that I could participate in that movement by teaching computer science, programming and simulation for architects, spatial planners and building scientists. My research, which I partly reprint herein, furthermore shows a clear need for further pushing the boundaries of planning practice into a direction where decisions are not based on gut feeling alone. The multi-faceted problem of planning rather demands to explore a variety of scenarios in which “what-if” questions can be asked. This thesis will show that some of these questions can be answered by Agent-Based Simulation, even at early stages of a planning project.

Figure 4: A selection of research groups in agent-based planning. Labels shown with gradient background depict commercial exploitation, where techniques developed in academia are slowly entering the industry. Derivative work bases on Wikimedia “world map” by Jose Lopez.
Notes on style and structure of this Habilitation

The Austrian University Act 2002 states that a habilitation is a “procedure for obtaining the authorization to teach”⁴ which includes the right to “supervise and grade academical works”⁵. As a precondition, the applicant must show “excellent scientific or artistic qualifications” and must have been “teaching at accredited institutions of post-secondary education, as means for proving his/her didactic abilities”⁶. These two categories translate naturally into the first two parts of this thesis, Scientific Excellence (p. 13) and Didactic Qualification (p. 173), after which I also give an Appendix (p. 185) in which I present additional material such as some further notes on conducted projects (p. 187), a Glossary (see p. 199) and a CV (p. 211).

There are two ways in which to write a habilitation thesis: As published monograph or as a cumulative thesis encompassing multiple published papers. I have chosen the latter form, since getting through the publication process for a book would be lengthy and publishing professionally would in turn mean charging money for it, even to cover only the expenses. This, however, would entail further complications concerning copyright permissions (as described further below).

Citation Style

In order to ensure the highest standards with regard to originality and anti-plagiarism, I employ usual standards of in-text citation (citation after first appearance and after relevant passages where they are needed for distinguishing different sources), up to the individual page where needed. Figures that are not originally from myself always cite their would-be source, even though I have redrawn them from scratch - i.e. with no intended resemblance to the original apart from visual style and metaphor. The magic word I use in that case is inspired by⁷, rather than based on or adapted from (both legal terms with severe implications). An exception to this rule is the use of figures within tables, which I always contextualize to include the author or work where the inspirational source was taken from. Furthermore, I have an explicit list of all figures with inspiration source or permission details in the Appendix from p. 203 onwards.

⁴ own translation of UG2002 §1: “Habilitation ist das Verfahren zur Erlangung der Lehrbefugnis […]”
⁵ own translation of UG2002 §1 “[…] mit welcher das Recht verbunden ist, […] wissenschaftliche Arbeiten zu betreuen und zu beurteilen.”
⁶ own translation of UG2002 §2 “[…] Nachweis einer hervorragenden wissenschaftlichen oder künstlerischen Qualifikation sowie der mehrmali gen Lehrtätigkeit an anerkannten postsekundären Bildungseinrichtungen zum Nachweis der didaktischen Fähigkeiten”

⁷ As stated by the Austrian Copyright Act (UrhG §§5 Abs 2), graphics cannot be reproduced lest permission is given, with one exception: If graphical content is used only as inspiration and the reproduced work is fundamentally different to the extent that its resemblance to the original is non-recognizable, no permission needs to be obtained (“freie Nachschöpfung”). What annoys me in that context is that graphics generated in that spirit are not a direct inclusion of a piece of thought of an author, as a text citation would be. I have nevertheless done my best to keep the “spirit” of the original figure while staying clear of copyright issues.
Part I

Scientific Excellence
Background: Early Planning of Complex Buildings

Hospitals, airports and industrial facilities (see Figure 5) are examples of complex buildings - buildings that have to serve a large quantity of users in a 24/7 fashion, employing highly trained personnel and sophisticated equipment to that end. Their planning is process-driven - first and foremost, they have to serve pre-established work routines and organisational structures throughout their operation. By contrast, behaviour-driven buildings are to support user activities without reference to a concrete work plan - e.g. residential buildings and shopping malls (see again Figure 5). The emphasis in this work lies on the first category, since it offers a high degree of formalization on top of which simulation and analysis can be applied. This chapter dives into early-stage planning, giving a background of this quite elusive phase both in terms of the planning project (p. 16) as well as with regards to techniques used (p. 20 onwards).

Figure 5: Complex buildings as highly occupied structures serving the processes of building users in a 24/7 fashion. Own depiction - location of various building types within the graph as estimated by the author.
**Early Planning Is More Than Conceptual Design**

Planning begins far earlier than architects think; it is the client who has to come up with a project in the first place. Some of the tasks involved in setting up a project clearly range into what design models would see as pre-design, others may come well before. Lawson 8 writes:

“In design, the problem usually originates not in the designer’s mind but with a client; [...] The design task, albeit ill-defined, is usually initially generated and expressed by a client.” Bryan Lawson, How Architects Think, p. 84

The American Institute of Architects (AIA) has described the early design process as conceptual design9, listing tasks that are to be conducted in each phase. Taking a closer look at this model (taken as one example among many process models of architectural practice, see Table 1), we note that many of the tasks shown depend on inputs from the client. However, and especially for complex buildings, thinking about objectives for building does not begin with a planning project but far earlier, as part of organizational change management: Identifying areas of change - gradually, through communication within the organization or because of external forces (cross-organisational reconfiguration, new regulations and policies, technological innovations, etc.), employing “Plan-Do-Check-Act” (PDCA) cycles such as the ones given by the Kaizen model10 (jap. kaizen, “change for the better”) and eventually coming to the conclusion that these changes necessitate architectural measures (refurbishments, extensions or completely new buildings).

Even then, it is still a long way before a design project begins. Planning offices may be appointed to conduct feasibility studies, elaborate on possible future usage scenarios and formulate spatial programmes on behalf of the client, which are then compiled together with his own inputs into a design brief. Lawson again:

“In many commercial situations the client may be represented by a professional, acting in that capacity more or less as a job. At the other end of the scale, many buildings are commissioned by people who have never acted as a client before. Sometimes the designer will work with an individual client, and at other times the client body may be represented by a whole committee. In the case of very large buildings commissioned by institutions or companies the programme may last several years and the membership of the client committee may change substantially.” Brian Lawson, How Designers Think, p. 84

This occupation from a client side means that many of the questions being asked in conceptual design (see again Table 1) have already been addressed when a project starts, e.g. objectives, space requirements, limitations, site/building programme and so on. In sum, this leads me to argue for a twofold view of early planning, where design actually starts at the client’s side, is compiled into a design brief or similar statement of requirements and then continues

---

8 B Lawson. How Designers Think. Routledge, 2005


### Phase | Expected tasks (partial)
--- | ---
**pre-design**
- design objectives
- limitations and criteria
- site requirements
- space relations
- initial approximate facility areas and space requirements
- flexibility and expandability, etc.

**site analysis**
- site analysis and selection
- site development planning
- on-site utility studies
- zoning processing, etc.

**schematic design**
- space layout or space schematics
- conceptual site and building plans
- preliminary sections and elevations
- preliminary selection of building systems and materials

Table 1: Conceptual design phases, according to AIA Form B163-1993
at the planner’s side (see Figure 6). That process is lossy: As will be shown later, because intermediate results of the client’s occupation do not enter the design brief, and planners are expected to “inflate” that document again through their own occupation (a circumstance which I have circumscribed with “Big Bang?” in Figure 6). On the one hand, this is a pity - calculations behind requirements such as the spatial programme would, for example, often be beneficial, on the other hand, it leaves room for creativity.

Specialized Design Methods: The Example of Hospitals

As stated, predesign is embedded into the organizational development of the client, and thus, early planning is more than conceptual design. However, the actual degree to which clients are involved and with which these activities are established as part of design methods largely depends upon the field in which planning is conducted. As example, let us here consider the hospital environment (the field I have predominantly worked on). Detailed models of how a project is to be set up in conjunction with the client (usually a hospital or clinical trust) are, for example:

- Hardy and Lammers’ hospital planning and design process\(^\text{11}\) (see Figure 7) starts off with strategic planning - planning of a whole health care region with a long-term vision in mind, conducted by the client. Physical and functional evaluations then looks into status quo of hospital buildings (existing site and buildings; physical condition and functional characteristics), followed by workload projections (future patient volumes, diagnoses, etc.) which lead to a changed functional program. Interestingly, all of these are “prepared by either staff members or consultants specializing in the facility planning” (Hospitals, p. 139), using flow diagrams and schemata for presenting their work. This somewhat antedates the next phase, master site planning, which consists of formulating a space program and doing block drawings (i.e. preliminary site sketches including e.g. landscape details or physical factors, schemata including circulation, bubble diagrams for depicting proximity and flows between units, adjacency matrices and so on). This phase is not conducted by the client, as

Figure 6: Early planning as a two-folded activity happening both at the client side and at the side of the planner, the in-between link being the design brief. Own work.

"experienced functional planners" (*Hospitals*, p. 190) are needed. This applies as well to the next phase, *schematic drawing*, which essentially converts the preliminary sketches elaborated so far into floor plans (quite contrary to the name of the phase).

- The UK Department of Health and Social Security (DHSS) has introduced planning guidelines coined as CAPRICODE\(^\text{12}\) (later extended into the CIM - Capital Investment Manual\(^\text{13}\) based on procedures within the UK National Health Service), which emphasize close cooperation between health authorities and architects from a construction economics perspective. Clinical development is structured into eight stages - *approval in principle, budget cost, design, tender, construction, commissioning and evaluation*, where the client is practically always involved in order to guarantee the best cost/efficiency ratio. To what extent this concerns the actual design is unclear - the client is specifically seen as decision-maker conducting also the verification of handed-in deliverables, all under the general goal of cost control.

- Joedicke’s *Entwurfsmethodik\(^\text{14}\)* is workshop-based method, using interviews and expert groups to coordinate conflicting or contradictory views. The planner is a *facilitator and coordinator of this process*, where clients, building users and professional consultants contribute equally to the design if the problem at hand demands it. The a phase called *problem determination\(^\text{15}\)*, the planning problem is divided into a hierarchical set of sub-problems which can be addressed separately. Problems are also explicitly formulated with reference to the spatial level at which they occur: urban level, site level or building level. As result of the first phase, a hierarchical description of the target state is reached. In the second phase, *generalized solution\(^\text{16}\)*, a variety of solutions to the sub-problems are elaborated, subjected to an evaluation and thereby reduced to a single solution that is ready to be detailed and implemented in the next phase, *solution in context of implementation\(^\text{17}\)*. Because of this rigid structure, this approach is not “participatory” in the now-common sense that it evolves in a bottom-up fashion. Rather, it is the planner who keeps all threads of the project in his hand. It is, nevertheless, a method where clients are equally “experts” as consultants, and projects can in principle begin as soon as a strategical decision has been made (i.e. before a tender).

- Lohfert’s *Methodik der Krankenhausplanung\(^\text{18}\)* takes a similar ap-

---

\(^\text{12}\)* Capricode, 1986

\(^\text{13}\)* Capital investment manual, 1994


\(^\text{15}\)* Problembestimmung, own translation

\(^\text{16}\)* allgemeine Lösungsstufe, own translation

\(^\text{17}\)* Lösungsstufe im Realisierungsbereich, own translation

proach to Joedicke insofar as the planning problem is divided into a set of sub-problems. However, and in contrast to the latter, he uses an project plan technique where higher-level problems are based upon solutions elaborated in lower levels, forming a pyramid-like elaboration structure. The general idea of Lohfert is that each planning problem shall be addressed exactly once, in the correct spot within the project, without needing to be revisited (acyclic problem decomposition, critical path planning).

Starting with regional planning, the method moves to demand planning for a region in which the strategy is laid down, leading to planning of specialized portfolios of clinics, implemented through functional planning, space programming, process planning and so forth, ultimately ending in the construction and provision of a clinical area, a building element or a whole clinic. Like Joedicke, Lohfert sees clients as consultants for their area of expertise (administration, budgeting and finance, personnel, etc.). There is no statement as to the extent of this involvement, as his work rather emphasizes the decomposition of the planning project rather than explicitly stating roles and responsibilities, which are to be assigned by the project management.

As can be seen, the diverging scope of the selected design methods for this field (and the fact that many more are available also for the general case, both regionally and internationally), it is fair to say that there is no lack of explanatory models. The remaining question is, however, whether these models can actually help in fostering a culture of simulation in early-stage planning, which is the main task I was faced with. Clearly, understanding the client is central to establishing simulation in the first place. Even in a single institution (hospital or hospital trust, the latter is reprinted from p. 65 onwards), the exact place within the project in which these efforts take place can diverge. Luckily, and unlike the location within the project, the deliverables used are quite uniform - which leads me to take a more elaborate look at them as next step.

**Occupational Media in Early-Stage Planning**

“**When**” early-stage design happens exactly within the initial stages of a project can diverge, but “what” is generated is surprisingly clear. In this section, different media through which occupation with the subject happens are scrutinized. I categorize them into either site analysis or functional design, for reasons best given by Edward T. White in his book “Site Analysis”:

“It is useful in discussing the influence of contextual analysis on design to differentiate between function and context as forces which locate building spaces and activities on the site. Function tends to locate building spaces in an introverted way in that they are primarily looking inward to each other for the rationale behind their positions in the scheme. Context, on the other hand, wants the spaces to migrate to different positions on the site in response to conditions
outside the building. In function, the attraction is between spaces. In context, the attraction is between spaces and external site conditions. Usually in a design problem these two (and all the other) project issues pull and push the spaces to determine their final placement in the scheme. They are in a very real sense competing with each other to determine the building form.” Edward T. White, Site Analysis, p. 22.

The concluding occupation with the design brief (p. 45) is motivated by the urge to verify the occurrence of the presented media in actual planning documents, of course with the hidden agenda to argue also for the two-fold nature of early-stage planning presented earlier.

Site Analysis

Site analysis is an activity that precedes the actual design; White defines it as contextual analysis, a pre-design research task that analyses site location, size, shape, contours, drainage patterns, zoning and setbacks, utilities, significant on site features (buildings, trees, etc.), surrounding traffic, neighborhood patterns, views to and from the site and climate. As he notes, a site is never an inert plot but “a set of active networks that are intertwined in complex relationships” (Site Analysis, p. 8, see Figure 8).

Figure 8: Site as set of active networks. Own work inspired by White 2004, Site Analysis, p. 8

Since I target complex buildings, I assume that a lot of contextual information is to be considered before beginning the actual design. I now give an enumeration of possible diagram types used for analysis, based on examples from White and other authors quoted in the text. In favour of an explanation that is focused on each problem area, I use diagrams segregated by type of displayed information rather than composite views. Also, as White’s diagram classification has many overlaps and is also quite repetitive, I reclassify...
according to the properties being depicted. Note that for simulation, *time-dependent* diagrams are of special significance (see also the paper “Meeting Simulation Needs of Early-Stage Design through Agent-Based Simulation” from p. 53 onwards):

- **Location.** The site can be approached at different scales: Regional level, city-scale, neighborhoods and parcels, at the building level and, even smaller, at the level of individual spaces\(^{26}\). Figure 9 then depicts the site itself inside its neighborhood, which in this hypothetical case is assumed to be situated around Schwedenplatz Square, close to Vienna’s center. Distances to areas related to the site are given in Figure 10 - e.g. the distance to the airport or the largest shopping street.

- **Static Features.** Natural and man-made features found at the site include e.g. contour lines as base layer\(^{27}\), a vegetation layer (left in Figure 11) and the existing built environment on top of that (right in Figure 11). Furthermore, views from and into the site (see Figure 12) also belong to the class of static features, even though one could argue that these do change over a longer course of time. The same argument would, by the way, also apply e.g. for the vegetation layer. We can see from this that what is assumed as static largely depends upon standpoint and time interval during which the planning problem is regarded.

- **Dynamic Forces.** Time-dependent properties such as physical factors - e.g. sun and shading (Figure 13, left), noise (Figure 13,
middle), wind/pollution (Figure 13, right) or climate are hard to depict. Their aggregation can be two-fold - first by season (in the case of the sun) and second by time of day, which can hinder the understanding of such diagrams. Consequently, it is typically the case that a time series of these is made, in order to show a progression. An even more complicated case occurs when the analyzed property represents dynamic system itself. Consider, for example, the analysis of exterior lighting as given in Figure 14. Here, a man-made system (street lighting) interacts with a physical force (sun), possibly already exhibiting a dynamic
behaviour. To record such a system on a diagrammatic basis might be hard, to say the least.

- **Circulation, Traffic and Movement Patterns.** Aspects of the circulative system, together with its use, are subject for these types of diagrams. Although being seemingly static, the underlying model is a very dynamic one: Travel times, for example, depend on distances (see Figure 15), means of transport (Figure 16), usage of the transportation system at a certain hour and so forth (Figure 17); they depict an aggregation, with added labels to indicate at which time what event happens. For example, White notes that there is “high traffic in morning and late afternoon” (right in Figure 15), which is problematic when many such annotations exist at the same time (speaking of complexity). Furthermore, because of visual cluttering, it is not possible to combine such diagrams easily - and consequently, it is often the case that there are multiple diagrams for each type of actor or property depicted.

Figure 13: Physical forces shown as a snapshot (left to right): Sun-shadow patterns at March 21st, Noise during rush hours, airborne pollutants in winter. Own work inspired by White 2004, *Site Analysis*, pp. 64, 98, 99

Figure 14: Example of coupled system (exterior lighting). Own work inspired by White 2004, *Site Analysis*, p. 60
Figure 15: Distances and travel times between site and related locations. Own work inspired by White 2004, Site Analysis, p. 47.

Figure 16: Circulation (left to right): Vehicular, pedestrian. Note that there are two separate diagrams split by involved actor, because of visual cluttering that would occur when overlaying. Own work inspired by White 2004, Site Analysis, pp. 57-58.
**Regulative and Legal Issues.** The zoning regulation, existing and projected surrounding property (e.g. age and condition, present and future use) and many more criteria act as guidelines and constraints for later planning work. Figure 18 shows an example of a zoning diagram with added building heights. Note that the zoning codes were mapped from the Vienna municipal codes to the U.S. codes which Edward T. White uses, in order to maintain some degree of consistency with our inspirational source. Such legislative aspects, albeit changeable (projected use, etc.), are usually given as plan documents or maps within Geographical Information Systems (GIS). An interesting aspect for planning is that future development of an area might include several (alternative) scenarios, which might be taken into account.

<table>
<thead>
<tr>
<th>Approximate mappings:</th>
<th>Vienna Zoning</th>
<th>US Zoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB (mixed)</td>
<td>B1 (general business), R3 (mixed residential)</td>
<td></td>
</tr>
<tr>
<td>GBGV (mixed commercial)</td>
<td>B1 (general business)</td>
<td></td>
</tr>
<tr>
<td>EKZ (shopping mall)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 17: Traffic patterns (left to right): Generators, axes and throughput. Information is aggregated over the course of a day. Own work inspired by White 2004, *Site Analysis*, p. 56

Figure 18: Example of neighborhood context analysis showing zoning and permissible building heights, own work inspired by White 2004, *Site Analysis*, p. 49

---

**Functional Planning**

**Classical Architectural Planning** - the one that is still taught in basic design studios - is functional: User requirements are gathered in the form of intended *activities*, which are to be served by the building’s programme via spatial units that offer the necessary capabilities. These units, called “functions”, have been at the core of architectural debate ever since the term was adopted in late
baroque, most notably by the Franciscan priest Carlo Lodoli (1690-1761): He regarded components serving no specific purpose (il suo proprio ufficio29) as non-integral, due to be excluded. This makes him an early predecessor of rationalist architecture, most notably of Horatio Greenough (1805-1852), who in turn influenced Gottfried Semper (1803-1879) in Germany and Louis Sullivan (1856-1924, form [ever] follows function30) in the U.S.

In this line of thought, what function is entirely depends on functioning of an entity, i.e. the congruence between intended activity and planned capability of the spatial programme. That this congruence is hard to achieve is evident, as there are many open ends to this quite elusive concept. Quoting Jan Michl from the Oslo School of Architecture31:

“On the one hand we tend to consider functional perfection to be an obvious, if difficult, aim of the engineer’s and designer’s effort. But on the other hand we have a creeping suspicion that such an aim is hopelessly out of reach.” Jan Michl, On the Rumor of Functional Perfection.

I now bring a description of the classical functional planning method with its five consecutive steps, before detailing out exactly why it is so hard to reach a congruency with this approach.

**From Activities to Schema.** In principle, I adopt the functional derivation as proposed by Edward T. White’s “Space Adjacency Analysis”32 which specifies functions, relates them using matrices and then groups them in a process named zoning. White published his account over how functional design proceeds in 1986, however, the method employed is not new. It has been around far earlier as Activity Data Method33, in which activities are specified as user requirements before functions come into play. White does not mention this activity specification phase altogether, so that we have opted to include a work that does: Schönfeld’s “Gebäudelehre”34 is a work which is explicitly based on the Activity Data Method, and provides invaluable insight into that area. Another topic that is only covered in brief in White’s work is the finding of an architectural schema/preliminary floor-plan as last step of the early design process. White uses this type of diagram fairly often, but does not properly introduce it nor define its constituents.


![Functional derivation, from activities to schema.](image_url)
In order to gain some rigor in the description of that specific diagram type, I have chosen to contrast his form with that found in Ernst Neufert’s “Bauentwurfslehre”\(^35\) and its English translation “Architect’s Data”\(^36\), in which the schema features as central diagram type for early design. Hence, I now give a description of the early functional design process, using graphical examples in the spirit of the mentioned books. More specifically, a derivation from activity to preliminary floor-plan is conducted, which goes through five consecutive occupational media (see Figure 19): activities which are aggregated into functions within the functional program, related to form the relationship matrix, transformed into a bubble diagram, after which bubbles are again aggregated to form zones within the schema. The space allocation plan is intentionally left out of this description, since it is just a spreadsheet of zone names and square meters; it is typically developed in parallel to the schema, i.e. when space requirements of individual zones are set. However, some authors\(^37\) start defining space requirements already with the development of the functional program, which is fine as well.

1. Activities. To begin with, activities that must be supported in daily building operation are specified together with the client. They act as requirements for the later spatial programme, and need to be defined at the same degree of detail (spatially and semantically). If this is not possible (e.g. multiple spatial levels exist), the activity specification is iterated for each such level. A further discussion on this topic is given later, under “Degree of Detail and Hierarchies” (p. 39).

Table 2 gives an overview over how activities are elaborated and depicted. Note that Neufert does not include an activity specification phase, which is probably due to the assumption that a space allocation program is already given (and thus, the workflow starts at a later stage). White specifies activities verbally, with no added nomenclature or level of analysis. Only Schönfeld approaches the subject from a process viewpoint: The activities are enumerated by involved actor, then ordered temporally to form activity chains. The origin of this way of thinking lies in what we would nowadays call Taylorism, i.e. the theory of analyzing and optimizing workflows by means of time and motion studies\(^38\) which was published in 1911. This theory rapidly caught foot in architecture, leading e.g. to the invention of the modern kitchen (1913 by Christine Frederick\(^39\) in the U.S. and a decade after that by Margarethe Schütte-Lihotzky in Europe; the latter design was later known as “Frankfurt Kitchen”, the first fitted kitchen and today the de-facto household standard\(^40\)).

Apart from temporally ordered activity chains, we today also employ causal order (i.e. if condition then activity a else activity b as in business processes), categorization into core activities and support tasks and so on. What remains unchanged, however, is the notion of activity used in planning: An activity is a recurring

---


\(^40\) M Schütte-Lihotzky. Warum ich Architektin wurde. Residenz-Verlag, Salzburg, 2004
task in the operational routine of a building, which is carried out by one or many building users (i.e. actors). In contrast to that, building users might also exhibit behaviour that cannot be enumerated in a straightforward manner, such as in the case of evacuation or shopping: Its outcome depends on the internal state of the user rather than a prescriptive sequence of activities to be followed (which is also how this is often simulated, as agents with internal state machines).

<table>
<thead>
<tr>
<th>Activity Specification</th>
<th>Symbols employed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neufert:</td>
<td>-</td>
</tr>
<tr>
<td>Architect’s Data/</td>
<td>-</td>
</tr>
<tr>
<td>Bauentwurfslehre</td>
<td>-</td>
</tr>
<tr>
<td>White:</td>
<td>Verbally, e.g. sitting, reading, telephone, eating</td>
</tr>
<tr>
<td>Space Adjacency Analysis</td>
<td>-</td>
</tr>
</tbody>
</table>

2. **Functional Programme.** Architects like to speak in terms of functions, a term subtly different from the notion of activities: Activities refer to the process of building users and describe what is done (in which order, in the case of activity chains). Functions, on the other hand, stand for the purpose of spaces, defining where activities will happen. The important point to consider is that functions are at first defined purely abstract, as if a planner were saying: “We will need a space in which a user can do this-and-that”. Through application of the functional planning method

![Diagram](image-url)  
**ACT.**

Table 2: Activity Specification
described in this chapter, they are consecutively enriched with information until they end up being inscribed in concrete spaces. Because activities and functions are different, there needs to be an intermediate mapping step. White describes this transformation as follows:

Although we are interested in the specific activities of our client’s operation and their functional interrelationships, for most projects we usually simplify our analysis task by listing the names of the building spaces in the matrix rather than the names of all the activities they contain. Edward T. White, “Space Adjacency Analysis”, p. 37.

Functions are defined in a much broader context than activities. Therefore, we can see functions as aggregate activities (e.g. eat and drink are summarized as cooking, see left part of Figure 19). One of the downsides of this surjective mapping is that, consequently, functions become detached from their activities. A discussion as to why this can pose problems is given in “Congruence Between Activities and Functions” on p. 41.

3. Relationship Matrix. The functional programme defines the set of all functions in the building. It does, however, not give any indication over which functions are related as by some common purpose, intended proximity or because of close cooperation within a process. Such relationships between pairs of functions are recorded as (half-)matrices (also see Table 3), where each matrix stands for a specific type of information. For example

- Interaction Matrices for defining a degree of cooperation (undefined, often, average, seldom, n/a) in the overall operation of the building.
- Adjacency Matrices for intended adjacency relationships (mandatory, desirable, neutral, negative).

The actual value of the relationships is visualized using symbols, colors and/or numbers, as outlined in the column “Symbols employed” in Table 3. As can also be seen in that same table, one can also group functions into larger sets (e.g. functions of a specific department, as given by White). While this can indeed help categorize the adjacency matrix, it introduces a level of hierarchy that may conflict with the zoning introduced in the next step. A more elaborate look at this problem is given in “Congruence Between Relations” on p. 43.

4. Bubble Diagram. By introducing a node (“bubble”) for each function and an edge for each relationship, the relationship matrix is transformed into a graph. The edge style depicts the value of the relationship (e.g. thin/thick, dashed/dotted/solid, color of edge). Referring to Table 4, we can see that bubbles can also be styled to depict relative importance or space consumption. White’s style of drawing the diagram is the most “standard” way (bubbles size depicts relative space consumption, edge style gives relationship value). By contrast, Schönfeld uses directed relations -
<table>
<thead>
<tr>
<th>Matrix</th>
<th>Symbols employed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half-Matrix</td>
<td></td>
</tr>
</tbody>
</table>

Neufert: *Architect’s Data/Bauentwurfslehre*

Full Matrix

- intensive
- frequent
- occasional

White: *Space Adjacency Analysis*

Grouped Half-Matrix

<table>
<thead>
<tr>
<th>Relationship Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>mandatory</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>A B C D</td>
</tr>
</tbody>
</table>

Schönfeld: *Gebäudelehre*

Grouped Half-Matrix

<table>
<thead>
<tr>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>spatially connected</td>
</tr>
<tr>
<td>▲</td>
</tr>
</tbody>
</table>

Table 3: Relationship Matrix
Table 4: Bubble Diagrams
hinting at the use of a full relationship matrix as basis. Neufert’s use of the edges is directed as well, but additionally, they depict flows between functions. A second oddity with Neufert’s way to use bubble diagrams is that he uses them spatially (refer to Table 5). This is very unusual, since this diagram type should be purely abstract (the location of the bubbles does not matter; the only notion of “spatiality” is the previously mentioned attribution of sizes to bubbles, if these stand for relative space consumption).

Bubbles are grouped according to a common characteristic - typically the adjacency relation defined earlier in the matrix, leading to a definition of co-located areas called zones. As an effect, we end up with each bubble being present in a zone, if bubbles not being present in any zone implicitly define one.

Table 5 gives concrete examples of zoned bubble diagrams: In all shown cases, zones are drawn as solid or dotted enclosures around bubbles. As is also clearly depicted, zones can be nested within each other, giving a hierarchy that may also carry additional meaning (e.g. building level, department level, room level). Depending on whether they can overlap or are strictly disjoint, we talk of either a semi-lattice or a tree hierarchy (see also p. 39 for a more elaborate discussion).

Interestingly, planners may also choose to group by criteria that have not yet been introduced. For example, White mentions operational groupings into an administration zone, production zone and support zone; grouping into open or closed spaces - spaces not requiring interior walls, spaces requiring interior walls; public (client) zones and private (staff) zones; zones of permanent occupancy/zones of intermittent occupancy. Essentially, these novel grouping criteria are additional relationships, not between functions alone but between groups of functions.

5. Architectural Schema. Zoning yields a variety of possibilities for partitioning the bubble diagram. For every such possibility, a preliminary floor-plan (schema, refer to Table 6) is generated by utilizing a set of transformation steps. In "Architect’s Data", Neufert enlightens us how this is done:

The sketch scheme is begun by drawing up individual rooms of the required areas as simple rectangles drawn to scale and put provisionally into groups. After studying the movements of the people and the goods (horizontally and vertically), analyse circulation and the the relationships of rooms to each other and the sun. Ernst Neufert and Peter Neufert, “Architect’s Data”, p. 40.

While this may be true when the space allocation program is given, in the general case, a more formal way is to derive the “rectangles” from the data of the previous stages:

- Every zone of the bubble diagram becomes a rectangle within the floor-plan. A caveat: Rectangles act not as means to express form, but instead as placeholders; the emphasis is on their relative placement, orientation and size. The finding of a form for
### Zoned Bubble Diagram

<table>
<thead>
<tr>
<th>Neufert</th>
<th>White</th>
<th>Schönfeld</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Neufert Diagram" /></td>
<td><img src="image" alt="White Diagram" /></td>
<td><img src="image" alt="Schönfeld Diagram" /></td>
</tr>
</tbody>
</table>

#### Symbols employed

- **Bubble**
  - ![Symbol](image)
  - ![Symbol](image)
- **Relation**
  - **strong**
  - **average**
  - **low**
- **Zone**
  - ![Symbol](image)
  - ![Symbol](image)

#### Zone e.g.
- ![Symbol](image)
- ![Symbol](image)
- ![Symbol](image)

### Table 5: Zoning
the zones is deferred to a later stage (called form finding phase), which naturally represents the end of early planning and is therefore not covered here.

To make this separation even more clear, let us take the example in Figure 20, which show an entrance zone being situated in the south of a schema. From this, we may read the orientation (south), its relative size and position to all other zones. Through form finding, the zone may later be implemented as an open area with no form at all, or by a curved entrance area with an attached cafeteria. Likewise, two zones being located side by side in the schema might finally become spaces in two consecutive floors, connected by a vertical access (e.g. staircase, lift), a single space or two separate spaces in the same floor.

- The functions that make up the zone are inscribed into it, either centered or located relative to its parent zone (as in the case of Neufert, shown in the first image of Table 7).
- The hierarchy of zones becomes a hierarchy of rectangles-within-rectangles (also see “Degree of Detail and Hierarchies”, p. 39). There is a further hierarchy introduced when zones overlap, as for example found in the topmost image in Table 7: In this specific case, a function is shared between the two underlying zones. In my own research (reprinted from p. 65 onwards), further options for dealing with overlaps were elaborated with reference to schemata generated in the planning of an 800-bed hospital in Lower Austria.

First and trivially, we regard strict containment (i.e. no overlaps). If we take the styles used for depicting the zones, we can distinguish between two types - zones that act as physical boundary (solid, see left in Figure 21) and zones that exist only conceptually (e.g. departments, inpatient/outpatient areas etc., see right in Figure 21). Such flavours of boundaries can be treated differently, e.g. during a simulation, when route choice is determined. Overlaps can be one of two cases:

(a) an overlap where a zone is subordinate to/being used by other zones, such as in the left part of Figure 22 (B is used by A and C).

(b) an overlap where a zone is shared by two zones as given in the right part of Figure 22 (B shared by A and C).

Such overlaps are transitive, as shown in the left part of Figure 23: If B overlaps A and C overlaps B, then C is also overlapping A indirectly through B. Another property of overlaps is that it also applies to functions, if these are positioned relatively and are thus are meant spatially, as mentioned earlier. The right part of Figure 23 gives an illustration: In this case, B is a function being shared between zones A and C. The inverse case - two functions A and C being used by zone B - is only possible if zoning is overlapping.

\[\text{Figure 20: Example Schema}\]

\[\text{Figure 21: (left) nesting (right) conceptual containment}\]

\[\text{Figure 22: (left) subordinate (right) shared zones}\]

\[\text{Figure 23: (left) transitivity (right) shared function}\]
As additional element of the schema, the Circulation is introduced as either a set of axial lines (e.g. with arrows as in Neufert and Schönfeld, see Table 6) or, in resemblance to zones, as a set of "circulative areas" that are also depicted as rectangles (in White and Neufert). Furthermore, Thresholds may be explicitly defined, for cases where one wants to mark the spot in which the circulation enters into a zone. The usual symbol is a triangle pointing inwards to the zone being entered.

Apart from two-dimensional schemata, there is also the case of three-dimensional schemata (see Table 7). It is questionable whether these really constitute preliminary occupational media, since vertical zone locations seem to be already fixed. The argument would then be that such depictions are more like floorplans (reduced perhaps to show not spaces in their concrete form but rather in a more abstract way, for example for way-finding in a building). My proposition for a resolution of this conflict would be the following: There is nothing wrong with using that diagram as a means for a rough vertical decomposition. However, the different levels should not be confused with final "floors" - they are rather a statement on vertical stacking that is simply not possible in the two-dimensional schema. This notion of preliminary positioning remains also for zones within each level.

Discussion, or: Abstract to spatial, not defining form. Functional design uses diagrams to depict structure (i.e. relationships between individual functional constituents, relative positions and sizes of zones) rather than giving hints at geometrical form. Starting with abstract functional relationships, the method gradually adds more and more spatial information (the first time in zoning - if zones represent spatial grouping, and later in the schema). Form is deferred to a later phase called "form finding", which reformulates the preliminary design in full geometrical detail, which marks the end of early-stage planning and the begin of the actual form design.

The absence of form from functional design, in line with Sullivan's catchphrase ("function" as cause and "form" as effect), has been drastically overemphasized. Purists like the Swiss architect Hannes Meyer claimed that form would altogether be derived directly from function and site, or in his words\(^\text{47}\):

\[\text{We examine the daily routine of everyone who lives in the house and this gives us the function diagram for the father, the mother, the child, the baby and the other occupants. we explore the relationships of the house and its occupants to the world outside: postman, passer-by, visitor, neighbour, burglar, chimney-sweep, [...] we calculate the angle of the sun's incidence during the course of the year according to the latitude of the site. with that information we determine the size of the shadow cast by the house on the garden and the amount of sun admitted by the window into the bedroom [...]}\]
Table 6: Schemata
Table 7: 3D-Schema
the new house is a prefabricated building for site assembly; as such it is an industrial product and the work of a variety of specialists: economists, statisticians, hygienists, climatologists, industrial engineers, standardization experts, heating engineers ... and the architect? ... he was an artist and now becomes a specialist in organization!"

Hannes Meyer, bauen, 1928, translation of the original text according to www.splankin.com/webdesign/arch346/Resources/Web/hm_building.html [accessed 14.03.2013]

It is true that functional design is a rather “scientific” derivation of a schema from activities, in contrast to design, which is arguably a more complex and creative approach and regards the building project as a whole. It is not true, however, that there is no rigour in the latter - it is only argumented differently, in design meetings, through analogies (formal or abstract), and so on. “Design” as such does not adhere to “a method”. “Design methods” are a framework - similar to how music theory approaches composition. But would a single composer benefit from knowing that he fits into a certain style when trying to create a piece? Probably not, although knowing about the general context is certainly helpful.

On the other hand, a functional design can always be attributed a posteriori. An (arbitrary but illustrative) example is the work of Günter Bacher, who reverse-enginiers crabs and gives individual organic structures found there a qualitative/functional meaning. Also, one can always go back from form to functional depiction - which is often done for signage (e.g. an overview map in a building) or in the course of reconstruction/enhancements of an existing building. This has the advantage that spatiality can be added to diagrams where they would not have been present originally: Referring to Table 5, we can for example see that Neufert is using his bubble diagram in a manner that resembles a floor plan, even though the positioning of zones would only come later (during creation of a schema or floor plans). Likewise, zones and physically colocated areas could be also shown as groups within the matrix (refer to Table 3). Such a “back and forth” between later and earlier stages is good - it contributes to a holistic understanding of the building rather than an isolated, inductive build-up of a structure.

Degree of Detail and Hierarchies. To maintain a consistent degree of detail is crucial for a functional decomposition; hierarchization of the planning task into multiple levels of increasing detail (e.g. site > building > department > workplace) is the usual approach, however, such hierarchies can be used not just for functions but for all problems addressed in the building project. For example, the architect and mathematician Christopher Alexander decomposes a planning problem recursively into sub-problems, until each sub-problem can be solved in isolation. This can be visualized as a problem decomposition tree, given in Figure 24. Each sub-problem is solved in isolation, the solutions are merged to solve the overall planning task. A merge step is not done in a straightforward, additive way. Solutions are composited such that the solution in

48 “wir untersuchen den ablauf des tageslebens jedes hausbewohners, und dieses ergibt das funktionsdiagramm für vater, mutter, kind, kleinkind und mitmenschen. Wir erforschen die beziehungen des hauses zu seinen insassen und zum fremden: postbote, passant, besucher, nachbar, einbrecher, kaminfeger, [...] wir errechnen die sonneneinfallswinkel im jahresverlauf und bezogen auf den breitengrad des baugeländes, und wir konstruieren danach den schattenfächer des hauses im garten und den sonnenlichtfächer des fensters im schlafzimmer [...] das neue haus ist ein industrieprodukt und als solches ein werk der spezialisten: volkswirte, statistiker, hygieniker, klimatologen, betriebswissenschafter, normengelehrte, wärmetechniker, ... der architekt? ... war künstler und wird ein spezialist der organisation!”

50 Space Adjacency Analysis, p. 36

the next-higher level includes the solutions of the lower levels. The form of the solution in the next-higher level, however, may deviate from the individual solutions at the lower levels - it is “more than their sum”, to be designed with a holistic view in mind.

The decomposition into a tree structure is not meant spatially - it is a decomposition of issues and constraints rather than physical boundaries. Alexander argued for the need to understand space not by disjoint sets, but overlapping ones (this is called a semi-lattice, as shown in Figure 25).²

“Whenever we have a tree structure, it means that within this structure no piece of any unit is ever connected to other units, except through the medium of that unit as a whole. The enormity of this restriction is difficult to grasp. It is a little as though the members of a family were not free to make friends outside the family, except when the family as a whole made a friendship. [...] For the human mind, the tree is the easiest vehicle for complex thoughts. But the city is not, cannot and must not be a tree. The city is a receptacle for life. If the receptacle severs the overlap of the strands of life within it, because it is a tree, it will be like a bowl full of razor blades on edge, ready to cut up whatever is entrusted to it. In such a receptacle life will be cut to pieces. If we make cities which are trees, they will cut our life within to pieces.” Christopher Alexander, A City is not a Tree - Part II

These thoughts are in line with earlier findings on social perception of space by Lynch³:

“There seems to be a public image of any given city which is the overlap of many individual images. Or perhaps there is a series of public images, each held by some significant number of citizens.”

Kevin Lynch, The Image of the City, p. 46

In a replique to Alexander, John Minett formulated his article “As the City is not a Tree... it should not be designed as a System”⁴. Although the article was originally targeted at city planning, the critique formulated within is applicable also to bubble diagrams presented earlier:

“Investigations into the way designers design show that they go through a series of definite stages. The first is concerned with clarifying the problem by breaking it down into its component parts or functions and identifying their requirements. The second is to gather ‘like’ components together into ‘sets’ (i.e. collections of elements which are thought to have something in common). The third stage is to link those components which work together into a system (i.e. a set of components which are linked together in order to perform a function). [...] the designer is responsible for ordering the relationships in the system, or between systems and sub-systems, to satisfy functional and aesthetic requirements. The links which make the system are designed to maintain equilibrium between the components, future change being controlled by the links. Consequently, the designer is in a very powerful position. He is creating frameworks relative to the way he thinks they should be. When he creates a system he is not only providing opportunities, but also constraints. His design will stabilise a set of relationships allowing only changes acceptable to the structure. Consequently he must be sure that the relationships

---


⁴ Minett, J. (1975). If the City is not a Tree, nor is it a System, Planning Outlook New Series, 16:4-18, quoted after http://www.rudi.net/books/202 [accessed 19.03.2013].
he sees are valid, and the links he creates are needed. Identification of a set does not automatically make it a system; it is the links that create the system.” John Minett, As the City is not a Tree... it should not be designed as a System.

Minett goes on to speak about the assumptions on which this approach is based:

• “That the [design] problem can be defined, in terms of agreeable objectives”.

• “That the components can be isolated and their requirements analysed”.

• “That there is a ‘best-fit’ relationship”, i.e. a configuration which solves a problem.

• “That the end product can be achieved in reality, because design takes account of the variables in the control of the designer and client”.

For me, these questions relate to the concept of atomicity, an issue that was already addressed half a century earlier by Betrand Russell, Alfred North Whitehead and Ludwig Wittgenstein, who argue for the existence of “simple” undividable facts (atoms), out of which more complex statements are built. Yet, as Russel adds55:

“‘simple’ must not be taken in an absolute sense; “simpler” would be a better word. Of course, I should be glad to reach the absolutely simple, but I do not believe that that is within human capacity”

Betrand Russel, Dr. Schiller’s Analysis of the Analysis of Mind, p. 646

The same view is also brought by Wittgenstein, who opposes the idea of an ultimate metaphysical “simple level” as either nonsensical (Unsinnig) or senseless (Sinnlos), since this level would fall beyond the limits of our language, and could thus neither be expressed nor comprehended56. Similarly, I have argued in previous work57 that “atomicity applies only at one spatial level”, and components defined there stand for the “terminology at that level”. That this idea of keeping everything at the right scale or granularity is indeed not simple when doing functional design will be elaborated in the next sections.

Congruence Between Activities and Functions. If functions are derived from activities (as stated earlier), it is interesting to look exactly how this is done and if these two concepts fit together:

• An activity is an (observable) act at a certain location in space and time having a duration. Activities executed in sequence form a process, i.e. a temporal and causal ordering of activities.

• A function is an attribution of a (bounded) space. Architects use the term more in the sense of purpose (or intent58), meaning that a user can perform a certain activity if there is a function supporting it. Referring to Figure 26, we can for example see that


the activity “sit” is using the function “waiting” standing for a waiting area in a clinic.

Are function and activity defined at the same level of granularity, or formulated differently: Are “sit” and “waiting” terminologies at the same spatial level? The explanation in Figure 27 shows that they are in fact not: Let us assume that in the “waiting” area there is also the possibility to “park a bed”. This makes the very notion of function ambiguous, and ill-defined. In previous work \(^5\) (see p. 109), I have therefore been working on atomar functions (called \textit{capabilities}), which are defined as the spatial counterpart of activities. Summing up this work briefly, if there is an activity “sit”, then there must also be an atomar function “can sit”. The benefit of this approach is that activities and atomar functions form a 1:1 relation, which means that they can be derived directly. Non-atomar functions can still be used, as given in Figure 28:

1. The basic building blocks we are using are “capabilities”, i.e. atomar functions. These can be derived directly from the intended activities (“park bed” > “can park bed”). As side-note, this enumeration of activities works only for \textit{process-driven buildings} - these are buildings in which the design is subordinate to the work processes (to be specified first-off). Examples of these building types are hospitals, airports and industrial facilities. Conversely, an example of a building type where this is not feasible would be residential housing, because building users are not following a routine of defined activities but are more \textit{behaviour-driven}.

2. A non-atomar “function” is then a composition of one or more atomar “capabilities”. We can see that these are not disjunct: “waiting” and “bed processing” both share “can park bed”. This is not a problem, since atomicity is kept under the hood. Going even further into detail, we could hide the atomar capabilities altogether from the designer: He could continue working with “functions”, from which we always could get back to the underlying “capabilities” via a mapping similar to the example. The benefit for nevertheless having them is twofold: Defined activities that have no reference to a “capability” do not make sense - one can not “sit” without a chair that offers “can sit”. Vice versa, a capability that is never used is useless (if nobody “sits” then...
why offer “can sit”?). This bidirectional check keeps activities and functions congruent.

Ambiguities in Schemata. Schemata typically lack a depiction of relationships between zones, in correspondence to the edges of a bubble diagram. However, such relationships nevertheless exist, which I want to exemplify using the example in Figure 29: Let us assume that two types of relationships among functions exist, positive (green) and negative (red). Let us further assume that the two functions “Surgery” and “Recovery” end up being in the same zone, while “A/E” unit has its own zone. Then, we could get the relationship between the two zones by aggregating the relationships of the underlying functions. However, this is not possible, since we have both positive (A/E-Surgery) and negative (A/E-Recovery) types between them, leading to an ambiguous relationship (“?”) that exists even if we do not depict it.

One possible method for resolving this issue lies in using only one type of relationship - “related”, from which we can infer the relationship between zones even for hierarchical cases, as shown in the next section (“Congruence between Relations”). The other method would be to explicitly redefine the relationship between zones, resembling Aristotle’s quote “the whole is greater than the sum of its parts”. It is still the question, though, if that solves the problem or just defers it to a different level of abstraction, introducing new ambiguities.

Congruence between Relations. As stated earlier (p. 30), functions in a relationship matrix may be grouped. Not always are these meant in the sense of zones - sometimes these are just organisatorial divisions without any spatial meaning. However, if grouping indeed antedates zoning, both need to to be kept consistent:

- Referring to the left part of Figure 30, we see that the grouping into “Z1” and “Z2” leads to equivalent zones in which the same functions are present.

- If this equivalence is not kept, i.e. one chooses zones that have no correspondence to the groups, then both notions of grouping contradict each other. Referring to the right side in Figure 30, Group “Z1” has functions F1-F5 and “Z2” has F6-F12, while the

Figure 29: Ambiguous relationships between zones. (left) Relationship matrix transformed into a (middle) bubble diagram. “Surgery” and “Recovery” are put into a common zone. One of them has a positive, the other one a negative relationship to the “A/E Unit”. When transformed into a schema (right) it is unclear what the “overall” relationship between the two zones should be.
zones in the bubble Diagram have F1-F9 and F11-F12. Such a contradiction may lead to misunderstandings among the planning team - some planners talk about groupings of the matrix while others refer to zones. Further on in the project, it may turn out that either of these classifications are poorly reflected in the final design.

Consistency is even more important for hierarchical relationship matrices and hence to hierarchical bubble diagrams, as shown in Figure 31. Assuming that we have only one relationship type, “related”, we can build up a consistent matrix in a bottom-up fashion. On the base level, we have the functions and their respective relationships. We then group these into zones Z1 and Z2 and are interested in inferring (not defining!) their relationship. The relationship is the union of all relationships between Z1 and Z2, or simpler: If there is a relationship between their contained functions, then they must be related as well.

For relationship matrices containing more than one quality, such an inference is not possible due to the ambiguities mentioned in the previous section. In this case, only a separate definition of relationships at different hierarchical levels is possible, which is problematic.

"When we use multiple matrices in a project, there is a danger that we will overlook the functional relationships between buildings or

---

between departments that have been listed on separate matrices. This problem can be eliminated by including a master matrix that lists all the buildings or departments or divisions so that we can study their adjacency requirements as well.” Edward T. White, Space Adjacency Analysis, p. 38

While I do agree with the first part of his quote, namely that there is a danger in overlooking functional relationships between matrices at different hierarchical levels, I cannot see that adding another level of relationships will help tackle the problem of redundance.

Occurrence of Occupational Media within the Design Brief

As a document stating the client’s intent, the design brief forms a pivotal point at the middle of early-stage planning, separating the “client-led” from the “planner-led” stage. Thus, the assumption was that some of the presented occupational media might be present in there, acting as requirements for the building project.

Because of a lack of previous work on that subject, I conducted a survey of 877 design briefs contained in the competition database of the Austrian Chamber of Architects and Engineering Consultants61, which was divided into two distinct phases:

- a shallow analysis, in which all images contained in 653 design briefs were extracted and categorized as being site or functional diagrams.

- a thorough analysis, in which the 224 remaining design briefs were manually analyzed, taking both images contained in the actual document as well as referenced documentation into account. A caveat: Because referenced documentation was in most cases not available for download but instead listed as “additional material” (only for competition participants), I had to assume their role as diagram without actually having seen the data. To assure the reader that this approach is sane, I explicitly state that only added data for which the depiction was unambiguous (e.g. zoning regulations as map, functional program as table) was added to the analysis. In all other cases (e.g. attached wind analysis, which is either described narratively or using diagrams), I did not count that piece of data to be present.

There are several things which are noteworthy about the conducted analysis. The database contained 877 competitions, from which 117 (13%) are awards. In almost every case, the latter do not give any graphical description of the deliverables, let alone requirements in the form of diagrams. In 100 cases (11%), the competitions were EU-wide and only had the official announcement available (basically a filled-in form with no figures or attachments), which was too little to analyze. 148 (16%) of the competitions did not have a design brief at all available. Filtering out these three (non-disjunct) groups, I ended up with 559 competitions which actually contribute
data to the analysis. However, it would not be correct to regard only the “filled” data sets, as the analysis is based on the whole database being available.

Figure 32 shows the analysis of all entries of the database regarding the presence of early-stage diagrams. The given categorization is necessarily subjective, as one diagram type can fit into many categories (think, for example, of a land use map, which can also be taken as site location). As categories, I had (1.) diagram types used in site diagrams - location of the planned area (bird-eye views of the planning area, neighborhood context), static features (existing structures, photos of the planning area), legal aspects of the site, diagrams depicting dynamic forces such as wind, noise and shading. (2.) Diagram types found in functional design, i.e. the depiction of activities or processes, depiction of functions, functional relationships in a matrix, bubble diagram or zoned bubble diagram, plus presence of an architectural schema or functional and spatial program as a table.

In more than half of all cases, a spatial and functional program was given, mostly as attachment (compare 10% in the shallow analysis vs. 58% in the thorough analysis). Likewise, the site location and static features are given in the design brief. Legal constraints such as the land use plan are typically given in the form of screenshots from online Geographical Information Systems (GIS) which contain the definite legal requirements. Overall, I noticed that site diagrams seem to be highly used, albeit not in “early form” as sketches, but instead using satellite imagery with overlays or other types of digital maps as background. One exception is the depiction of dynamic forces, which seem to be lacking from almost all briefs. I can only speculate as to why this is the case: The client does probably not
have the means to conduct a study on dynamic forces, and therefore, a sub-contractor would do the required work in the form of an expert report. The few examples I found in this respect were for noise distribution and wind direction close to two railway stations, a depiction of alpine danger zones regarding avalanches and a figure depicting a fresh air corridor that required to remain intact. On the other hand, it might well be that clients think that these kinds of information are not in the main line of what he would regard as essential to include in a design brief; it is rather the architect who has to consider these factors when doing the actual design.

As for functional diagrams, the results of the analysis were quite surprising: There seems to be a lack of nearly all forms of early functional occupation in the design brief (exception: the architectural schema, which is typically drawn on top of a satellite image and thus gives a concrete form for the building in question). On the other hand - and this is quite astonishing as well, virtually every design brief for which I could obtain data had a functional and spatial program attached! So how can a functional and spatial program be elaborated without using one or the other form of design aid which I have presented earlier? For simpler programs such as housing projects, one might perhaps infer this directly. However, as we turn to complex projects - e.g. hospitals, airports, prisons, and industrial facilities, it is hard to imagine that that a client or planner acting on behalf of the client can come up with a tabular enumeration of all spaces and respective sizes, lest some form of schematic occupation has been conducted beforehand.

In summary, I attribute the lack of occupational media in design briefs to its role as a handover document (refer to Figure 33):

- First, the planning work evolves at the client side. In this context, early-stage media are used for deriving a spatial and functional program, which is a compiled view of the design requirements given in tabular form. The originating diagrams themselves, however, are left away.

- Upon entering a competition, an architect generates schematics and further “sketchy” representations in an effort to derive form from the design brief. The hand-ins for a competition show floor
plans, sections, renderings and so on, but not the intermediate results.

As a matter of fact, both phases may involve diagrams - which are themselves not deliverables but used for occupation with the planning project. This is only a theory, however. To prove that this really applies would involve a lot of work in interviewing clients and architects, which is beyond my actual scope of bringing Agent-Based Simulation into early-stage planning.
Benefits of Agent-Based Simulation for Early Planning

Agent-Based Simulation (ABS) refers to a technique in which components of a system are modeled and simulated individually, leading to an emergent behavior of the system as a whole (see Figure 34). According to Macal and North\(^6\), such individual components, called agents, have the following essential properties:

- They are self-contained and uniquely identifiable individuals.
- They act autonomously, in a self-directed manner. More specifically, agents relate their sensed information to subsequent decisions and actions, which the authors call behavior. Interestingly, the rules governing this information process can be specified by “anything from simple rules to abstract models, such as neural networks or genetic programs that relate agent inputs to outputs”.
- They have a state consisting of a set of state variables, which varies over time. The behavior of an agent may act on these variables, or on the state of the environment in which the agent is placed.
- They are social, in the sense that they have the possibility to dynamically interact with other agents and the environment.

Macal and North go on to further describe useful properties of agents such as being goal-directed, i.e. having to achieve goals with respect to behaviors. In my work, I distinguish between a “strict” type of goal-directedness, i.e. following a predefined schedule


Figure 34: Simulation of individual agents leading to emergent behavior of the modeled system. Source: own depiction in Wurzer, Kowarik, Reschreiter (Eds.), Agent-Based Modeling and Simulation in Archaeology, 2015, Figure 3.9, p. 73, reprinted with kind permission from Springer Science and Business Media.
which I call *process-based*, or “behavioristic” simulation in which agent judge their actions by internal rules and then come to a decision concerning their actions. The latter may be circumscribed also by such models as the Belief-Desire-Intention (BDI) model\textsuperscript{63} that incorporates philosophical and psychological views, or (NBD-Dirichlet) models\textsuperscript{64} that cover shopping behavior.

Regardless of what ABS truly is, whether we call it *Agent-Based Modeling* (ABM), *Individual-Based Simulation* (IBS) or *moveable Servers* (a term coined by Discrete Event Simulation [DES], which assumes that there are entities serving several requests in parallel, as given by their *capacity*) or something else, the programming paradigms underlying this discussion are clear and in no way ambiguous:

- The world in which agents are placed is either *spatially continuous* or *spatially discrete*. Continuous (left in Figure 35) means that agents move about freely, while discrete agents move in *cells* (typically square, as shown right in Figure 35, other common shapes are hex cells or cubes for the three-dimensional case). The practical difference is that a cell can behave as “agent” as well, influencing agents as they pass along (also see “Differentiation to Cellular Automata” on p. 51). An agent, likewise, can address the set of agents being “in that same cell”, or the set of neighboring cells (e.g. for quads: the Von-Neumann neighborhood consisting of the four cells to the north, south, east and west or the Moore neighborhood additionally consisting of north-west, north-east, south-east and south-west). In comparison, agents in a continuous world typically inquire other agents in a certain radius. The cell is thus a unit of integration in which values may be stored. For example, each agent may increment an attribute of each cell he visits. This attribute (let us call that *occupancy* for practical reasons) will then give the sum of agents that were located there.

- The time step with which agents move is practically always “one tick” (which is *time-discrete*). In pedestrian simulations, this often means *fixed intervals* of one second (or similar units - an hour, day, month, a season or even a year). One can also implement *arbitrary intervals* with techniques from DES\textsuperscript{65}: Each agent *schedules* its next activity by passing an *event description* (e.g. “in 17.8 seconds, call agent.move() on agent 10”) to a *scheduler*, which in


turn puts it into a Future Event List (FEL), sorted by time in ascending order. The Scheduler repeatedly gets the next event from the FEL and and sets the simulation time to the timestamp given there. It executes all activities occurring in that instance (i.e. consumes all events having the same timestamp), passing control to each agent (which can again schedule activities). The case of time-continuous ABS, where agents are executed at arbitrary instances in time is practically never used, one notable exception being the simulation of artificial stock markets.66

- Agents execute code. Depending on its intent, we can distinguish between implementations of a formal model (e.g. a solver for an individual particle in a flow field) and a “direct computational representation of systems”67. The latter can often be mapped to a finite state automaton representing different behavioral states and conditions for transitioning between these, which may refer to state variables of that same agent, the environment or another agent within the environment.

An ABM is deterministic if the same parameterisation always leads to the same simulation result. In all other cases, the model is stochastic. Programmatically, the stochasticity may come from the explicit use of randomness within the code (e.g. random direction) or from implicit factors outside the code - e.g. the order in which agents are executed (which most simulations will randomize), competition for common resources. In order to achieve significant results, stochastic models thus require repeated simulation runs, which is costly in terms of computational performance.

Differentiation to Cellular Automata. It is an often-recurring question in how far ABS is connected to Cellular Automata (CAs)68. Originally invented in the 1940s by Stanislav Ulam and his colleague John von Neumann, the approach became popular in the 1970ies with Conway’s Game of Life69. CAs are composed of a square lattice in which each cell has a finite number of states and can access their neighborhood, just as in ABMs. The simulation proceeds in discrete, fixed time steps. The true difference to ABMs is that cells cannot execute arbitrary code; they are rather limited to predefined update rules that determine the next state of a cell based on its current state and the state of its neighbors (think: pattern matching). Thus, we can say that spatially discrete and time-discrete ABMs extend CAs such that arbitrary code can be run when updating. Thus, I have argued to see cells in ABMs as (non-movable) agents, which is also the way in which they are implemented in some simulation platforms70.

---


68 S Wolfram. A New Kind of Science. Wolfram Media, Champaign, IL, 2002


Paper: Meeting Simulation Needs of Early-Stage Design Through ABS

To argue for the application of ABS within Early-Stage Planning - especially the occupational media presented earlier (see p. 20), I presented a paper together with my colleagues Nikolay Popov (of Unitec NZ) and Wolfgang E. Lorenz (Vienna UT) at eCAADe 2012. Furthermore, we held a two-day workshop in which we applied the concepts shown in the paper utilizing the Netlogo simulation platform (see Figure 36; also see “Teaching Agent-Based Simulation” on p. 175).

Authorship Information. All authors contributed equally to the paper. Nikolay Popov covered the aspects of site planning while I focused on functional planning, Wolfgang E. Lorenz targeted the aspect of ABS as design tool. The accompanying programs were written by me and Nikolay Popov in our stated fields of occupation.

Main Contributions.

- Identifies early-stage occupational media that may benefit from the application of ABS.
- Argues for the introduction of ABS as a design tool in its own right.
- Provides hands-on examples in the Netlogo simulation environment as a separate download.

Type of Work. Peer-reviewed conference paper, accompanying workshop (see p. 176).

Citation. Wurzer, G., Popov, N., and Lorenz, WE. (2012). Meeting Simulation Needs of Early-Stage Design Through Agent-Based Simulation, Proceedings of eCAADe 2012, Prague, pages 613-620
INTRODUCTION

In architecture and urbanism, problems have always been multifaceted and designers have tried to address them syncretistically. For example, each aspect of a design problem (e.g. site, circulation, client operation, costs) brings in a specific view and often wants to determine the design solution. Therefore, care has to be taken to balance and weight argumentation, both of which are hard to do when considering a problem of sufficient complexity. Agent-based simulation can contribute methods that help in this context; however, it remains underutilized in the early stages of architectural conception. To elaborate how and in which areas agent simulation can meet the decision needs during early-stage planning is therefore the main focus of this paper.

BACKGROUND

During the design process, multiple stakeholders from different disciplines have to take design decisions among multiple alternative (and likely competing) solutions. Like Rittel (1984), we are specifically interested in the argumentation behind such choices. We focus on early stages of architectural conception, which include site analysis (White 2004), functional programming and production of schemata (White 1986). In this context, Agent-based Simulation (ABS) can evaluate or generate a set of solutions according to the planning aspects being considered, by transforming them into a simplified model that uses the following concepts:

- **Agents**: The active entities within the simulated world.
- **Space**: The environment that agents act in. According to the simulation used, this may either be discrete (i.e. split into cells) or continuous, two- or three-dimensional.

Abstract. During early-stage planning, numerous design decisions are taken in an argumentative manner, based on occupation with the building site according to the different influencing aspects (e.g. topology, wind, visibility, circulation, activities etc.). In this context, sketches, diagrams and spreadsheets are the workhorses for elaboration. However, some of these phenomena are dynamic by nature, and are rather poorly modeled when utilizing static media. In our work, we thus show how agent-based simulation can be used to compute and visualize dynamic factors, in order to inform the decision process on a qualitative level. As a matter of fact, simulations may be used as a design tool in their own right, for analysis and objectified comparison among multiple design variations.

Keywords. Agent-Based Simulation; Early-Stage Planning; NetLogo; Design Process.
in order to establish abstract categories of organizations of space. Semantic models, on other hand, approach spatial configurations in terms of meaning. These meanings are of social, environmental, or programmatic quality - for example ‘commercial’, ‘residential’, ‘proximity to stream’ and ‘sunny’ or ‘shady’. Joyce, Tabak, Sharma and Williams (2010) furthermore highlight the multi-scale applicability of ABS for early stage planning, and stress that it should be regarded not only as analysis tool but also as design driver in its own right.

CONTRIBUTION OF THIS PAPER
It is obvious that a coherent taxonomy of ABS in the early stages of design is infeasible, if not impossible. Therefore, while taking into account the current corpus of knowledge as presented in the background, we follow a more pragmatic approach for evaluating potential applications of ABS:

1. We identify decision aspects (see Fig.1a) in early-stage architectural conception that are poorly met by static media and may benefit from a dynamic visualization/simulation approach (see "Simulation Needs in Early-Stage Planning").
2. We describe how ABS can be used to meet the mentioned simulation needs (Fig. 1b), based on our own work in the preparation of an architecture model suite, which we describe as we move along (see "Agent-based Simulation as a Design Tool").

Figure 1
(a) Planning aspects that influence design decisions are (b) evaluated using Agent-based Simulation among a set of proposed solutions. The results are then (c) weighted, in order to arrive at qualitative statements that are tailored to the planning task. This decision process (d) is often iterated during early-stage planning, and influences subsequent simulation runs.
3. We take a step back and discuss the presented applications of ABS in the larger context of early-stage design decision processes (see “Discussion” as well as Figs. 1c, 1d).

**SIMULATION NEEDS IN EARLY-STAGE PLANNING**

Early-stage design solutions are generated and evaluated in a multi-objective parameter space, in which each planning aspect offers a different view on the problem. Typically, such a view is presented (digitally or on paper) as a map, sketch, diagram or calculation - i.e. in a static and deterministic manner. We argue that dynamic aspects such as wind, noise, functional usage etc. are poorly captured using this approach, which regards objects (things) in space as primary and time and change as secondary. In the following section, we therefore identify some of these aspects worth simulating, highlighting these in **bold**. Our scope within the early design process is given by (White 2004, White 1986): From urban context to site planning, functional programming, schema and preliminary design.

**Dynamic aspects concerning urban context and site planning**

- **Climate** is determined by the interplay between **temperature, humidity, wind and rain** temporally and spatially. Together with solar radiation, these factors can form the basis of preliminary thoughts on sustainable building design.

- **Location** of a site within a landscape or urban environment is given as a spot on a map. However, this does not account for **travel times**, which depend (1.) on the transportation infrastructure, the circulation and the volume of traffic using it and (2.) on the functional relations at that scale (travel will take place only to a limited number of other locations).

- **Natural physical features** (e.g. elevation, soil type, bearing capacity, vegetation) are derived by Geographical Information System (GIS) data. However, certain aspects such as **drainage patterns and water levels** are dynamic, and may require simulation.

- **Man-made features** (e.g. [built/open] spaces, roads) within and adjacent to the site define **visibility** (view into, from and through the site) for both the present state as well as a potential future use (as defined by zoning and future development plan). Built form is also vital for calculating the change in **shadowing** patterns.

- **Neighborhood context** (i.e. areas directly surrounding the building site) establishes possible sources for **noise** and **pollution**. Major features (natural or man-made) located adjacent to the building site also define visibility axes that need to be preserved during Site Planning. Apart from physical aspects, analysis might also deal with the **social context** (e.g. demography, crime rate, etc.), which is also a factor that absolutely cries out for modeling and simulation.

- **Circulation** (i.e. vehicular and pedestrian movement) is dynamic in multiple senses: Apart from the obvious fact that traffic volume varies over time and space, it is possible to also model the individual participants and their **route choice** (static route choice for visiting function sequences, adaptive route choice according to behaviour model and situation - e.g. shopping, wayfinding, egress situation, etc.).

**Dynamic aspects concerning functional programming, schema and preliminary design**

- **Client operation**, i.e. usage of functions in a specified **temporal and spatial sequence** is dynamic by nature. There are two different viewpoints in this context: One may look at the activities performed by (the different types of) **building users**, or at the spaces which serve as functional containers and are being frequented by the former (i.e. **space occupancy or level of service** [Fruin 1971]). As flow between functional areas is changeable over time, so are the resulting **functional relationships** (i.e. this can be compared to the adjacency relations that
are specified in a static manner).

- **Location, dimension and orientation** of the building spaces with regard to the **sun path** are vital for ensuring natural lighting. Incident **solar radiation** can furthermore be used to consider energy efficiency at an early stage (also taking into account **shadowing**). Other aspects that apply at site level, such as **air flow**, **acoustics** and **view** (into, from and through the building) could in principle also be taken as decision aspects, however, as these factors depend already on form and choice of materials, an analysis might be more appropriate in later stages.

**AGENT-BASED SIMULATION AS A DESIGN TOOL**

In ABS, objects are viewed as secondary to the formulation of principles and processes through which things evolve and change. Simulation "seeks to formulate principles of architecture in this space of processes allowing space and time (architecture) as we know it to emerge only at a secondary level" (Testa et al. 2001). The emphasis is on understanding and exploring individual behaviours of design elements and their mutual influence on other elements. This viewpoint also encompasses recognition of hierarchies or levels (Witchesky and Resnick 1998) and understanding how complex, collective, macroscopic patterns emerge from entirely local and simple interactions of individual units.

To showcase how ABS can be used to cover the dynamic aspects presented earlier, we have developed an extensive "architecture model suite" for the freely available NetLogo ABS platform (Witchesky 1999), which we will describe in due course. The suite itself is being made freely available, with full source (see [1]).

**Setting up the cell space**

Before an ABS can be performed, the space in which the agents act must be initialized. For many of our simulations, this simply means that the cell space has to be cleared. However, some more elaborate models require the existence of specific types of spatial data:

- **Landscape Data.** Environmental information as provided by modern GIS can be imported into the cell space (see Fig. 2a). Both raster data (consisting of a grid of values) and vector data (consisting of points, lines, or polygons) are supported. Another possibility is the automatic generation of the landscape using fractals (e.g. midpoint displacement algorithm [Mandelbrot 1988]), for cases where reference to the environment is not meant literally. Generating landscapes has an additional benefit: as it is derived from fractal geometry, it reflects the overall shape in each of its parts. This property can in turn be used to see whether simulation results are similar across different scales (and if so, one can estimate the results of the model on a fine scale by running it on a coarser scale).

- **Schemata.** We offer the possibility to import preliminary sketches (see Fig. 2b) containing early-stage schemata with circulation depicted as axis-aligned lines and spaces as axis-aligned rectangles. The individual shapes are extracted in a two-pass algorithm: First, we use a connected-component labeling algorithm to extract clusters of pixels belonging to one shape. This is then fed into a feature extraction algorithm (based on the idea of the Hugh transform) that converts these pixel clusters into lines (circulation axes) and rectangles (spaces). As another possibility, we also offer an automatic schema generator (see Fig. 2c). A third possibility in the form of a schema editor (see Fig. 2d) can be used for digital sketching as well attributing imported sketches with additional information (e.g. for functions, since we are not trying to recognize handwriting from the sketches).

**Early-stage simulation of dynamic aspects**

ABS can either happen per planning aspect (i.e. many simulations running separately) or in the form
of one model covering multiple aspects at once. The risk of the latter approach lies in possible interference, which is why we have chosen to keep them separate:

- **Topography, Drainage, Water levels.** Our work in topographic simulation (see Fig. 2a) calculates surface drainage by dropping agents (“rain”) randomly onto the cell space, then letting them follow paths downwards into valleys. This example stands for a wider variety of models to take slope and elevation into account (e.g. water level computation or agents flocking along gradient, see Fig. 3a); additional entities acting as obstacles for the flow (barriers, dams) are given as interactive tools.

- **Wind, Pollution, Noise, Acoustics.** Simulation of wind flow in and around the building site, based on early-stage specification of spaces and pre-existing built environment, can act as a tool for a quick assessment of wind pressure and undesired turbulences. In our work, we use a Lattice-Boltzmann cellular automaton (see Fig. 3b) for performing necessary calculation Computational Fluid Dynamics (CFD) calculations, given the wind direction as parameter. Likewise, this model can also be adapted for the simulation of pollution propagation. The simulation of noise and acoustics is another extension of this method, although on a very basic level that does not fully take reflection, absorption, interference and other wave properties into account (lacking data in early stages).

- **Shadowing, Visibility.** Topography and site data (i.e. spaces attributed with zoning or building heights) can be used to calculate shadowing (see Fig. 3c). Agents are used as to cast rays in the sun direction, leaving shadows. The process can be iterated to show the dynamic change during a day. Visibility can also be computed by ray-casting. In principle, one might think that aspect would not be dynamic; however, there might be cases of occlusions that are time-based (e.g. tree foliage, draw bridges, docked ships) - in which this becomes useful. Generally (refer to Fig. 3d), there are two distinct measures of visibility, one “from” and one “to” a point of interest. Another method are dynamic visibility polygons (i.e. isovists), as given in (Turner et al. 2001).

- **Movement.** The simulation of pedestrians has been the predominant and most naturalistic use of ABS. For our work (see Fig. 4a), we have employed the movement model by Blue and Adler (2000), which is a lane-based model originally aimed at vehicular traffic simulations. We have extended this model for use also as a pedestrian model, by giving it 360 degrees freedom rather than being lane-aligned. The implemented model also records spatial occupancy (i.e. densities) and way-lengths per agent. For traffic that is strictly regulated (i.e. either users following fixed processes or vehicular traffic.), flow along a network-based model (e.g. as in Tabak 2008, Tabak et al. 2010) may fit better. Fig. 4b shows such a model for computing the minimal path along such a circulative network, which computes entry and exits points into-, and shortest path over the circulation (static route choice, dynamic simu-
The addition of behavioural rules, e.g. for shopping activities or egress, can be added at a higher layer, as in (Dijkstra et al. 2011). Another interesting feature is the group formation found in crowds (see Fig. 4c), as given by (Reynolds 1987).

- **Functions and Activities.** Agents as entities that perform activities in functional spaces have previously been researched e.g. in (Wurzer 2010). Our demonstration model (see Fig. 4d) selects, for every agent, a target space according to functions it intends to visit. The resulting passage along the circulative network is the same as before. Further methods of network analysis (e.g. reachability according to space syntax[2], functional relationships from agent flow [Wurzer et al. 2010]) are also included in our architectural suite, but not shown here.

- **Spread.** Agents can be used for generative processes as well, as shown in the City generation example in Fig. 5a and b: The progression of agents along a circulation being built up, leaving spaces as they go along, can be interesting from a process point of view (i.e. the generation itself is the design aspect). The sketch generator presented in Fig. 2b is also an example of such a process.

**DISCUSSION**

Evaluating a set of proposed design solutions via ABS (Fig. 1b) enables a comparison that can inform the decision process. As mentioned, simulation can either happen per planning aspect (i.e. many ABS running separately) or as one model covering multiple aspects at once. Albeit outcomes from simulation seem to be quantitative, they are not to be misunderstood as hard data: The reason for this is that they are derived from preliminary (and thus fuzzy) input, and are thus also fuzzy on the output side. What is needed is an interpretative post-step, in which the obtained values are given meaning.
For example, way lengths computed are in abstract ‘cell space units’, they can be compared relatively, however, a better way would be to map them to a scale that is applicable (e.g. building scale) and then think about implications for the simulated group of persons (e.g. too long, long, medium, short). As the evaluated planning aspects are not equally important, they must be weighted (Fig. 1c) before the decision process takes place. This weighting lies outside the simulation, as it represents the discussion during decision-taking (Fig. 1d).

Among the modeling and simulation community, an often-discussed topic is also that of scale of the simulation model (i.e. microscopic, mesoscopic and macroscopic), which should ideally only be microscopic. Our choice is a more pragmatic one: As long the model is agent-based (and agents are: the animated parts of the world, each cell, all nodes and edges of a circulative network), we are happy to consider it. We are well aware, however, that in strict terms some models (e.g. the Lattice-Boltzmann CFD) are at least mesoscopic. A similar argumentation is also heard for model scope: An agent-based model should constrain itself to the minimal implementation needed to describe an effect. However, as we consider ABS as design tool, some additional steps such as the data import from sketches are necessary, in order to be able to work within the design workflow in place. Improved support for the import of sketches (e.g. more shapes in the schema, according to [Achten, Bax and Oxman 1996]) are definitively a future work item in this respect.

SUMMARY

We have argued for the introduction of Agent-Based Simulation (ABS) into the early-stage planning process, in order to be able to capture and evaluate design aspects that are inherently dynamic (e.g. wind, pedestrian flow, functional usage). One of the main advantages of this approach is that it makes it possible to experiment on simulations of the real thing, not on the real thing itself. Such computer models are essential planning instruments that make it possible to ask multiple ‘what if?’ questions about the system of interest. Our work and contribution in this respect is that we are transferring simulation concepts into early phases of architectural design, so that these questions can be asked earlier and thus with more influence on the rest of the design process.

REFERENCES


Figure 5
Modeling Spread. (a,b) Generative urban modeling, (c) spread of functional spaces.


Bonabeau, E 1997, ‘From Classical Models of Morphogenesis to Agent-Based Models of Pattern Formation’, Artificial Life, 3(3), pp. 191-211.


Fruin, JJ 1971, Pedestrian planning and design, Metropolitan Association of Urban Designers and Environmental Planners, New York, USA.


White, ET 1986, Space adjacency analysis: Diagramming information for architectural design, Architectural Media, Tallahassee, USA.

White, ET 2004, Site analysis: Diagramming information for architectural design, Architectural Media, Tallahassee, USA.


[1] www.iemar.tuwien.ac.at/processviz/early-stage-sim
Data Sources for Early-Stage Agent-Based Simulation

The application of ABS within early-stage planning is limited by the data available at that stage. It is therefore vital to establish a sound basis for simulation in this step, before coming to applications and advanced concepts.
Paper: Pre-Tender Hospital Simulation using Naive Diagrams as Models

If we want to take clients seriously, we should base “early” simulations on whatever occupational media they have produced in the pre-tender phase. Luckily, we were invited by Manfred Pferzinger (UMIT, FH Krems and MediPro) to join in on a project for the Landesklinikum Wiener Neustadt (a clinic in Lower Austria having 868 beds) where we could work with earliest-phase occupational media. More specifically, our task was to simulate patient flow for a rebuilt future version of the clinic (see Figure 37), based on a preliminary Schema produced in close cooperation with the clinic’s staff (a process which took Manfred Pferzinger one year[7] and was based on a huge amount of interviews and workshops).

From all the work done for this habilitation, this was the project with the “earliest” application of ABS. However, as we found out, that does not mean that there is a lack of information: Every clinic nowadays records, for every hospital case, the sequence of functional units that a patient has visited. The primary system where this information is stored is the Hospital Information System (HIS), however, there are other (connected or stand-alone) systems such as the Operation Room Management System that store more detailed records of the patient’s visit. We were given anonymized versions of such data in the form of “treatment chains”, i.e. sequences of functional units for each patient visit. The schema was given in Microsoft Visio, which is also where we wrote our simulation (“VisioSim”, an ABS in Visual Basic for Applications running directly in

---

Microsoft Visio also because of the client’s wish for the possibility to print the results at arbitrary resolutions. The simulation uses the nesting of the drawn functional units to build up a hierarchy and furthermore infer semantical information (e.g. function shared between two zones). It animates the flow of patients between functional units, utilizing them as capacity-constrained resource. The flow and utilization are depicted in the form of a bubble diagram and resource monitors (refer again to Figure 37) which can be saved directly in the schema. As a testcase, we simulated resource utilization for the operative area\textsuperscript{72}. In summary, these tests showed the practical benefits of having a simple simulation model that is manageable by clients or planners acting on their behalf, suited also for presentation before stakeholders.

**Authorship Information.** A large part of the work was done by Wolfgang E. Lorenz and myself\textsuperscript{73}, Manfred Pferzinger additionally contributed inputs on the schema he had produced on behalf of the client. The simulation was written by us Wolfgang E. Lorenz and me.

**Main Contributions.**

- One of the earliest applications of ABS in hospital planning, to the best of our knowledge.

- Reverse-engineers a client-generated schema into a simulation environment, by identifying a set of rules underlying the depiction (see especially Figure 3 of the paper).

- Imports data from a Hospital Information System (HIS) to generate patient arrivals and real treatment chains (see Discussion and Results on p. 160 of the paper), which is unprecedented to the best of our knowledge.

- Simulates the individual process of each patient, acting on capacity-constrained functional units (see especially Figure 5 of the paper).

- Infers a bubble diagram visualizing patient flow and function utilization (see Figure 6 of the paper; see also Figure 37).

**Type of Work.** Peer-reviewed conference paper and accompanying flow simulation software. Contribution to LK Wiener Neustadt (see p. 187) and STABLE AIR projects (p. 187).


\textsuperscript{72} a more elaborate simulation (e.g. for the whole hospital) would have gone well beyond the scope of our work, since we did not the available resources to assess the capacities of every unit

\textsuperscript{73} I estimate that I have done around 50\%
PRE-TENDER HOSPITAL SIMULATION USING NAIVE DIAGRAMS AS MODELS

Gabriel Wurzer(a), Wolfgang E. Lorenz(a), Manfred Pferzinger(b)

(a) Vienna University of Technology
(b) UMIT, Hall in Tirol

{gabriel.wurzer|wolfgang.lorenz}@tuwien.ac.at, manfred.pferzinger@umit.at

ABSTRACT
Hospital simulation has so-far concentrated on late phases of architectural planning, in which the design is already fully formulated and undergoes optimization. This paper moves into the exactly opposite direction - it tries to embed simulation into the earliest phase imaginable, which, interestingly, is well before any architectural planning occurs: The pre-tender work that is done by or on behalf of the client, using naïve diagrams based on interviews with the medical staff as models.

Keywords: early design process, hospital planning, diagrammatic tools, pre-tender simulation

1. INTRODUCTION
The preparation of a tender represents the earliest work done in hospital planning: Projected medical demands of a care region are mapped to either the establishment of a new hospital, or, more often, to an extension, adaptation or refurbishment of an existing one. Classically, the tool of choice for 'simulating' and 'calculating with' future demands has been spreadsheet software. Apart from statistical data, diagrammatic information (e.g. envisioned work processes and spatial arrangement of departments, obtained by interviewing the medical staff) is also generated. However, and in contrast to spreadsheets, there is a great lack of methods for interacting with the so-encoded data - especially when it comes to animation/simulation of (patient and material) flow.

Contribution: During the past half year, we have thus been working on reverse-engineering diagrammatic representations into simulation models, dealing with needs-orientated flow descriptions (patient flow), preliminary floor plans (schemata) and functional decompositions (bubble diagrams), in order to obtain a simulation targeted at the pre-tendering phase. To discuss the techniques used in this context and give a wider perspective on other possible applications is the main topic of this paper (see Section 4, “Contribution Details”).

Establishing simulation in the earliest phase of planning enables clients to get a better overview of the project they are writing a tender for. Furthermore, diagrammatic data produced as result of the subsequent competition phase can thus be evaluated and compared, which is advantageous for a wide audience:

• Planners. The spatially simulated flows can inform the design and enable a cross-check of requirements for the planned building. Generally, the flow given by the client represents a preliminary concept (“schema”), in which the spaces are not fully formulated. However, the processes depicted therein stay essentially the same, even after the architect has designed the final form (“floor plan”). By adaptation of the schema into the final floor plan, the architect can simulate the flow in his presented concept, while still relating to the client’s vision.

• Staff. The hospital staff can be trained using the very same simulation, in which context also an acceptance check can take place. As the initial flow concept is typically generated by the same persons that will see the simulated final design, this enables to re-use the previous knowledge for evaluation.

• Client. Verification of requirements, as mentioned, can help in the decision process during the competition phase. Moreover, as the hospital goes into operation, realtime data concerning the actual patient flows can be exported from the Enterprise Resource Planning (ERP) system, in order to input and visualize these in the sense of a “management dashboard”. To some extent, this also facilitates a verification of the building’s operational concept.

2. RELATED WORK
Diagrams (White 1986; Seonwook and Miyoung 2012) are used as representations for the envisioned organization, i.e. space allocation (zones and their adjacency relations, circulation), hierarchy and responsibilities (organizational chart), temporal or causal order (processes, flow). The sources for this information vary; however, one common point is that planning is never done in isolation, but by drawing on pre-existing knowledge of several health professionals, such as physicians, nurses and therapists, administrative personnel and (most importantly) the patients...
themselves. Tool support for employing diagrams as interview technique, used to let workshop participants record daily work routines in a game-like manner, was presented by (Wurzer, Fioravanti, Loffreda and Trento 2010). Simulation in the same setting, as “assessment tool”, has so far not been conducted. What exists in early-stage spatial simulation are approaches exploring space utilization (Tabak 2008; De Vries, Jessurun and Dijkstra 2002), verification of a functional program (Wurzer 2010), usage simulation and pathway visualization (Wurzer 2011) and early process simulation (Wurzer 2012).

3. BACKGROUND

As stated earlier, work that leads to the definition of early diagrams (most importantly: the architectural schema) is conducted collaboratively between health professionals and patients. The overall goal is to define requirements that enable effective workflows according to customer needs and expectancies. Because objectivity during such an elaboration process is of vital importance, work typically proceeds in workshops targeted at a specific problem domain.

Outputs are at first centered on process knowledge - i.e. daily work routines, clinical procedures and practices, such that one might produce business processes for outpatient, inpatient and day hospital treatment, according to the different areas of the hospital as defined in (DIN 13080 2003) as result. One may also specify procedures in case of special situations - e.g. mass accidents, which is especially important for trauma clinics. With these processes in mind, one may look at the intended spatial configuration, which is the topic we are focusing on in our work. In more detail, we employ three specific types of diagrams produced within pre-planning:

1. The architectural schema (see Figure 1a) as a hierarchy of “spaces within spaces”. A space is a bounded (usually rectangular) region that is denoted by a name. It contains a set of functions, i.e. names of activities that may be used in that space. Each function is a resource of limited capacity, of which the usage is computed during the subsequent simulation (see Simulation). It is crucial to note that the schema is fundamentally different from a floor plan showing the form of each space: It rather gives the approximate area and location of each space in two-dimensional arrangement, which is then detailed into the fully-formulated three-dimensional building layout during the competition.

2. Flows (see Figure 1b) depicted as arrows on top of the schema, giving a simple yet effective way to express paths of building users (Lohfert 2005). Usually, these are color-coded to distinguish different kinds of traffic (patients, staff, visitors).

3. Bubble diagrams depicting the adjacency relations between functions in a purely abstract manner; a function is visualized as circle (“bubble”), the adjacency relation to another function as connecting line. Adjacency can either be “close” (denoted e.g. by a green line) or “separate” (denoted e.g. by a red line), “not given” or “not applicable” (no line or gray line). Additionally, it is common to depict this relation also by arranging close functions into clusters.

Figure 1: (a) Architectonical schema as arrangement of nested spaces with functions. (b) Superimposed flows.

Figure 2: Bubbles diagram showing bubbles (representing functions). The size of a bubble corresponds to its assumed area consumption, the relationships between functions are given as colour-coded lines.

When using diagrams as source of information, one can exploit hidden semantics given by the topology of shapes present, in order to infer semantic relationships. More specifically (refer to Figure 3):

- Nesting of spaces (Figure 3a) can be inferred from topological inclusion.
- Conceptual containment (Figure 3b), in contrast to ‘real’ nesting, hints at a grouping of spaces using an abstract boundary, signified by a dashed line.
- Using and being used by (Figure 3c) according to the z-order - the lower element is using the higher one, transitively.
4. CONTRIBUTION DETAILS

Our work first derives a spatial model from a schema (see “Rectangle analysis”, 4.1). The result is a semantically rich model of a hierarchy of spaces, together with the functions they contain. This is then used in the second phase, where patient flow is computed using an Agent-Based Simulation (ABM) that computes the progression of patients through the spaces, utilizing a sequence of functions as resources. Arrivals of patients are given as spreadsheets, based on either (hypothetical) arrival times and functional sequences, or, more commonly, using real data obtained through either the Hospital Information System (HIS) or the underlying Enterprise Resource Planning (ERP) System. As functions are the resources within the simulation, their usage is recorded and visualized in the third phase as Bubble diagram: For each function, a bubble with circle size according to capacity is drawn. The circle is colored according to functional utilization compared with capacity (under-utilized, over-utilized or well-utilized). Relationship lines between function are also depicted, based on the simulated flow between functions. Conceptually, this work is similar to e.g. (De Vries, Jessurun and Dijkstra 2002) - being a design and decision support tool based on employing simulation in the architectural workflow. Our focus on earliest stages (i.e. pre-tender phase) is, however, unprecedented in hospital planning, requiring careful thought over what data already exists, as will be shown in the following elaborations.

4.1. Rectangle analysis

Schema diagrams are drawn intuitively, as rectangles-within-rectangles and possible overlaps. In this section, a quick run-through of the analyzed features is made (also refer to Fig. 4):

- **Hierarchy buildup.** The pair-wise analysis of topological relationship between each two rectangles gives either “separated”, “completely included in/completely including”, “intersecting” or “touching”. In the first (trivial) case, the rectangles are completely separate and thus modeled as own entities. In the second case “included in/including”, the rectangles form a parent-child relationship. When “intersecting”, the rectangles also form a parent-child relationship; however, it is still unclear which is the parent and which the child rectangle. Z-order of the rectangles can be used as tie-breaking mechanism - the child element being the one that is arranged ‘on top’. Intersections are commonly used spaces that are used by more than one parent - e.g. a central operation theatre being used by many connecting areas. Accordingly, the hierarchy we build up offers, for each rectangle, the ability to have more than one parent - thus forming a semi-lattice rather than a tree structure (Alexander 1965). The last case, “touching” rectangles, is ignored - we assume each of the both rectangles are children of a common parent structure, and the touching relation being there to depict adjacency, not hierarchy.

![Figure 4](image-url)
• **Space and function attribution.** Up to this point, it is not clear whether the analyzed rectangles stand for spaces or functions, since both are nested rectangular elements. By use of manual attribution, we can infer that a given rectangle represents the latter structure - in which case it must also contain a capacity (integer). Additionally, in order to facilitate the later use of the function as resource, we also allocate a list currently occupying and queued agents.

4.2. Flow Simulation
Flow is given by arrivals in spreadsheet format, where each line represents one single patient (see Figure 4). Also on this line are 1.) the arrival time and 2.) the sequence of functions to visit, including stay time per function. Then, the simulation scheduler progresses in discrete time steps, in which:

1. **Arrivals** for the current instance are instantiated, producing agents with a fixed protocol of functions to visit. The first of these functions is immediately removed from the list, and the agent location is set to the space containing the function.

2. **Movement** of agents is simulated for all agents that are crossing to the next function. This happens with reference to the underlying schema, which is interpreted as circulative network.

3. **Occupation** of functions is simulated by employing the active and queue lists of the function under consideration; the total time spent per function is available in the agent itself, reflecting the amount of time taken for individual treatment. Simulation constructs such as passivation/activation can happen in this step, or, trivially, the agent waits for that time-span.

4. **Goal selection** happens for those agents that have finished using their function, by removing again the first element of the list of functions still to be visited. In case there is no such function, the agent is removed from the simulation.

The simulation runs as long as there are elements in the arrival list or there are active agents on the schema. In case the simulation ends, the visualization is prepared.

4.3. Visualization
In architectural workflow, Bubble diagrams are used to depict relations among functions; however, these are only intended relationships (i.e. intended by the architect, from close collaboration to dislocated). In our simulation, the results are used to build up such a diagram, based on the actual usage of functions and flows between them (also refer to Fig. 5a). For each function (depicted as a circle), we depict the usage as its radius, the interpretation of this usage compared with the capacity as color (under-utilized=blue, over-utilized=red, well-utilized=green, not enough data=gray). The flows between functions hint at their level of cooperation. By thresholding, we can get a measure of closely cooperating function pairs, depicted as lines. However, in contrast to Bubble diagrams that have been specified manually, we cannot state which functions should not cooperate at all, and should thus be dislocated (e.g. for means of privacy, hygienic aspects, etc.) - our approach is inherently positivistic in this aspect. The resulting Bubble diagram can nevertheless be compared to a manually-made one, as validation.

As visualization, we can record the usage of functions over time (see Figure 5).

![Bubble Diagram](image.png)

**Figure 5:** (a) Visualization of functional relationships obtained by using the flow simulation. Circles depict functions, links between them cooperation in the (simulated) work process. The size of each circle corresponds to the usage of the given function, the color is a comparison between usage and capacity (green: well utilized, red: over-utilized, blue: under-utilized, gray: too little data). (b) Depiction of usage of each function (red line indicates the capacity having been set by manual attribution).

5. DISCUSSION AND RESULTS
The stated approach was implemented and tested with anonymized patient trails (inpatients and outpatients, 1900 individuals) exported from a hospital’s ERP system (SAP) and OT management system (60 individuals) for the course of one day which we see as being quite average - a Wednesday in non-vacation time. Technically, we received a sequence of time-stamps together with the service point (e.g. 10:05 Radiology, 11:39 Pediatrics) for each anonymized patient. From this, we could build a trail “Radiology > Pediatrics” and inferred durations from the time delta. The resulting duration is, arguably, not correct - but it is not wrong either:

- **Shortcoming 1.** The given timestamps are recorded either at the start or at the end of the service duration, which is bad. But, even worse than that, we can safely assume that both cases are present in the same dataset (e.g. OT times measured at start, radiology at the end of the service duration). Thus, the data is necessarily fuzzy, and must not be interpreted...
quantitatively but rather qualitatively and with a grain of salt.

- **Shortcoming 2.** We cannot compute durations from a single time stamp (or, as in the example mentioned earlier, we do not know how long the patient was in the Pediatrics unit). To counter this problem, average service times were used where available.

- **Shortcoming 3.** The duration computed has not necessarily got to do anything with the real service time. For example, the time delta between 11:39 Pediatrics and 10:05 Radiology would be 1 hour 34 minutes, which is a rather long time for, say, an x-ray. Again using average service times (e.g. x-ray: 4 minutes), one can dispatch the patient to the next function and use the remaining time as waiting time (i.e. Radiology(4m)>Pediatrics(1h 30m)).

In contrast to the mentioned problems, we also saw a large benefit: We were able to transfer knowledge about a hospital to a new design, using arrivals and current trails as input for a new design. For this to be possible, a transformation and mapping step was incorporated into the interpretation of the timestamps: Each current service point name was replaced by the name of the future planned function (e.g. ‘Pediatrics’ becomes ‘Pediatric Centre’). In the same instance, we also distinguished trails by the type of patient (inpatient/outpatient) and functions present in the trail, in order to colour-code them (see Figure 6a). In this fashion, an outpatient with the trail Radiology(4m)>Pediatric Centre(1h 30m) would be tagged as child. By the assumed type of person, we also choose an entrance space to complete the arrival, i.e. arrival at 10:05: Main Entrance(0m) > Radiology(4m) > Pediatric Centre(1h 30m) > Main Entrance(0m). The passage times between functions are disregarded.

6. IMPLEMENTATION DETAILS

Our implementation uses Microsoft Visio for graph drawing (see Figure 6a). Under the hood, we are employing the bundled Visual Basic for Applications scripting language as means for enabling simulation, i.e. topology analysis and animation/simulation. The former point deserves some more detail: Visio is an excellent tool for doing topological analysis; it comes with a built-in support for finding the spatial relation between pairs of shapes, resulting in either containing, contained in, overlapping, touching or none. Likewise, the support for animation and automated diagram drawing (which we needed for the bubbles) is excellent, and lead to a total deployment time of less than a month. Also, the possibility for attribution of shapes via shape data (see Figure 6b) proved a valuable tool with which the users were already acquainted. Likewise, the ability to define custom shapes made it possible to put usage monitors (see Figure 6c) into a palette, with users being able to drag them onto the drawing sheet and anchor them to the function to be monitored.

Deployment of the simulation is also fairly easy - the user needs to open a diagram containing the scripts in parallel to the diagram to be simulated. By this, we are able to reach a wide audience that is not tech-savvy, or even (one might say) reluctant to install additional simulation software. For the end-user, the simulation is invoked via an additional menu of Microsoft Visio, which brings up the user interface (refer to Figure 7):

- In the main screen (Figure 7a), the user selects an arrival spreadsheet and can then start the rectangle analysis and simulation.
- During the simulation, the user can scale bubbles and compute usage (bubbles and throughput (relations).

![Figure 6: Implementation in Microsoft Visio. (a) Graph editor used for inputting schema and, subsequently, for depicting the flow of agents across a set of functions, which is visualized as bubble-diagram. (b) Manual attribution of shapes and setting of capacity for functions. (c) Depiction of function usage over time.

![Figure 7: Screens used for the simulation. (a) Main screen with choice of spreadsheet and buttons to start rectangle analysis, reset simulation, run simulation, scale bubbles and compute usage. (b) Scale bubbles screen, giving the choice to scale usage (bubbles) and throughput (relations).](image-url)
which the user can override by pressing a “compute usage” button.

CONCLUSIONS
We have presented a novel approach that helps clients simulate a very preliminary schematic diagram, targeted at the pre-tender phases of a hospital project, where requirements have to be elaborated in close collaboration between medical staff and patients. The output of the approach is suitable as input for the competition, since it speaks the language architects understand (i.e. bubble diagrams, functions and spaces).

REFERENCES

AUTHORS BIOGRAPHY
Gabriel Wurzer earned his Ph. D. degree in Process Visualization and Simulation for Hospital Planning from Vienna University of Technology in 2011. His research in architectural sciences focuses on tool support for early-stage planning of complex buildings, with regular contributions to both Pedestrian and Evacuation Dynamics conference (PED) and the Education and Research in Computer Aided Architectural Design in Europe conference (eCAADE), from which he was awarded the Ivan Petrovic Prize in 2009. He is also an active researcher in archaeological simulation, together with the Natural History Museum Vienna.

Wolfgang Lorenz has a degree in Architecture from Vienna University of Technology and is currently working on his doctoral thesis on Fractal Geometry and Quality in Architecture, in which he investigates the concept of applying fractal geometry to architecture. Apart from his research work, he also gives lectures on “programming for architects” at the department of computer aided planning and architecture.

Manfred Pferzinger got his Ph. D. degree from the Private University for Health Sciences (UMIT) in 2012, focusing his research on public health and health care quality, in which subject he is author of numerous articles and books. Apart from being associated scientist at UMIT, he is lecturer for the University of Applied Sciences in Upper Austria (Steyr, Austria) and the University of Applied Sciences in Tyrol. He was founding member of the Austrian Competence Circle for Clinical Pathways (A3CP) and currently works as managing director of medipro consulting, a company that focuses on health care efficiency.
Many complex buildings are process-based, meaning that the design is driven by pre-existing operational routines that are highly formalized, for example as flow charts. These can easily be simulated by means of Business Process Simulation (BPS), even at very early stages of planning. However, one thing generally lacking in BPS is the ability to simulate activities which depend on the building layout (think: schema). In order to enable such a functionality, I have been working on a hybrid simulation in which a BPS can communicate with one or more ABS which simulate such behavior on its behalf. An illustrative example for this would be taking a lift: The time taken by each agent (i.e. process instance) does not only depend on the building layout, but also the state of the environment and cannot be specified beforehand. In summary, the static process simulation can thus be augmented by dynamic simulation within the schema.

Authorship Information. I am the sole author of the article and the accompanying two programs - a mock BPS on the sending side and an ABM on the receiving side, executing a wayfinding algorithm.

Main Contributions.

• Identifies pre-existing processes of an organization as possible starting point for early-stage simulation.

• Enhances BPS where needed to include dynamic behavior within the building layout. Substantiates this claim using an implementation (see Figure 3).

• Makes the relationship between process planning and the functional design explicit (see p. 333 of the paper), in order to inform the mathematical simulation community which was the audience of this article.

Type of Work. Peer-reviewed article with accompanying implementation (BPS and ABM).

Citation. Wurzer, G. In-Process Agent Simulation for Early Stages of Hospital Planning, Mathematical and Computer Modelling of Dynamical Systems, 19(4): 331-343, 2013
In-process agent simulation for early stages of hospital planning

Gabriel Wurzer*

Digital Architecture and Planning, Vienna University of Technology, Vienna, Austria

(Received 16 June 2012; final version received 19 December 2012)

In the early stages of hospital planning, work processes are typically modelled in a static manner by using flow charts or business process modelling notation as means. Diagrams of this kind are easily simulated; however, employed process engines lack possibilities for dealing with dynamic aspects of the process which depend on the building layout (e.g. elevators, behaviour of automatic delivery carts). If one could give planners the opportunity to employ dynamic entities without having to change their usual workflow, one of the benefits would be that they are not being forced to resort to naive assumptions (e.g. 15 seconds per floor) that are still commonplace in today’s planning practice. As another contribution, we show how agent-based simulation can be used as an analysis tool by using the state of the process simulation to generate arrivals.

Keywords: hospital planning; functional design; process simulation; agent-based simulation; hybrid

1. Introduction

Hospitals, like airports and some types of industrial facilities (e.g. oil platforms), are process-driven buildings: Their design depends foremost on the planned work processes that enable them to operate day and night, 365 days a year. Therefore, the process model of such a building constrains the architectural design, which must evolve in close cooperation between process planners and architects.

Because processes are modelled in a highly formalized manner (e.g. as flow charts), one might think that the application of simulation lies at hand from the very start of a building project. However, such static process descriptions lack the ability to also include aspects that depend on the building layout, such as the transition of persons and material from one space to the other, possibly using dynamic entities such as lifts as they move along. Resorting to naive assumptions (e.g. fixed passage times) might be inadequate (again taking the lift as example) and, furthermore, cumbersome to elaborate. In early planning, there are usually several variants of the spatial concept rather than only one for later phases.

Our work therefore focuses on overcoming the mentioned problems by embedding dynamic entities into an otherwise static process model. Broken down into further detail, our contribution consists of:

- A thorough look at ‘simulation needs’ in the early stages of process-driven building design (see Section 3). Such a survey is surprisingly novel, as the community has

*Email: gabriel.wurzer@tuwien.ac.at

© 2013 Taylor & Francis
previously targeted hospital simulation problems but not their context within the planning process.

- An extension of static process simulation such that dynamic entities (acting in a spatial context) can be represented. Technically, this is achieved by invoking an agent-based simulation on behalf of the process simulation (see Section 4).

Our work here is purely foundational: We want to augment the now-common working style of planners in a non-intrusive manner, that is, extending rather than reinventing design tools available. The choice of an agent-based simulation on top of a process simulation fits exactly this line of reasoning. As a matter of fact, we do not focus on results, but on the concepts needed to obtain them. If process modelling tools were extended in the spirit proposed herein, it is likely that a paradigm shifts away from separated planning of organizational processes and functional/spatial design towards a more integrated planning practice occurs. Apart from reducing the communication overhead, this also translates to faster turnover times. Especially when it comes to architecture competitions, this can be a very significant factor.

2. Related work

Business process simulation (BPS) is based on linking a graph-based model to a discrete event simulation (DES) that simulates its behaviour over time. There exists a variety of software packages implementing BPS (e.g. [1,2]) plus some DES packages that provide a ‘flow chart’ like approach by means of a server/client-based model [3–6].

When it comes to simulation that requires an understanding of the spatial concept (as in the previously mentioned examples of elevators and automated delivery carts), we see that most of the DESs seem to focus on late design phases, that is, phases in which the spatial concept is already fixed and subject to optimization (e.g. via [7]). Especially in hospital planning, this might be a problem, since spatial design is subordinate to the process model, and might thus not be evolved as far as the latter when a simulation is performed. We have, therefore, previously presented a coupled pedestrian/process simulation targeted at early design stages, in which the early concept (also called schema) is taken into account [8].

Our efforts for this paper are approaching the problem from a different side: Our goal is to enable planners using a static modelling approach (i.e. flow charts) to include dynamic entities into their process descriptions, based on a BPS being linked to an agent-based simulation (ABS). We are aware of many approaches being occupied with this specific hybrid mode [9,10]. However, none is focused on the planning context of early-stage hospital design, which is essential when producing an approach that is adapted to working routines now in place.

3. Background

The simulation needs for early-phase hospital design are closely connected to the design process. In the following subsections, we will describe the typical planning tracks and deliverables in early design, before coming to the actual problem areas in which simulation can provide valuable input when being used as a design tool. Because of space constraints, we have omitted a discussion on the influence of different design methods used, and may forward readers interested in this topics to [11,12].
3.1. The early design process

There are two design tracks that are important to early-phase hospital design, Building Organization and Functional Planning. The brief overview of these tracks that follows is important, since they form the context of work routines within which an ‘early-stage’ approach needs to operate.

3.1.1. Building organization

Building Organization (see upper part in Figure 1) is occupied with the planning of the organization from the side of business administration, that is, definition of the organizational structures (departments, sub-departments), work processes and responsibilities within these. In essence, the planning work proceeds top-down: starting with a very coarse outline of business activities required for operation (Figure 1(a)), a basic formulation of processes can be derived by introducing temporal and causal order (Figure 1(b)). The notation of these processes depends on preferences of the project team, two usual options are flow charts or process graphs according to the recently standardized business process modelling notation (BPMN).

As the project progresses, some activities might need to be further detailed in order to be fully defined. This can be done by using sub-processes, which establish a hierarchy of activities within activities (Figure 1(c)). Furthermore, when detailing a process, responsibilities for each activity are also assigned to different collaborating departments within the organization (see Figure 1(d)). The finished product and goal of the building organization is thus a description of the whole operation of the building from a business side (also called ‘process model’ of the organization). The process model acts as input and constraint for the functional planning track.

![Figure 1. Early-stage planning tracks. (upper part) Building Organization: (a) activities formed into (b) processes and (c) sub-processes. Furthermore, assignment of process responsibilities to different departments leads to (d) process model, which acts as input and constraint for (lower part) Functional Planning: (e) Functions are (f) related via an adjacency matrix, (g) grouped to form (h) spaces within the architectural schema. Circulation is additionally inscribed using arrows.](image_url)
3.1.2. Functional planning

Functional Planning (see lower part of Figure 1) starts with a definition of building functions (i.e. capabilities of a building, see Figure 1(e)), based on the intended vision (laid out e.g. in the tender document, project description, etc.) and process model of the organization. These functions are then correlated in an adjacency matrix [13] by the degree of collaboration, ranging from ‘adjacent’ for closely collaborating areas to ‘dislocated’ for areas that do not cooperate or must be separated, for example, because of hygienic considerations (Figure 1(f)). Adjacent functions are then grouped into spaces (refer to Figure 1(g)): in the example given, ‘operation theatre’ and ‘recovery’ are put into one common space (signified by a dashed border), while ‘trauma’ stays isolated and gets its own space. The so-found spaces are then arranged in a preliminary floor plan called ‘schema’ (see Figure 1(h)), with each space being represented by a rectangle.

In this context, the rectangular form of every space is not to be taken literally, since it merely gives the proportion, approximate size and location in relation to other spaces. The concrete form for each space is beyond the work done in early-stage design – it occurs later, in a phase called ‘Form Finding’. Apart from the spaces, the schema also contains arrows that give the preliminary circulation system (e.g. corridors) of the building. Aside from the graphical notation, the schema is typically also given as spreadsheet form (‘Space Allocation Plan’), which listing spaces (often grouped by function), cardinality (e.g. 2x) and usable area per space (e.g. 25 m²), commonly regulated by guidelines and planning handbooks such as [14]. Depending on project structure, the Space Allocation Plan may be produced during Functional Planning or be given before the actual planning work starts, as an input (e.g. when extending a building).

3.1.3. Co-evolution through design decisions

It is noteworthy that the activities of Building Organization and Functional Planning are not sequential but inherently parallel: As process model and the spatial concept are detailed and evolve side-by-side, the planning team has to ensure consistency of both models. Furthermore, the spatial concept might fork of a variety of alternative designs, which must then later be reduced or merged by design decisions (i.e. documented argumentation within the planning team leading to a set of choices, see [15,16]). For process-based buildings in the early planning stages, these design decisions are typically based on:

- **Urban context.** It is the relationship between the planned building as a whole (i.e. arrangement of spaces and circulation) with its surrounding environment and the existing infrastructure [17]. For example, traffic patterns resulting from local public transport and motorized individual traffic have to be taken as a constraint. Visibility of landmarks has to be preserved by (and likely used in) the proposed design.
- **Adjacency.** Short paths between collaborating units (defined by the adjacency matrix), considering the adjacency matrix [13], process model and expected volume of building users requiring service. Vice versa, a separation of spaces for reasons of privacy (e.g. secure areas vs. public spaces, inpatient vs. outpatient areas) and for sustaining building operation (typically by service corridors, allowing for repairs and delivery ‘behind the scenes’).
- **Separation of traffic.** Different routing according to type of traffic [18], for example, separation of staff from visitors and patients, low-priority from high-priority traffic (e.g. emergencies), building users with appointment from the ones without, soiled from clean material, and so on.
• **Location, size and proportion.** Placement of spaces is not isolated from considerations of the building as a whole; for example, certain areas of work favour natural lighting (e.g. patient rooms, energy considerations for the whole building), while others can do without. Proportion and size of individual spaces determine the opportunity of future adaptations (e.g. change in equipment), while at the same time being subject to optimization (minimal area needed per function, compactness).

• **Orientation and way-finding.** Depending on intended user spectrum, orientation can play a vital role for the whole building project. The transition of building users from one space to the other must be considered both in terms of the process as well as existing previous knowledge about the building layout. Spaces serving processes used by temporary building users must be easy to reach (i.e. no signage required) and memorize (e.g. using a main corridor connecting all departments). A clear readability of space can also help in fire safety and evacuation planning, conducted in later phases [19].

• **Extensibility and adaptability.** Both extensibility and accessibility of a building is given by the configuration of spaces and circulation [14]. The first one deals with openness to the outer environment, and the second with (usually multi-functional) hub spaces that serve as distribution points for pedestrian traffic, often located at prominent positions within the building. The ability to adapt the spatial concept to future requirements of the process also requires an evaluation from a multi-functional view (e.g. interdisciplinary use of a space, shared workspaces, etc.).

• **Adequacy of planned concept.** The adequacy of both spatial concept and process model is an overall judgement of the building’s design under consideration of the planning task. Argumentation focuses on whether the design satisfies the vision and financial context stated by the client. In the planning team, the discussion is centred on the types of functions present and sizes of their respective spaces as well as structure of the processes and needed resources.

### 3.2. Early-stage simulation needs for hospitals

Given the mentioned design decisions in early planning phases of process-based buildings, simulation can contribute tools for assessing a variety of aspects which can then be weighted according to the planning objectives (i.e. multi-objective analysis). Statements produced in this manner are necessarily qualitative, since spatial concept and process model are in a preliminary stage.

#### 3.2.1. Visibility, accessibility and way-finding

The analysis of these parameters may be done statically (for the whole building, its arrangement of spaces and circulation network) or dynamically (by simulating individual processes). In the first case, reachability analysis of the circulation network can be conducted for both interior and exterior spaces by using the methods provided by Space Syntax [20,21], which can also compute the visibility from each point in the building (e.g. for hiding areas for supply and disposal). View shed computations, usually found in geographical information systems (GIS), can be used for the same purpose. Way-finding, on the other hand, requires a dynamic simulation of individuals following their processes (e.g. using ABM). In this connection, algorithms from pedestrian dynamics may be used to simulate the physical movement under the influence of congestion (e.g. via [22]).
3.2.2. Space placement and dimensioning

Previously defined adjacency relations can be verified by simulating the planned processes by means of BPS, ABS or system dynamics (SD). The volumes of traffic between the spaces, distances travelled over the circulation and simulated times taken must correlate with the relationships given in the adjacency matrix. Furthermore, the dimensioning of spaces can be checked by considering the volume of persons present in each time step: In the simplest case, the occupation is related directly to the presence of persons in a space (e.g. in entrance areas). Moreover, presence in a space may relate to waiting for a shared function (e.g. examination), which can be modelled as server with a specified number of resources (e.g. two doctors) and one or more queues. By correlating the observed size of queues with the space requirements for waiting areas (distinguished e.g. for sitting and lying patients), it is possible to attain a hint at minimal areas required. Norms and regulations further contribute to these space requirements, which could be checked using approaches from automated building code checking [23], albeit in a simplified form.

A further opportunity for comparing the placement of spaces is that of building physics simulation: Some workplaces might require daylight; others must be protected from it. Preliminary environmental simulation (light, shadowing, wind), can also hint at energy demands which are elaborated in later phases.

3.2.3. Movement, circulation and traffic

Different options of route choice can be simulated by either assigning way-points between subsequent activities of the process explicitly (e.g. delivery of goods in zigzag shape, one floor at a time) or by interpreting the circulative network as graph on which shortest paths are computed. As a matter of fact, spatial arrangements can be judged by the time it takes to move across the circulation (which is also depending on the processes in place). A separation of traffic can further be achieved by attributing the circulation with allowed types of traffic (e.g. patients, visitors, staff), and taking these into account during either automatic or manual route planning. A further attribution of the circulation arrows has to be performed for distinguishing between horizontal traffic (taking place in the same level) and vertical circulation (lifts and stairs), among which movement models and speeds might differ.

3.2.4. Usage

Functions give the purpose (or intent) of spaces, processes model their planned usage over time. By coupling activities of the process to the underlying functions, a static check for unused function or activities that have no reference to a function (i.e. the underlying spatial concept) can be made [8]. Furthermore, temporal usage of functions obtained via process simulation can be used to compare the prominence of the spaces containing them and help think of possibilities for multi-functional use: Areas that are used only part-time (e.g. lunch room) may be conveniently used for other functions (e.g. meeting room) during the rest of the day.

4. An early-stage hybrid simulation

Our hybrid approach extends process simulation through agent-based simulation in two fundamentally distinct ways:
Figure 2. Overview: Processes are fed into a BPS, which communicates bi-directionally with a set of ABS performing the actual ‘dynamic’ work.

- **Intrinsically** by adding dynamic behaviour to nodes inside a process (see Section 4.1).
- **Extrinsically** by invoking external simulations that perform analysis, using the current state of the process simulation to generate arrivals (see Section 4.2).

Figure 2 gives a general outline of the model that we are employing: A process definition is fed into a BPS, which communicates bi-directionally with an ABS. BPS and ABS differ in their time bases (discrete event versus simulation in seconds); therefore, some synchronization is required. In simplest approach (which we have undertaken), the BPS’ scheduler was customized to progress in seconds instead of advancing to the next future event. We are aware that this is a very odd intervention – after all, discrete event simulation is about having a future event list that controls progression of time. One can, however, constrain the ‘progression in seconds’ to cases where ABS are active, in which case the BPS are required to wait anyway.

4.1. **Adding dynamic behaviour to a static process**

In order to inject dynamic behaviour into a static process, we introduce a new type of activity coined as ‘agent node’ (see Figure 3(a)): This is essentially a proxy for an agent simulation that is to be executed on behalf of the BPS. There are different types of agent nodes, with each one standing for one dynamic aspect that is missing from the static process.

Figure 3. A process in which (a) an agent node was embedded. All nodes are (b) annotated with a room stamp. Agent nodes furthermore hold additional parameters that are (c) passed to the underlying ABS.
The example we have implemented is an ABS of pedestrian movement through the architectural schema (see Figure 3(c)); but generally, there will be several types of these dynamic models running in parallel, each having their own type of agent node and respective ABS running in the background. The agent node itself holds parameters specific to its model (e.g. ‘from’ and ‘to’, see Fig 3(b)). Process simulation is then performed in the following manner:

1. Upon entering an agent node, the current process is passivated.
2. A message is broadcasted to the ABS connected with that agent node type, which the process ID and parameters mentioned before. The ABS creates an agent having the given process ID and initializes it with the parameters given (in our case: computes route along the circulation, from the source to the target space).
3. After all processes have been executed for this time step, the process simulation sends a message to the ABS to start simulating this time step and waits until this is done. This effectively hands control over to the ABS.
4. After receiving the command to start simulating, the ABS advances by 1 second. It then sends a message back to the process simulation that it has finished, also containing the IDs for all agents that have reached their goal (i.e. finished simulating).
5. After receiving the finishing messages of all ABS, the BPS reactivates processes with the given IDs and goes to the next simulation round.

Some further logic may be also embedded into the BPS to announce process creation and termination, for example, creating and removing agents at the same time as in the BPS (eager creation), and to use agent nodes in order to assign tasks. In this manner, information can be stored within each agent, which can use it accumulatively over the course of several calls to the same model.

4.2. Using processes as inputs for dynamic analysis

The proposed simulation approach has so far only used ABS intrinsically, that is, as delegate that executes dynamically within the process. By extending the approach to pass the set of active processes as arrivals for an ABS that runs externally, one can use it as analysis tool covering the aspects mentioned under Section 3.2. As illustrative example of such a tool, we have implemented a throughput visualization (see Figure 4(b)) that computes flows through functional areas by utilizing a pedestrian simulation algorithm ([24], see Figure 4(a)).

As pre-step and requirement for this extension, the nodes of the process have to be attributed with room stamps (see Figure 3(b)). The ABM must be able to derive physical locations within the design from these – for example, by using lookup table or through interrogation of the design itself. In full detail, the model consists of the following steps:

1. The BPS samples all processes at time $t$. For every process, the room stamp attached to the executing node is recorded. Counting the total number of times that it appears then yields the occupancy per room stamp, which is then broadcast as message to all ABS. The BPS again can continue simulating, since the analysis is running completely detached (i.e. extrinsically).
2. Upon receiving a simulation request, the ABS create agents in the given rooms according to occupancy. The agents are arranged randomly and until the capacity
of the containing space is reached (in which case an error can be reported saying that this room is under-dimensioned).

(3) With all agents being in place, the simulation starts and computes its respective aspect. In the example given in Figure 4(a), agents perform an egress simulation to the nearest reachable exit. The simulation records the flow through each area they pass, and visualizes the throughput as ‘flow lines’ with thickness according to pedestrian volume (Figure 4(b)).

In practice, \( t \) will correspond to ‘interesting’ points in the operational schedule of the hospital (e.g. early morning, morning, mid-day, early afternoon, night/weekends). Differences between the results produced at times \( t_i \) are especially interesting for looking at the building under different usage scenario which vary over the day. For example, the variation in throughput over a set of arrival times can be depicted as colour of the flow lines – red for large variations and green for minor variations.

4.2.1. Implementation details

We have extended a commonly used process modelling platform (Microsoft Visio\textsuperscript{TM}) for which a multitude of BPS exists (e.g. ProModel\textsuperscript{TM} Process Simulator, Simul8\textsuperscript{TM} and Arena\textsuperscript{TM} Integration). However, because we needed to customize the scheduler, we coded our own BPS in Java\textsuperscript{TM}, based on the open-source graph-based programming code provided by [25]. As ABS platform, we have employed NetLogo [26], which is also available under open-source license terms.

In brief, the system we have implemented works as follows: we first start all ABS. Then, we let Visio\textsuperscript{TM} output the process model as file with which the BPS is invoked. From then on, the BPS communicates quasi-bi-directionally with all ABS in the following fashion (refer to Figure 2): ABS act as servers, listening on a certain port for incoming messages. The BPS is a client, issuing a request at regular intervals (e.g. 0.25 seconds). To buffer messages, BPS and ABS each have two message queues (incoming and outgoing). One communication round is as follows: in each request, (1) the BPS removes one message from its outgoing queue and (2) sends it to all ABS. If there is no message to transmit, the BPS issues a ‘nop’ (no operation, nothing to be done) message instead. The ABS (3) receives the message by the BPS and puts it into its incoming queue lest it is nop. It then (4) responds.
to the request by removing one message from its outgoing queue or sending \textit{nop}. The BPS (5) receives the message and puts it into its incoming queue, if it is not \textit{nop}. At both sides, the incoming and outgoing queues are manipulated by the programs, the sending occurs automatically.

The exchanged messages themselves correspond to the scenarios we are supporting. For intrinsical simulation, we can \textit{create agent with a certain ID in given room} (e.g. ‘process 12 entrance’) or \textit{simulate one second} (‘step 1’). The extrinsical simulation understands \textit{create occupancy in room} (e.g. ‘occupancy entrance 9’), \textit{simulate until completion} (‘start’). The ABS knows only how to announce that it has finished simulating (‘end’).

The actual implementations used for the ABS are based upon the previous work [27–29] and fall beyond the scope of this article: In essence, what we want to show is the feasibility of the ‘in-process’ approach, and not reflect on already published material. What is common to both ABS is that they interact with the preliminary design (e.g. exported from AutoCAD™ by plotting at a resolution of 0.5 m per pixel, then colour-coding and importing into cell space).

5. Including the toolkit into early-stage design

Under Section 3.1, we have presented the early design process of hospitals from the viewpoint of the two main professions involved (Building Organization and Functional Planning). In this section, we intend to elaborate how exactly the presented concepts fit into the co-evolutionary planning work and can help take design decisions (see Sections 3.1.3 and 3.2). As will be shown, intrinsical simulation is for the simulation of individual activities, whereas extrinsical simulation is intended for analysis.

In both cases, processes and preliminary spatial concept need to be available: They act as a model for the overall simulation. The input (i.e. patient arrivals) is to be generated either by use of an assumed distribution or, preferably, by reference to actual patient volumes (e.g. from a similar clinic). These data are extractable from an Hospital Information System (HIS), although quality may vary; the usual case is that it is known when a patient arrives (timestamp) in a certain functional unit (e.g. radiology). From this information, the arrivals can be reconstructed to a certain degree. However, and as a word of warning, examination durations are not adequately recorded, as waiting times in between consecutive stages are not explicitly mentioned. The best approach is therefore to measure service times at the functional unit directly.

5.1. Applying intrinsical simulation

\textit{Intrinsic simulation} approaches the planning problem from the side of Building Organization, using BPS as primary and ABS as secondary/subordinate simulation. This mode is synchronous as seen from the BPS side, since the ABS executes until the agent that stands for the process instance is released. The proposed Agent Nodes are delegates that execute \textit{dynamic activities} that depend on and influence the spatial concept of the building. Referring to Section 3.2, these tasks range from way-finding processes of building users under consideration of visibility and accessibility (Sections 3.2.1 and 3.2.3), possibly also influencing the dynamically changing space layout and dimensioning (Section 3.2.2) to questions of resource allocation and queuing (Section 3.2.4). This means that ABS used in this context to serve a single activity and purpose, with no synchronization in between multiple ABS running in parallel. In fact, these models might not even be made visible to the user without request – the primary tool is the BPS.
5.2. Applying extrinsic simulation

Extrinsic simulation stops execution of processes at time \( t \) and uses simulation to gather performance indicators over the spatial concept under the current occupancy at that instance. The questions presented under Section 3.1.3 are the context in which this analysis happens. However, we have so far left out how to actually measure and analyze these factors, given the setting that the process and spatial model used in early planning is by definition preliminary, not to be assessed quantitatively but rather qualitatively. As an example, adjacency specification between each two functional areas can be specified as ‘adjacent’, ‘neutral’, ‘dislocated’ and ‘not applicable’ (see Figure 1). The analysis of a spatial configuration under the influence of the simulated process would first try to map the distance between each pair of spaces along the circulation to these linguistic categories, and then weight that result according to simulated traffic (i.e. ‘high traffic’, ‘medium traffic’, ‘low traffic’). Both points can be conducted efficiently by using fuzzy architectural spatial analysis [30], which represents a linguistic term by a membership function and outputs a continuous likeliness in \([0,1]\). This output would then be either presented directly or be interpreted automatically using, for example, a rule-based system that outlines the competition requirements. A good choice for such a rule-based evaluation that results in yet another (derived) membership function would be the domain of Fuzzy Inference Systems (FIS), for example, the Mamdani-type or Sugeno-type FIS.

6. Outcome, summary and conclusions

We have presented a simulation approach by which dynamic entities can be embedded into an otherwise static process. Our efforts are targeted at early stages of hospital planning, where simulation has the potential to become a design tool for qualitative decisions among a multitude of design variants. We have presented two different modes for such ‘in-process’ agent-based simulation: (1) intrinsical simulation: introducing nodes that add dynamic behaviour and (2) extrinsical simulation: passing all active process instances to and agent-based simulation that performs a (spatial) analysis for the sake of decision-making in the design process, for example, for comparing different spatial configurations. The latter simulation needs in early design have been argued for in an own part (see Section 3.2), which is also a contribution in health care design, in which we see the true impact of our work: The possibility to shift simulation from late stages of design (where possibilities of change are limited) to early stages (where design decisions are of fundamental significance) continues to be our main goal in research.

Acknowledgements

We would like to acknowledge the support given by Niki Popper, Martin Bruckner and Felix Breitenecker who have been of vital importance since the early phases of this paper. Furthermore, we wish to thank Simon Peyton-Jones and Jan Michl for their contribution to scientific presentation and discourse.

References


**Paper: Function & Action**

The inclusion of building users into the early-stage design process is often done in the form of interviews and workshops, lead by planners on the client’s behalf. However, the information supplied in such context is often difficult to put into a planning perspective - especially when there are different heterogeneous interests and agendas involved. The idea of this paper was to sample daily routines of the staff, excluding the indirections that result from interviewing. To facilitate a simple data entry even for untrained users, we came up with the idea of using a game metaphor (see Figure 38): The user steers his avatar through a schema of the future building, being offered the functions underlying the zone he is currently in. He can now choose to perform the activities of his daily work routine, which is recorded in a database and can be exported (for example into an ABS). As further benefit, we can bring all functions of the schema that have not been used to the attention of the planner. Vice versa, if some needed function does not exist in a space, the user can allocate one (which is again reported back to the planner). As extension (also mentioned in the paper), we name causal relationships within user routines - substantiated in later work (see p. 145).

**Authorship Information.** The paper is the result of a two-day long workshop at La Sapienza in Rome together with my co-authors Antonio Fioravanti, Gianluigi Loffreda, Armando Trento and Gianfranco Carrara as host (not on the paper). The initial inputs of this meeting were then compiled into a paper74, for which I wrote an accompanying program.

**Main Contributions.**

- Argues for the benefits of gathering daily routines of building users without an intermediate mediator.
- Lets building users test a preliminary design by walking through it in a game-oriented environment, providing feedback (p. 391 of the paper).
- Lets planners perform a consistency check on unused or missing functions (p. 393 of the paper).

**Type of Work.** Peer-reviewed conference paper with accompanying (web-based) implementation.


---

74 my contribution regarding the paper was around 60%
Function & Action

Verifying a functional program in a game-oriented environment

Gabriel Wurzer \(^1\), Antonio Fioravanti \(^2\), Gianluigi Loffreda \(^3\), Armando Trento \(^4\)

\(^1\) Vienna University of Technology, \(^2,3,4\) Università degli Studi di Roma “La Sapienza”

\(^1\) wurzer@iemar.tuwien.ac.at, \(^2\) antonio.fioravanti@uniroma1.it, \(^3\) gianluigi.loffreda@uniroma1.it, \(^4\) armando.trento@uniroma1.it

Abstract. The finding of a functional program for any kind of building involves a great amount of knowledge about the behavior of future building users. This knowledge can be gathered by looking at relevant building literature (Adler, 1999; Neufert and Neufert, 2000) or by investigating the actual processes taking place in similar environments, the latter being demonstrated e.g. by (Schütte-Lihotzky, 2004) or new functionalist approaches of the MVRDV group (Costanzo, 2006)). Both techniques have the disadvantage that the architect might assume a behavior which is seldom experienced in real life (either through lack of information or by failing to meet the building user’s expectations). What is needed is a verification step in which the design is tested on real users. We have devised a game-like environment (Figure 1a) in which it is possible to capture the behavior of future building users in order to verify the relevance of the design even at a very early stage. As result of applying our approach, we can find previously overlooked usage situations, which may be used to further adapt the design to the user’s needs.

Keywords. Requirements Checking, Participative Design.

Introduction

Lawson states that “the best test of most design is to wait and see how well it works in practice” (Lawson, 1997). We feel that this ‘reality check’ has to be done well before the building goes into operation, in order to meet the future building users expectations. It is an often quoted dilemma that people cannot state their expectations about the daily life routine in a new building without reference to some sort of solution. As architect, taking assumptions in the form of an initial functional program is good, what is missing is a validation step in which the preliminary design is verified on real users.

We have devised a game-like environment in which users can enter their would-be behavior in a planned building, based on their normal work and life routine. Our key goal is

- to let building users provide feedback on the presented design (e.g. what they think is still missing, see page 391)
- to be able to tell which planned functions are actually used (see page 393) and at what time or because of what cause
As game setting, we take the preliminary design (Figure 1b) which the architect has annotated with function map (refer to Figure 1c): For us, functions are spatial and have an influence boundary in which they can be utilized. We adopt the notion of function as proposed in (Wurzer, 2009), in which functions are further detailed into basic elements of work routine (e.g. kitchen into cook, prepare, store etc.). As final input into our program, we use an accessibility map (Figure 1d) that specifies what pieces of the design can be walked on.

Given a goal verbally (e.g. “go through your working day”), a future building user may now walk his virtual representation through the design, being able to perform actions as he goes along. The actions the user can perform in any location correspond to all functions in close proximity (i.e. functions that have the player’s position contained in their boundary). Should none of these functions fit the desired activity of the user, he can enter one himself (“other”-button in Figure 1a). Furthermore, the player might leave a comment for the architect at his current location (“note”-button), similar to what e.g. (Moloney, 2002) proposed in his “String Collaborative Virtual Environment”.

Another important piece of information about the building user’s behavior is the time or event that triggers it (e.g. “at 11am”, “from 9am to 6pm”, “when a call comes in”). This is entered every time the user wishes to create a new behavior. As a matter of fact, we can infer temporal and causal usage of functions, which would otherwise be hard to draw from just using the functional program.

Contribution, benefits and applicability of our approach

Out of all behaviors that have been entered by future building users, we can infer the usage of functions both temporally and causally (see page 393). The architect can view the behaviors on top of his design and can check whether his initial assumptions concerning usage were correct. Furthermore, he may notice which functions remain rarely used and which functions seem to be missing. The latter is done by taking all actions for which the user has entered a function himself.

It shall be noted that the process of testing a design should begin in a planning stage which still brings the possibility to adapt some of the planned functions (or else it would make no sense to verify). As input, we take the design (CAD drawing or image, concrete or sketch) that is manually attributed with functions by the architect. The work on the side of the building users - going through their daily work routines, specifying when each part of the work routine starts etc., can be done in parallel to the further planning work. After that, we require an analysis.

Figure 1
(a) Game Environment based on (b) preliminary design, (c) function map (d) accessibility map to record behavior of user.
phase (see page 393) from the architect’s side to view the gathered data, go through it and check the initial assumptions made in the design. This process can be iterated – or adapted to gradually bring more details into the work routine (e.g. users first give their high-level tasks, then go to special cases in subsequent phases).

**Evolution and Related Work**

The design of the game environment proposed herein is a natural extension of recent work that lets planners enter planned processes in an architectural schema (Wurzer, 2009). However, those processes are only correct if the planner can correctly anticipate the usage of the building by different building user groups. Because this can be a daunting task, we thought of an approach that could be used by the building users themselves to enter their activities, either alone (i.e. in an unattended game session) or in a workshop environment.

Apart from our previous work, the possibility to enter activities and behavior in a CAD environment has previously also been proposed by (Ekholm, 2001). However, the novelty in this case is the questioning of the building user as means of participative design, as this has (to the best of our knowledge) not been presented before. We are aware of approaches that use game technologies to measure user satisfaction on design variations (Orzechowski, Timmermanns and de Vries, 2003; Orzechowski, Timmermanns and de Vries, 2000), however, these seem to be of more use in the late stages of the design process when the final form is being generated. Others have tried to formalize the future building user’s requirements as rules and to execute these in a game environment (Trento, Loffreda and Kinayoğlu, 2009). However, as the authors conclude, this formalization involves a lot of design knowledge, which the users may not possess. A mediator (program or human) should aid in translating the user’s preferences into formal rules. We definitively support these considerations, as an analysis step is always needed after data (being it rules or behavior) has been gathered, which we also do in this approach.

Apart from architectural planning and verification, there are other fields also pointing into the direction of developing a more graphical questionnaire for paths taken and actions conducted: NIST has conducted a survey with occupants of a 32-story high-rise office building who had to evacuate due to a fire incident (Kuligowski and Hoskins, 2009). One of the main interesting points asked was the set of specific actions taken by the respondents. However, as the survey was form-based, there was no hint on “where” actions were performed. If a location-based interviewing tool had been available, survey data would have been more context-specific. Another important aspect that has lead to the invention of the tool is the sampling of usage for simulation. However, we will not cover this aspect in this paper, as we rather rely on using statistical (i.e. spreadsheet) analysis of the entered behaviors. Also interesting is the relation of this work to storytelling approaches such as (Kelleher, 2006), the true difference being that we want to enable a user to tell a story about himself, not somebody else.

**Description of the game environment**

**Entering behavior (building user)**

As seen in Figure 1a, the game environment is a tool that basically offers an avatar centered on a design. This avatar is then moved over the design with the cursor keys, taking the accessibility map (spots on which walking is permitted) into account. At every location of the design, the program furthermore knows of utilizable functions using the underlying function map. Accordingly, it will update the user interface in which the user can choose what function to call, or put differently: what action to perform. The separation between function and action is crucial to us (also refer to Wurzer, 2009): While the function is defined by the architect, defining “what could be done in every space”, the action is an actual
instantiation of a function (i.e. a function that was put into effect). A user may enter a different action than the ones that are suggested, therefore indicating that the functional program was somehow incomplete. Furthermore, one may give feedback for the avatar's current standing position as a note on the map, which appears as a label.

As the user moves the avatar over the layout map, all steps which he does are recorded. In the user interface, the corresponding functionality is included as a metaphor in the form of a video recorder: Pressing the "new" button starts a new (empty) process. It is noteworthy that we have the clear concept of allowing only one process to be edited at a time, of which we display the name in the top-left corner. Toggling the "record" button records the movements of the user. The timeline at the middle of the screen gives the current position within the process that is described. Once the user has toggled the "record" button, he may not move his avatar further without having record on (i.e. we do not allow for jumps in the process, it has to be continuous). At any time, the user may drag the time slider at the bottom of the screen to a certain instance in the recorded time, thereby updating the position of the avatar. The user can insert behavior from the current time onward using again "record". Furthermore, as in film, we have the ability to cut at a specified position, using the "scissors" button. The cut will range from the current position to the end, having again the ability to add new behavior at the end.

The play button will play back the process beginning at the current time. Should the current time indicator surpass the recorded duration, the time indicator is reset to the begin of the process and the playback continued. Furthermore, we have "jump to start" and "jump to end of behavior" buttons to ease temporal navigation. A process may be saved by using the "load" and "save" buttons at the top of the tool.

If the user selects a function from the list of functions at the very bottom of the screen, the process inserts a new action into the process. The action is similar to a key frame in animation – it is a fixed point in the behavior that executes a function. As per default, each action has is one second long. However, the user has the opportunity to specify how long it takes by specifying a value in the "takes" editing field. One can jump to the next and previous action by pressing the "next" and "previous" button. Furthermore, as the time slider is dragged on an action, the used function becomes selected in the functions view at the bottom of the screen (and can be changed to another function if the user wishes).

Besides actions, notes can be left at the avatar's current position using the "note" button at the lower right part of the screen. They are a valuable tool for communication between architect and building user, since they can give a connotative feature to the process (from a metaphoric standpoint very similar to subtitles). Once created, notes can be selected and edited using the mouse.

To end with, each process has a starting condition. This is either (1.) a time when the process is scheduled to start, or (2.) an event which triggers it. Events can either be the execution of a function or the performing of an action by a named process.

- If a time is given as process start criteria, an according dialog appears. The user can determine whether to enter a range (e.g. 1am-2am) or a single time stamp.
- If a function is taken as starting condition, the user has to select from a dropdown list which function should trigger the process. If there still is no such function, the user can enter a new one.
- If an action is taken as starting condition, the user is presented with a list of all processes and contained actions and selects one from it. Two processes that have the actions of the same name are handled as different (internally, the process name is prepended when issuing the event).

Behavior analysis (architect)
The architect can perform analysis in the application in multiple ways:
• By loading the activities the users have entered into the application depicting the trajectory and function usage on the design (static image, no simulation or animation of the routes) and comparing this to the initial assumption over the building user’s behavior.
• He can see functions with no usage in any process displayed prominently on the map. These functions are likely to be unnecessary – i.e. they are candidates for removal.
• Vice versa, functions which have been entered by the user himself (because there was no function that matched the action in the mind of the user) can also be shown prominently, as they give a hint at forgotten usages.
• The notes that were entered by the building users can also be shown on the design. Furthermore, there are cases when a usage itself is to be questioned – the interesting questions being when and why this behavior occurs. Using the entered starting conditions for every behavior as basis, we can generate a report (standard spreadsheet) containing a specific view on the data:
  • Outputting times at which processes start can be a hint at a possible incompleteness in the description of the usage of the building (when).
  • We can also output for each process the corresponding start function or action (why). This can be beneficial when there process is complex and its cause is not clearly known (being part of the daily work routine does not necessarily mean that it is reasonable).
  • Exporting quantitative measures can furthermore give a measure of importance for each function. The key figures we use are how many processes a function starts and how often it is used in a process.

Limitations
We have certain limitations in our approach that we know can annoy advanced users, but have novices enter processes in a more intuitive way. First, when using our cutting tool during behavior editing, we discard everything in the timeline to the very end of the recording. Second, we have no notion of junctions in behavior (i.e. “alternative behavior” that executes from a certain time on). This could, in principle, be beneficial from the planner’s point of view (i.e. seeing all alternative behavior at once), however, we have left it away since alternative behavior can be decomposed into several behaviors. Third, actions can only be attributed durations by stopping the recording and getting back to the last action and then setting the duration to the correct value (the time is inserted into the timeline). Last, we cannot distinguish between a note left because of thoughts concerning the process and a note left because of the preliminary design. Therefore, clicking on a note has to bring the corresponding process up, in order to provide further context.

Performing User Tests
We have developed our ideas by using mock-up testing on five un-trained candidates with no background in architecture or CAD. In detail, we took a cardboard frame with an attached player figure in the middle and laid it over a large design of a high-rise office building. In order simulate the games world moving, we then subsequently moved the design. Furthermore, flash-cards corresponding to the functions at each locality in the map were laid before the candidates, with the possibility to choose one so as to perform an action. Recording, playing back, cutting and inserting behavior in close correspondence to a video recorder was taken up after a temporal list of actions failed to convince. Another feature originating from input by our users was the possibility to leave notes or feedback at locations on the map.

For the analysis part, we have performed mock-up tests with two architects being the subjects, taking only the graphical part (missing functions / unnecessary functions / notes) into account. We could, however, not observe accumulations of missing functions but rather isolated cases. A variety of such
missing functions were proposed, not all legitimate, and some through misinterpretations of existing functions. Clearly, there has to be an analysis phase for guaranteeing validity of the entered data.

As next step after the mockups, we are currently in the phase of programming the actual application (a web application in Adobe flex) which will be ready until the eCAADe takes place in September. Until then, also, data collection in the context of actual project work (in the hospital domain) will be gathered in order to have a real case to put forward.

Concluding

We contribute a novel approach that uses a game environment to let prospective building users walk an avatar through the preliminary design, performing their daily work routine as they go along. Each action within the work routine is the invocation of a function from the pre-defined functional program of the building. Should the user want to perform an action for which he cannot invoke a function, he creates one himself (missing function). Functions which were not used in any work routine, on the other hand, are probably not needed (unnecessary function). As benefit, both of these situations can be brought to the attention of the architect even in a very early stage. Furthermore, our approach may also be used for the temporal and causal origins of usage scenarios.

References


Article: Towards Use Cases in Sparse Architectural Data Exchange

As an excursion into data exchange, I looked into similarities between information systems in the Architecture/Engineering/Construction (AEC) field and healthcare. While this comparison is not obvious at first sight, the trend of serving applications “into the cloud” will likely lead to a situation in which the repositories that store AEC data will be online as well - just as they are in the healthcare. In recent years, developments in that domain have shifted the focus of data exchange away from data structures towards the specification of use cases in which an exchange takes place. The AEC field is currently still in the first phase: Ongoing development in Building Information Modelling, most prominently the Industry Foundation Classes (IFC) specification have led to the ambition of defining a data model in which all information of a building’s lifecycle can be represented. However, that does not mean that such information is always present in an ifc file. In fact, most of the times, the data representation will be sparse - every application exports only the parts it finds relevant. This might lead to troubles even in the same application domain, if vendors choose to include different parts or representations of information, making an import impossible. Concepts from healthcare interoperability promise a solution to these problems, which is why I tried to bring them into the AEC field.

Authorship Information. I have written this article alone, based on my professional experience in the field of e-Health (most notably T-Systems and Systema Human Information Services).

Main Contribution.

- Argues for use cases as basis for sparse data exchange, in analogy to the “Integrating the Healthcare Enterprise” (IHE) framework that is being successfully used for communication between different heterogeneous information systems in the healthcare sector.

Type of Work. Peer-reviewed article.

Towards use cases in sparse architectural data exchange

Gabriel Wurzer*

* Architectural Sciences, Vienna University of Technology, Austria, Email: wurzer@iemar.tuwien.ac.at

Data exchange continues to be one of the most challenging problems in the Architecture-Engineering-Construction (AEC) sector. One aspect that has still not been covered to date is a clear definition of use cases in which a data exchange occurs. The definition of data exchange with reference to use-cases is beneficial, because exchanged data may be limited to include only elements that are needed in the current context (sparse data exchange), so that interfaces between applications can be kept simple. Our work draws on experiences in healthcare data exchange, which has successfully implemented exactly these concepts in the past years.

Keywords: data exchange; use cases; standardization; frameworks; small is beautiful

1 Introduction

Data exchange in the AEC sector is currently built around data formats which are containers for a multitude of potentially useful information; however, this potential is not fully utilized when importing into an application, since only certain aspects of the data may be interesting. Vice versa, applications may supply only a sparsely-filled file when exporting; thereby failing to provide data that would indeed be useful.

At first, it might seem that the answer to these data provision problems are already provided by current Building Information Modelling (BIM) approaches and according standards such as the Industry Foundation Classes (IFC). However, we argue that these problems of overrepresentation and underrepresentation have nothing to do with the data model being used, but rather come from the lack of consideration of the context in which the data is to be exchanged. For example, both a Computer Aided Design (CAD) package and building physics software might make use of IFC. This does, however, not guarantee that information needed on either side will be present in the exchanged file - it merely guarantees the syntax and semantics of data being present.

If data interchange would focus on use cases, meaning 1) in what context, 2) which information is to be exchanged using and 3) what data format software vendors could concentrate on adding new use cases rather than invest in supporting new file formats. The transmitted data will be sparse – i.e. “good enough” to fulfil the use case, but not more. As a matter of fact, interfaces would be kept simple and easier to keep stable.

Quite luckily, the presented ideas are not new. In fact, healthcare data interchange has introduced them about a decade ago, in a framework called...
Towards use cases in sparse architectural data exchange

Integrating the Healthcare Enterprise (IHE). Taking these already-defined concepts into architecture is the main contribution of our work, which is further broken down into:
- a description of the IHE framework, with special emphasis on use cases for sparse data exchange (Section 3)
- the transfer of these concepts into the AEC domain, using the example of a CAD package coupled to an evacuation simulation (Section 4)
- an in-depth discussion of potentials and limitations of the presented concept, both technologically and with regard to the professional world (Section 5)

2 Related work

The need for a collaborative data exchange lies in the nature of planning projects, in which actors across multiple disciplines hold data in a variety of representations. The key question of how to bring information from A to B has a long history in the AEC sector: Early product model formats such as IGES concentrated on the syntax of the exchanged entities, but failed to supply additional semantics. This restriction was lifted to a certain extent with STEP, which also considered functional aspects of a product model. Recent exchange formats such as IFC further introduce an object-oriented representation of entities, which is a basis in all BIMs. The taxonomies found in these approaches are coherent but fixed, based on strict hierarchization and inheritance as known from object-oriented programming (OOP). Newer approaches work on lifting this predetermined structure by additionally representing and transmitting design knowledge (i.e. semantics of entities, inheritance relationships, etc.).

Further work diversifies the interchanged knowledge into General Knowledge and Specialist Knowledge as ontologies tailored to the involved discipline. A step-wise data interchange triggered by use cases has, however, not previously been looked into.

![Figure 1 IHE use case consisting of (a) general description (b) workflow and (c) technical specification](image-url)
3 Use cases in healthcare interoperability

In the healthcare sector, data is segregated in between multiple information systems, each of them storing data according to the needs of the involved medical discipline (e.g. laboratory, radiology, administrative staff). Previously, data interchange between these systems was being conducted in a custom-tailored manner, by exchanging precisely the data needed in a standardized format. However, this approach was costly and hard to support, given that every hospital had its own landscape of systems and, accordingly, its own way of implementing data exchange among them. In order to improve the situation, a framework specifying exactly what data would need to be exchanged in precisely what form was therefore released under the name Integrating the Healthcare Enterprise or short IHE.4

IHE defines data interchange use cases (i.e. typical situations in a workflow in which data interchange occur) by first giving a brief description, intended purpose and context in the workflow (Figure 1a). This rather non-technical description of scope is elaborated in a community process among industry partners and professionals.

After that follows a definition of the use case as seen from workflow perspective, in the form of a diagram listing data exchange transactions among two or more actors of the workflow (Figure 1b). These actors can be thought of as being software components, in the simplest form: two information systems wishing to exchange data. Transactions describe a sequence of actual data transfers, i.e. messages being exchanged. As systems in healthcare are always on-line, this usually happens synchronously.

The most important part of a use case is the specification of each transaction by prescription of data formats and their filling (Figure 1c). In this context, IHE does not invent new formats – it rather uses formats that are well-established and regulates how they are to be filled with data that has to be transferred. The work behind this specification is usually done by industry partners, again in a community process.

A use case which has been fully specified becomes ready for a trial implementation phase (usually lasting one year). In this phase, industry implements the specification into their software products and gives feedback on shortcomings or possible augmentations that should be made. After the trial phase has ended, the use case is released for general implementation and subsequent testing, in the following fashion: Twice a year, IHE holds huge developer gatherings (so-called Connect-a-Thons), in which all software that claims to implement a specific use case is tested for interoperability. Applications implementing the same use case are therein required to each take the role of the different use case actors, and perform the necessary transactions between them. If there is no error, IHE certifies that the partners have correctly implemented this actor. An application having successfully tested all actors and transactions of a use case is furthermore handed a formal certificate (Integration Statement) that can be used when advertising the product or to comply with tenders: Usually, software is required to have a certificate no older than two years, thus ensuring that continuous testing has been taking place.

Towards use cases in AEC interoperability

Software being used in the AEC sector is fundamentally different than that in healthcare:
- healthcare systems are online systems, i.e. centralized servers being reachable around the clock, with a variety of interconnections to allow for message passing
- AEC software is decentralized, typically running on a single workplace

In technical terms, this difference leads to the usual asynchronous data exchange for the AEC domain (file export / import), whereas healthcare systems can communicate synchronously over the network. Clearly, a centralized communication layer linking different decentralized applications together would be possible, e.g. in the form of a common BIM server to which applications can publish messages containing data while at the same time being informed of messages to which they have subscribed. The description of such a server is, however, beyond the scope of this paper. A good starting point for further work on this subject would be for instance Beetz and Chen et al.

For the rest of this paper, we assume three different modes of data exchange:
- an exchange using files, i.e. by repeated steps of exporting/importing. In this case, intermediate communication of files happens asynchronously, using any means of transport in between (e.g. email)
- an exchange between different applications running in parallel (either on the same machine or on different hosts in the same network)

This scenario is synchronous, as applications may pass back and forth messages and get results without requiring further user interaction to facilitate the exchange. In technical terms, this requires applications to be implemented so that they listen on a specific port for incoming messages (i.e. an application becomes a server). Third, we assume that each application is connected to a BIM server that facilitates a publisher/subscriber data exchange (as outlined before).

---

**General Space Group Query**

**Description, Purpose, Context in Workflow**
Provides a way in which Spaces (i.e. bounded regions of a building spot or building) can be exchanged, supplying a query as criterion. Relates to: IFC 1.0, IfcSpace.

**Involved Actors and Transactions**

![Figure 2 Use case in which a Spaces Consumer queries spaces of a Spaces Provider](image)

**Specification of Transactions**

- Query Spaces" uses ECMA-357 (E4X) as query language and IFCXML 2x3 as data format. If no query is given, all spaces are returned in the following fashion:
- further details on data format and prescription of data items having to be sent...
Basic Use Case

As next step, a prototypical development of a use case is outlined, taking the case of interoperability between two applications wishing to exchange space data (i.e. bounded areas within a building) as example. The use case will be called “General Space Group Query” (Figure 2) and can be employed in a wide range of areas, e.g. adjacency analysis during early planning or topological considerations. The use case considers the interchange of space data between the actors “Spaces Consumer” and “Spaces Provider” using a transaction “Query Spaces” as means. The use case prescribes that data semantics introduced by IFC 1.0 (Entity IfcSpace) will be used in the data format IFCXML, Version 2x3. The query itself has to be formulated as defined per ECMA-357 E4X, a query language targeted at xml. Table 1 lists an example of query and response, for illustrative purposes.

The choice of prescribed standards would in reality be based on a community process, and thus could arrive at a completely different choice (e.g. using DXF as data format and an xml containing query parameters explicitly rather than using a query language). Because the use case is composed of just one transaction, the physical data exchange is simple in all three modes of transfer (asynchronous/file, synchronous/parallel, synchronous/BIM server). The true value of the concept comes into play when using multiple (possibly nested) transactions, described in due course.

Use Case Composition

Software is written in a layered fashion, with lower layers performing work on behalf of higher layers of functionality. Use case composition allows for this approach in an efficient manner, as basic use cases like the “General Space Group Query” presented earlier can be re-used. In the following example, we look at a CAD application exchanging data with a pedestrian flow simulation (Figure 3). The reason for taking such apparently different application types into the example is deliberate, since we want to show that the concepts presented herein are applicable not only inside AEC, but may extend the field further even to fields that are not encompassed by usual BIM data structures such as IFC.

The use case starts with the CAD application acting as “Flow Consumer”, initiating the transaction “Retrieve Flow”. As input parameters, an E4X query describing the spaces for which flow is to be computed and utilization matrix (expected occupancy for each space, in a custom xml format given in Table 2) has to be supplied. Typically, the latter data will have to be provided by the user, since a CAD package does not usually store data about expected occupancy.

- The pedestrian flow software acts as “Flow Provider”. Upon being invoked, this actor triggers a transaction “Retrieve Spaces”, which utilizes an underlying “Spaces Consumer” actor in the same way as presented in the basic use case example. After having obtained the spaces using the supplied E4X query, the pedestrian flow simulation internally computes the occupancy (using e.g. a pedestrian egress model as technical means). As result of this simulation, the occupation of each space is returned, again in custom xml syntax.

Again, the choice of file formats that were involved in the transfer would be determined in a community process, using available standards as means. However, in contrast to healthcare interoperability, it may not always be possible to avoid creating a new data format, which is why we have chosen to use a custom xml for the presented example.

From a physical data exchange perspective, use case composition is ideally suited for cases in which software is online (i.e. the synchronous/parallel and synchronous/BIM server scenario). A file-based exchange might involve too many import/export steps to be practical.

5 Discussion
The presented concepts for sparse data exchange based on use cases require, first of all, a re-thinking of data exchange on the physical level. The usual (asynchronous) file-based approach has severe limitations when dealing with
data exchanges consisting of multiple steps. Therefore, we strongly argue for the introduction of centralised communication which means to enable a synchronous transfer. Current tendencies in this respect are already underway, either in the form of BIM servers or central repositories that allow version control and collaboration among project partners.

From a semantic standpoint, we have shown that standards such as IFC can capture data representation needs, but not the typical interchange process that may employ only a limited set of data entities (“sparse data”). Use cases are beneficial for a definition of the latter aspect, capturing the data needs according to the workflow in which the data is transferred. Furthermore, the presented concept links data exchange to the data formats used by the involved domains, which might be reluctant to adopt each other’s standards. Healthcare interoperability has seen a wide range of these trans-disciplinary disputes (eventually extending to the standards that are now in place), and reacted not by dictating one, but rather superimposing a framework that regulates their use for specific interchange scenarios. If the AEC sector wants to extend its scope, utilizing same approach would be worth consideration.

Coming to the benefits and shortcomings of the presented concepts, the main idea is to supplement activities underway in the standardization field, not replace them. Therefore, argumentation in favour of the presented content focuses on gaps that exist in the now-common data exchange, i.e.:

- transferring as little data as possible in formats mutually agreed upon by the involved fields, thereby keeping interfaces simple and stable
- Seeing data exchange in the context of the intended workflow, such that applications should support new use cases rather than new data formats.
- regular verification of data exchange by interoperability testing (e.g. Connect-a-Thon) among the exchanging applications, rather by bi-lateral agreement on interfaces by involved companies

On the negative side, we are seeing that interoperability based on use cases can only come as far as the process in which the exchange occurs is clearly captured and remains static. Because data transfer is optimized such that it is “good enough” for fulfilling the use case (but may include little more) exploration of the data in unforeseeable ways is extremely limited. Also, it remains yet unclear whether dynamic steps in the design process can be captured in the proposed form.

For software users, the potentials of the presented concept lie in an optimization of daily work routine steps having to do with data exchange. More specifically, a user would see steps for which he would normally have to enter another software product being directly available in his platform (e.g. as in the use case composition example, there could be “Retrieve Flow” button in the CAD software). In the background, the necessary exchanges would be invoked and the result of the computation returned in a way that makes sense for the requesting domain. Exchange of files per mail, which often leads to problems of versioning and timeliness (e.g. recipient is currently on holiday) would be alleviated by using synchronous communication among software packages that are able to communicate on-line. Recent tendencies of using software in the cloud (software as a service, pay-per-use models) are already aimed at making software available in this manner.
6 Conclusion

We have brought the concept of a data exchange based on use cases, which captures in what context which data needs to be transferred, using previous work in healthcare interoperability as a basis. The reason for this lies in the fact that standards (e.g. IFC) focus on data representation, but give no hint at what data might actually be present once it comes to the actual exchange. Prescribing exactly what data is to be sent in which context enables software vendors to keep their interfaces simple, tailored to the workflow in which an exchange occurs. Furthermore, it also allows verification of software claiming to be interoperable in a straightforward manner, using the domain knowledge of the use case as test bed.

For the user, the potential of the presented concept clearly lie in the improved interchange between applications of different domains. The used integration approach does not define new standards, but leverages formats already established in the specialist field. This also means that users of different professions can work together by using their (domain-dependent) information, which are later linked together by the use case.

Acknowledgements

This paper is a direct follow-up that addresses earlier work done by the group of Gianfranco Carrara at the Department of Architecture and Urban Planning of Sapienza University Rome. The comparisons to healthcare data interchange using IHE are furthermore based on actual work in this sector, in which the input of co-workers at T-Systems and Systema (Reinhard Egellkraut, Harald Bartl, Peter Divjak, Karl Holzer, Alexander Göttermeier, Melanie Mairhuber, Alexandra Plank-Adam, Willi Salomon, Wolfgang Prettnner) has been of major influence. For sake of clarity, some concepts have been abbreviated or renamed (e.g. IHE Profile becomes use case).

Bibliography

Using Early-Stage ABS as Design Tool

Reports on contributions that seek to extend planning by ABS, establishing it as one of many tools that are already used in that context. As will be shown, its utility is not only in spatial design - it may also be used for mining into organizational processes as part of business organization planning.
**Article: Schematic Systems - Constraining Functions through Processes**

This article argues for a more rigorous view on functions, which (as Jan Michl puts it\(^75\)) architects and designers have always used as a carte blanche:

“The functionalist notion of function operated as a carte blanche: having been empty the notion of function made the architects and designers free to define it in ways that always legitimized their own aesthetic priorities.” Jan Michl, *Form Follows WHAT? The modernist notion of function as a carte blanche*, :49:

Tying functions to their utilization by activities of a simulated processes gives a natural way to justify their occurrence in a spatial programme: Functions not being utilized in any process are superfluous (or need to be justified by other argumentation), while activities that have no reference to an underlying function are missing their spatial implementation. In this way, I am using ABS as a design tool for *constraining functions through processes (and vice versa)*.

**Authorship Information.** The article and all mentioned implementations were written by myself and sums up my dissertation “Prozessvisualisierung in der Krankenhausplanung”\(^76\).

**Main Contributions.**

- Remove ambiguities in the notion of “function” through redefinition as “capability” of a space, which acts as a direct counterpart of an activity in a process (see p. 201 of the article). Provides a mapping from capabilities to functions, so that designers can continue to work using that term (ibid.).

- Extends the schema into an ABS with superimposed business processes, in which a consistency check of unused or missing functions can be made (see Sections 3.3 and 3.4 of the article).

**Type of Work.** Peer-reviewed conference article, based on the extension of an earlier conference paper\(^77\) for which I received the Ivan Petrovic Award\(^78\). Numerous accompanying implementations.


\(^78\) http://www.ecaade.org/conference/ivan
Schematic Systems – Constraining Functions Through Processes (and Vice Versa)
Gabriel Wurzer
Schematic Systems – Constraining Functions Through Processes (and Vice Versa)
Gabriel Wurzer

ABSTRACT
We propose a novel computer-based design method for planning process-driven buildings which extends the traditional architectural schema to include processes. Each function in the schema can be tied to a process, giving us the ability to find (1.) functions that are not present in any process (2.) processes that lack some of their required functions. As benefit of our approach, we can keep functional program and process models of the building consistent and help bridge the communication gap between process planners and architects, simulating the entered processes as we go along.
1. Introduction

The planning of complex buildings such as hospitals, airports and industrial facilities is process-driven: Above all, these buildings need to support their users in their daily work routines (i.e. processes), which are planned in parallel to the design and evolve with it over the course of the building project. In this context, the architect has to define a functional program that facilitates these work routines, in close cooperation with the process planners who define the process model of the organization in question. However, through the co-evolutionary nature of process model and functional program, two consistency problems arise that must be considered by both architect and process planner:

- Functions that are present in the functional program but are not used in any process are likely to be unnecessary or left over from previous iterations, lest the architect can give good reasons why there is indeed need for them with reference to some other argument than the process (restrooms, to give a trivial example, will not need to be rigorously justified).
- Steps in the planned work routine that are not linked to a function can be considered to be incompletely specified, as they remain abstract and without reference to “where” they will be carried out. There are two reasons why this might happen: First, the process planner might simply have forgotten to name the corresponding function. Second, the process planner might have named a function, but the functional program has changed, yielding a reference to a now non-existing function.

Keeping functional program and process model consistent throughout the project is the key problem we want to address in our work, as there is to date no integrated planning tool that links processes to a design, or vice versa. More precisely, we wish to contribute a novel kind of computer-based planning method called schematic systems that:

- Provides a combined diagram for processes and functions, as extension of the classical architectural schema (see ‘Extending the architectural schema’, beginning on page 202).
- Can find functions not present in any process and processes that lack functions (see ‘Executing Constraints’ on page 206).
- Can optionally be used to simulate the entered processes, helping both process planners and architects to get an overview of the planned operation of the building (see ‘Optional Extensions’ on page 207).

Our work complements other approaches that are focused on recording activities (see ‘Related Work’, page 200), but it is distinctively different: To the best of our knowledge, no previous work specifically considers using processes as constraints for the specification of functions. Furthermore, no
other approach explicitly allows for process modeling and simulation directly in the schema. In combination, these innovations lead to a very efficient and simple basis for communication between architects and process planners.

We have implemented our approach as software (see page 208) and are now evaluating it in actual planning work in the hospital domain (see page 210). Our insights indicate that the approach is well-suited for the early stages of conception of a process-driven building and for cases where a redesign in later stages is necessary. Furthermore, the addition of process simulation directly inside the schematic diagram has gained wide-ranged approval from the side of our project partners in the hospital domain.

2. Related work

Albeit being visually based on classical architectural schemas (described e.g. in [1]), our approach clearly relates to the ‘Activity Data Method’ developed in the 1960ies [2], in which activities are gathered as building user requirements. More recent work in recording activities as part of generic building modeling [3] directly in CAD systems has been given in [4,5]; however, process modeling only appears as future work item. Other work on office building user simulation [6] has a concept of processes; however, the main goal lies not in the simulation of planned processes but in replaying recorded behavior for the sake of understanding and predicting space utilisation in an office building environment.

Having hierarchical design methods in mind, our approach also relates to CASA [7], SEED [8] and KAAD [9]: The objectives are given in the form of processes, the design options are given as the arrangement of functions and subsystems planned by a body of collaborating professionals (architects, process planners). A difference which we clearly see here is that the named approaches clearly focus on the generation of alternative design solutions and their subsequent evaluation, which is not our goal. We try to achieve a rigorous design by justifying the required functions by processes.

As side-issue, we wish to note that the use of planning tools based on tree hierarchies for large structures is heavily debated [10]: A systems planning approach would imply that all needed resources should be available in the system as a whole. However, since we know that inhabitants or building users think in neighborhoods [11], it is important to allocate functions according to their local perceptibility rather than efficiency. This effect also applies at smaller scales. For example, toilets can never be planned efficiently; rather, one has to ensure immediate availability throughout the building.

Another interesting area well beyond the scope of this paper is the representation of buildings and activities on the data level, taking into account the ongoing efforts for standardization in the Building Information Modeling (BIM) domain. In brief, we can say that there are ways of representing activities on this level; however, the tools that would use this
data to support architectural programming are still missing. We may forward the interested reader to [12,13].

As visual representation of business processes, we have chosen a simple flowchart-like notation that could eventually be extended into the more generic Business Process Modeling Notation, which is described in detail in [14].

3. Elaboration

3.1. Re(de)fining the term ‘function’

The notion of function is ambiguous [15,16]: Architects use the term more in the sense of purpose (or intent), while an outside observer would derive ‘function’ from an action that was actually performed. It is hard, if not impossible, to build up a rigorous basis for the statement of functions in the presence of such dualities. We therefore have to introduce two new termini in order to make difference between these two notions clear (also see left image in Figure 1):

- **Capability**: the ability to perform a certain action in a space (e.g. capability can sit and can park bed in a hospital’s waiting area).
- **Action**: the implementation of a capability in space and time (e.g. action wait in the same waiting area that either uses can sit for patients that can walk or can park bed for reclined patients).

Extending on this point, we argue that architects are in fact planning the capabilities for their spaces, while process designers plan a sequence of actions (also called process), which uses these capabilities.

The big difference between capability and function lies in the underlying semantics, as is shown with reference to the right part of Figure 1: A function represents a set of capabilities. The function of waiting stands for can sit and can park bed. However, a function of hygienic bed processing consisting of can park bed and can clean bed also has the very same capability park bed. Functions are therefore not disjointed, they might share some (or even all) of their capabilities. In contrast, capabilities are not further decomposable. They can be directly used in an action, as they refer to the intended work routine that underlies the function. This clarification will be vital when binding to functions by processes, or as we would now say: actions to capabilities.

3.2. Processes as seen by architects and process planners

Architects and process planners have radically different ways of expressing work routines of future building users:
• Circulation paths (Figure 2, left) are used by architects to describe a work routine within the (preliminary) design.
• Business processes (Figure 2, right) are used by process planners to describe processes in an abstract way, without needing a reference to the actual space where the action is to take place.

Either representation has its advantages and disadvantages: Circulation paths offer an intuitive depiction of trajectories; however, they do not state the actions out of which the process is composed. This is the domain of the business process, which names the sequence of actions that need to be carried out. However, unlike a depiction of paths on a design, business processes offer no control over the ‘level of detail’ or scale of each action, as process planners often mix highly detailed descriptions of process steps (“ask patient to wait”, “call up patient”) with very coarse ones (“perform operation”) in the same process.

We argue that there is no need for two separate views on processes. In fact, the disadvantages of the one representation may be alleviated by the advantages of the other representation, as will be shown in the next subsection.

3.3. Extending the architectural schema into a schematic system

Creating a common view of processes in the architectural schema and using these to constrain function requires several extensions, which we are about to describe now in full detail. However, before starting off, we wish to briefly recapitulate what a schema is and for what it stands for, in order to be complete.

A schema (also see left in Figure 3) is a diagram that gives the rough placement of spaces and circulation of a possibly preliminary design. A space is a rectangle which stands for one or more functions that have been grouped (this is also often referred to as zoning). The rectangular form is in no way not meant to be taken literally, as the final form is to be determined separately in a phase called form finding. Circulation axes (depicted as lines with arrows) are furthermore used to give an idea of the accessibility of...
As first extension of the schema, we now wish to transform functions to capabilities (refer to right part of Figure 3): Each function is substituted with its capabilities, either manually (by thinking about probable actions that are conductible in the context of the function) or automatically (with reference to a database that contains mappings from each function to the corresponding capabilities). Then, we introduce a process as ordered sequence of actions in a space, where each action is bound to at least one capability (see Figure 4). Actions are spatial – meaning that they have a concrete location. In contrast to this, capabilities are non-spatial – they are merely enumerated by their space and wait to be used by a process.

In the next extension step, we wish to take a deeper look at the hierarchies found in processes and transfer these into the schema. The constraint that each process must be contained in a space (i.e. actions may not cross space boundaries) is primarily targeted at achieving a common “level of detail” among actions in the process, as is the case with circulation paths (refer again to Figure 2, left): This type of process representation gives a common scale to all contained trajectories, as they are laid over a design, which is available in only one resolution. As additional thought, business processes have the ability to specify actions that are themselves composed of several sub-actions. When dealing with a process on a high level, all
sub-actions remain hidden, giving a hierarchy of processes within processes. We want to apply the same concepts to spaces, in the following manner (refer to Figure 5, left):

- Spaces can themselves contain sub-spaces. Each sub-space must be fully contained in the boundaries of its parent space, building up a spatial hierarchy in the form of a tree.
- The capabilities of a sub-space are inherited to the parent space (i.e. if I can operate in the operation theatre, I also have the capability to operate in the hospital). However, an action that references an inherited capability may only do so when located inside the volume of the capability’s defining space (i.e. even though I can say that I can operate in the hospital, I will need to go to the operation theatre in order to perform this activity).
- Every process generates a capability by the same name in its space (i.e. the process perform operation in operation area generates a capability can perform operation). This process-generated capability is inherited to the parent space just like any other capability.

To build a process that spans multiple spaces, one goes to the parent space and creates a process that uses the inherited capabilities of the sub-spaces. As example of this, consider the process perform operation which was defined in the operation area (Figure 4). The task is now to extend this process into an emergency operation process, as seen from a global view. Going one step upwards in the hierarchy, we come to the hospital space, which has inherited all capabilities from the spaces operation area and emergency ward (Figure 5, left). Among these are the capabilities can examine and can
perform operation, which are used in the right image of Figure 5 to actually build the new emergency operation process. Note that the usage of the inherited capabilities was restricted to the area of their defining spaces, given as dashed boundary.

Using the latter example, we have shown how a process can be composed of sub-processes. The addition of a spatial hierarchy furthermore leaves each sub-process in a defined scale, as all actions are on a common ‘level of detail’ that is defined by the containing space. As a side-note, the introduced hierarchy can also be used to effectively hide processes in subspaces, thus easing the visual load for the planner.

We have so far introduced processes in the form of actions (i.e. distinct points in the schema). What is still missing is the depiction of trajectories, which are obtainable automatically by using information about circulation axes and thresholds:

- Between each pair of consecutive actions, a directed line is drawn (left in Figure 6).
- If the line crosses boundaries of spaces, then it needs to be refined. We then have to consider all hit boundaries in a sequential manner, beginning with the one that lies nearest to the first action. If the boundary under consideration has thresholds, the line is split into segments so that it can pass over the nearest one (middle in Figure 6).
- After passing over the thresholds, we refine the line a second time. We consider each segment of the line, starting with the one that begins at the first action. If the segment under consideration crosses a circulation axis, the segment is again broken up into sub-segments, in order to be able to dock orthogonally at the circulation axis and to walk over it (right in Figure 6).

The proposed algorithm gives a unified view of business processes (actions) and circulation paths (line segments in between) in the changed architectural schema, which we wish to call schematic system, in order to distinguish it from the original diagram. The term system refers to the introduced hierarchy of spaces and processes, whereas the term schematic stands for the origins of our method.
3.4. Executing constraints and depicting warnings

Using the presented diversifications of functions and processes as well as the extensions of the architectural schema, the task of checking process model and functional model for consistency becomes trivial. If we are to find all functions without process and all processes that lack some of their required functions we need to:

- Check if all capabilities have an action that uses them: If we find a capability where this is not the case, this capability is unnecessary or must be justified by other arguments than its usage in a process. As each function is a set of capabilities, we can extend the argument further and say that a function which consists of a high number of unused capabilities indicates that it will remain underutilized. A function consisting of only unused capabilities is altogether unnecessary.

- Check if all actions refer to a capability: An action that does not have a reference to a capability remains abstract, which is not desirable – the question of “where” and “based on which function” an action will be executed must be answerable at all times. Either was the reference forgotten, or the architect has changed (or removed) the function, in which case the process is left empty-handed.

All unreferenced actions and capabilities found in the consistency check are to be brought to the attention of the planner as warning symbol (see left and middle image in Figure 7). In our preliminary implementation, warnings can be acknowledged and switched off, giving some reason as to why the function or capability does not have a reference. Furthermore, in processes, there is a type of action called condition that does not necessarily need to have a reference to a capability, since it represent a decision between two or more alternative paths in the execution and is genuinely abstract by its nature.

Besides using warning symbols, we can also use icons for actions (see Figure 7, right). This can help decrease visual load on the planner’s side and is in our view preferable to using text (especially in domains that have a standardized iconic language – think e.g. of airports). Furthermore, symbols that are normally used in flowcharting can also be utilized, which would make for a more “business process”-like look of actions.
3.5. Optional extensions of the proposed method

The nature of the circulative axes found in a schematic diagram is that they are always in the floor plane. It would, however, be nice to also have vertical circulation in the diagram (see Figure 8, left). However, since the 2D schema is unfit for depicting such a vertical circulation, we have extended schematic systems also to three dimensions. The right part of Figure 8 gives an example of such an extension, both for spaces as well as a process.

As further extension, we have looked at the introduction of process simulation directly in the diagram. The reasons for this are manifold:

- It is often the case that the usage of a process-driven building is non-equal over the run of the day. Therefore, there can be different ‘hot spots’ for processes at different times, which can only be visualized when resorting to animation.
- Process simulation may also give a hint at space requirements for areas that are directly dependent on the process (e.g. waiting areas in hospitals are directly dependent on expected queues before examination spaces).

Developing a process simulation involves defining a scheduler (see Figure 9, left) which “starts” the processes at given times, thereby putting an agent on the first action. We have looked at two possible start conditions: Time-triggered (e.g. ‘when the time has reached 1 minute’) and event-triggered (e.g. ‘when a process executes the action acute operation is needed’). Each generated agent then runs along the sequence of actions, performing additional task as he reaches an action (e.g. for junctions: decide on which action to take next). In our implementation, this behavior is momentarily given as programming code. However, there are other approaches that offer a more graphical and user-friendly interface for this task [17], which we would like to adopt in the future.

As another topic for an optional extension, we have looked at automatic categorization of circulation axes, using Bill Hillier’s Space Syntax method [18,19] (see Figure 9, right). In essence, the method can give a categorization...
of circulation from a reachability point of view. It takes a set of lines (the circulation) as input, computes a graph from it by taking the center point of the line as graph node, and connecting this to the center points of all the lines it touches (justified graph, also see middle in Figure 11). The process continues with a shortest path analysis on the graph, where the sum of all hops from one center point to every other center point in the graph is computed. The result is a number per center point, this is normalized and gives us a hint at the prominence of each circulation axis. The right part of Figure 9 shows circulations in a schematic system, scaled by their reachability (i.e. the thick circulations are ‘main circulations’ and so on).

4. Prototype, extensions and re-implementation

Our work has been initially implemented as a prototype 3D application (see left in Figure 10) which offers two distinct modes, a systems view and a process view:

- In systems view, schematic systems can be edited and attributed with capabilities. As systems are hierarchical, we have chosen to let the program focus on only one system (the active system) which we shown as a spatial container for its immediate sub-systems. The spatial hierarchy can be navigated either directly by double-clicking a sub-system (jump to subsystem) or double-clicking the active system (jump to parent), or by using a browser-like address-bar at the top of the application.
- In process view, the user can edit processes consisting of one or many actions. As each action is placed, a dialog (see right top in Figure 10) offers the opportunity to link the action to a capability.

A sanity check dialog (right bottom in Figure 10) furthermore tests the model for consistency, as it can find all capabilities that are not used by an action and vice versa. The user may react by removing the questionable actionable capability or action, or by affirming it as needed.
We have initially tested our tool using the case of a co-working scenario in a high-rise office building [20] as test bed. The task was to model a cross-corporate desk sharing environment with dynamic delivery and resupply processes on top of the three-dimensional schema of the building. As key users, we chose to involve three architects from inside our institute and one external process planner, as our emphasis was on gathering feedback rather than prematurely moving out into the field with the approach.

From the initial modeling runs, we observed that the architectonical schema with superimposed processes served both parties (architects/process planner) well as common communication media. Editing was conducted in a straightforward manner, even though the 3D interface was generally seen as cumbersome and non-intuitive. Changes on either side were accurately discovered and dealt with. A strict separation concerning who may edit what (i.e. architects edit the schema, process planners do the processes) was, however, questioned by both parties. Omitted functions and unreferenced actions were discovered in the course of the scenario, however, through the rather limited nature of the test project, no real revelations pointing to planning errors were observed. From a visual standpoint, the cluttering of the screen with actions posed severe problems. Therefore, we extended the program so as to include the possibility to selectively display processes either through switching process visibility on/off or by filtering, either of which was positively acknowledged. A further point of critique was the lacking integration of the application into other programs (CAD/process modeling package), which our users saw as essential for further re-use of the data in subsequent phases or projects.

After doing the prototype, several additional programs were written in order to explore possible solutions to shortcomings that our initial tests...
had shown, or to supplement the method with more features that would truly enhance the planning of complex buildings:

- The implementation of a process simulation (see Figure 11, left) inside the schema was considered to be a natural extension by all our test users. Our architects said they would benefit from being able to see the temporal and causal usage of their functions while our process planner said that it would make sense for him to look into where a process was being utilized. As further benefit, the simulated usage of functions can reveal unobvious errors in the process model (example: functions that seem to be perfectly utilized by processes can be unnecessary if these processes are never triggered).

- Implementing the process simulation also required including automatic computation of trajectories between each pair of actions (shown in Figure 11, left), in order to send simulated persons along the circulation instead of walking straight from A to B, regardless of spatial boundaries.

- An implementation of space syntax (see Figure 11, middle) further provided us with a method of showing the prominence of each circulation axis. Although our test users were quite neutral towards this feature, we have nevertheless included it into the method as optional part, as it might turn out beneficial in the face of more complex buildings.

- Furthermore, we have also explored (but not included into the method) the use of pedestrian dynamics [21] (see Figure 11, right), so that (1.) agents will avoid bumping into each other and (2.) we can use the pedestrian flow data to visualize throughput for every trajectory in between two actions.

The sum of all findings from the initial tests and subsequent implementation of ‘explorative extensions’ has lead to the reimplementation of the prototype as 2D software (see Figure 12), which we now employ in a real hospital planning project in Upper Austria (LKH Vöcklabruck, 586 Beds). More specifically, it is our task to assist in the introduction of a novel triage system using our tool, in close cooperation with...
with the hospital employees and administration. For the hospital, the most important points why a use of our approach seemed interesting were:

- The use of simulation as a means of objective comparison of different usage scenarios.
- Visualization of processes and schematic systems as means to communicate with building users, architects and process planners in an easily-understandable way.

Apart from having a design case in the hospital Vöcklabruck, we will continue evaluating and improving our approach, using our initial test candidates for feedback. After the actual design case is finished, we furthermore plan to roll out the software to a wider community (especially from the field of process planning).

5. Conclusions and future work

We have shown a novel method for planning process-driven buildings which binds functions to processes in order to avoid (1.) functions that are not present in any process and (2.) processes that lack required functions. Our approach is based on the architectural schema and extends it to incorporate processes that can also be simulated. Bringing together processes and functions in a common diagram is the main benefit of our approach, as this reduces communication overhead between architects and process planners and eases presentation of the processes. Furthermore, our approach has the ability to keep functional model and process model consistent, which is of utmost importance in complex project situations we wish to address.

We have implemented our approach twice, first as 3D prototype and then as more refined 2D tool which is currently being employed in actual project work. As future extensions of the tool, we want to introduce a database for the mapping of functions to capabilities (as architects are
generally not willing to give up their notion of function). This database should be targeted at different areas of building design such as hospital planning, factory planning and airports. Second, we plan to integrate our tool into modeling packages (CAD / process modeling), in order to further support the workflow found in the planning of process-driven buildings.

Acknowledgements

Thanks to: Jan Michl (for diversifying functions), Georg Franck and Christa Illera (supervision), Sigrun Swoboda, Andreas Jonas, Kamyar Tavoussi (for their work) and Michael Bacher (for his hospital expertise).

References


10. Minett, J., If the City is not a Tree, nor is it a System, Planning Outlook New Series, 1975, 16, 4–18.


Gabriel Wurzer
Vienna University of Technology
Institute of Architectural Sciences
Treitlstrasse 3, 1st Floor (259.1), 1040 Vienna
wurzer@iemar.tuwien.ac.at
**Paper: Reverse Engineering Hospital Processes Out of Visited Nodes**

This paper looks into the simulation of treatment chains, i.e. sequences which patients visit during their treatment. In contrast to now-common discrete simulation models describing processes through a sequence of nodes and connecting (directed) edges, we use only nodes in which we put dispatching logic, since (quote) “patients freely cross from one node into another, depending on individual treatment”. The reason for this is twofold: (1.) We are after modelling actual patient pathways, which are fundamentally different from prescriptive “process descriptions” that offer an idealized version (standard process) but leave details on the actual occurrence of activities. (2.) Modelling each individual treatment chain as a standard process is impractical, not solely because of the complexities in enumerating all possible variations of activity sequences but also because of the visual clutter that would occur when being overlaid on a schema. We thus propose a visualization where “often-occurring” treatment chains can be found and reverse-engineered into standard processes, thereby establishing the simulation as a process design tool.

**Authorship Information.** The paper was largely written by Barbara Glock and myself, with inputs from Felix Breitenecker and Niki Popper. The preliminary implementation of the dispatcher editor was written by myself.  

**Main Contributions.**

- Argues for the use of a process simulation model consisting of nodes and contained dispatchers containing transitional logic (see Section III.A in the paper).

- Identifies and visualizes often-occurring node sequences as a pre-step for establishing “standard processes” (see from Section III.C of the paper onwards).

**Type of Work.** Peer-reviewed conference paper, accompanying preliminary implementation of a graphical programming language for dispatchers.

Reverse Engineering Hospital Processes Out of Visited Nodes

Barbara Glock\textsuperscript{1}, Gabriel Wurzer\textsuperscript{2}, Felix Breitenecker\textsuperscript{1}
\textsuperscript{1}Institute for Analysis and Scientific Computing
\textsuperscript{2}Institute of Architectural Sciences
Vienna University of Technology
Vienna, Austria
barbara.glock@student.tuwien.ac.at
gabriel.wurzer@tuwien.ac.at

Abstract – During hospital planning, "flow-chart"-like depictions of processes (nodes and edges) are often overlaid over the floor plan of a clinic, in order to model patient pathways. However, such static definitions may not be accurate, as patients freely cross from one node into another, depending on individual treatment. Modelling each possible sequence of nodes is neither practical nor intelligible. This means graphical complexity due to overlapping edges. In our work, we thus present an edgeless process model in which each node acts as "dispatcher": It carries the logic for choosing a next node, thus enabling us to omit edges and introduce dynamic distribution of simulated patients. "Processes" (i.e. often-perceived sequences of activities) can nevertheless be inferred in an a posteriori step with an algorithm that will be presented, utilizing the actual patient pathways as indicator. Furthermore, the visualization of how processes are generated out of activity chains by reverse engineering will be discussed too.

Keywords - hospital planning; process simulation; treatment chains;

I. INTRODUCTION

Processes act as fundamental basis and constraint for the planning of a clinic: Due to the complexity of the work processes that the building has to support in a 24/7 fashion, it is not possible to plan such a building without knowing all occurring activities and activity-holders in advance. Modelling of processes is usually done in graph-like fashion, i.e. we see the activities as nodes and their temporal and causal ordering as edges. However, having to draw edges might be suboptimal from a modeling point of view, as is shown in the example depicted in Figure 1(a). Assume that we have a standard process for an emergency department, in which a patient either enters via the main entrance (walking or sitting patient) or is brought in by an ambulance (lying or sitting patient). If we have to draw the process on top of the clinic’s floor plan, already some visual cluttering occurs. However, and more importantly, the process drawn does not reflect the chain of activities that are carried through in the course of treatment accurately; for example, some patients are immediately treated without undergoing triage (when arriving with an ambulance), others go through the registration process and eventually end up in the treatment area twice, with an x-ray in between. Figure 1(b) shows the real decomposition of the patient volume into the different activities encountered, which cannot be easily modeled in the process fashion seen in Figure 1(a).

![Figure 1: (a) Example of an emergency department process modeled as spatial flowchart. (b) Activity decomposition for the whole patient volume.](image-url)
However, as stated before, this would lead to visual cluttering and is furthermore not practical (too much work for too little benefit). In fact, why model processes statically at all, if a dynamic sequence of activities is to be reflected?

The idea to give up the concept of static edges connecting nodes lies at the core of what we want to present in this paper. More specifically,

- we propose an approach in which nodes act as dispatching units, which enables us to model dynamic chains of activities in a comfortable manner (see Section III.A),
- we give up the notions of edges in favor of an alternative way with which agents are moved between the nodes: A dispatching algorithm written by the user acts within the nodes, which in turn moves the agents through the floor plan (see Section III.B),
- we reverse-engineer “processes” from the occurring activity chains, taking into account the utilization of each activity chain during the simulation. As result, we are able to depict the activity chains that occur most frequently (see Section III.C).

To sum up, we can eliminate unnecessary clutter, when depicting processes visually and ensure that activity chains of patients can be modeled accurately and efficiently. The mining of often occurring processes in these activity chains is yet another contribution which we bring forward in our discussion of this subject and concentrate on our main idea, the simplification of process descriptions and subsequent simulation within these.

III. ELABORATION

In the following sections we will define what we exactly mean by a process model consisting only of dispatching nodes and tokens that flow between them. In the first part, we will outline the structure of our model and define the basic terminology that will be used throughout the paper. We then go on to elaborate how a process simulation can execute the model, in order to obtain a flow (utilization of each node and sequence of nodes visited). In the last part, we analyze this information to infer “processes”, i.e. often-recurring activity chains.

A. Basic Process Simulation Model

Our process model is composed of the following two basic constituents:

- The process token, which represents an instance of an executed process.
- The nodes, which are visited by the process tokens and act as dispatcher, in the following fashion: A token that enters a node executes code contained therein, resulting in a next node to visit. This node is set as next goal for the token (transition). There are two special cases of nodes, in the form of start- and end-nodes. Start-nodes generate a process token, i.e. they cannot be set as goals. End-nodes terminate a token, which means removing the process instance from the simulation. There is also the case of process composition, in which a node passes the token on to another process being called on its behalf. In that case, the start-node of the called process becomes the entry point and the end-nodes the exit point for the sub-process invocation.

A token can be visualized as an agent, when in the floor plan. The transitions between nodes can be animated, yielding a “patient-flow”-like depiction intended for end-users (architects, planners).

B. Simulation within a Node

A token having arrived in a node executes the code contained therein, in order to arrive at the conclusion what next node to take. In our case, this “code” is again graph-based, which allows us to utilize almost the same methods for spatial and non-spatial simulation:

- A program has one entry point (start-node) and many exit points (end-nodes).
- Between start- and end-nodes come a variety of intermediate nodes that read data contained in the token, perform calculations on that data and finally decide what next goal to select. The description of this programming language is deferred until later, for now we just wish to express that the “code” is yet another network through which the token needs to travel.

II. BACKGROUND ON PROCESS MODELING

Process modelling typically includes “flowchart-like” process descriptions showing activities as boxes connected by arrows. Several flavors of activities and connectors exist, both from the representation as well as the semantic side: For example, flowcharts are standardized under ISO 5807:1985, however, there is the more recent Business Process Model Notation (BPMN) that has gained widespread usage in computer science and business administration. Apart from notation, some models explicitly aim at semantic aspects of these models – such as Event-driven Process Chains (EPCs) or Activity Diagrams within the Unified Modeling Language (UML). Simulating such process definitions uses Graph-Oriented Programming (GOP, see [1]) in conjunction with a discrete simulation [2] that handles aspects such as resource allocation, queuing and so on. From a more mathematical side, these approaches can also be seen as Petri Nets [3], in which a process token representing a process instance moves along a graph of nodes representing transitions. The latter aspect is a difference to the aforementioned notations, in which nodes stand for the states and edges for the transitions. Even though Petri Nets offer the advantage of being formally analyzable, they are hardly used in the context of business administration and building planning, which is our context. We thus omit the discussion of this subject and concentrate on our main idea, the simplification of process descriptions and subsequent simulation within these.
• The token itself becomes a data container, similar to what programming languages call context: For example, a node executing “c=a+b” would reach into the token to retrieve the values for a and b stored there, storing the result in a new data entry c.

Figure 2 shows an example of the whole model, both outside and inside nodes: Initially, we start the process on the start-node arrive1 (Figure 2: a), which generates the process token and executes the contained code graph (Figure 2: b)). Typically, in this step, we will read the intended activity chain for that process instance from a spread sheet and store that within the token (Figure 2: b) in goals). Let us assume that this would be register > triage > treat > exit, for the sake of the example. The code would set the goal of the token to the first entry, in this case register. The token (Figure 2: c)), is then animated spatially as it transitions to the registration node register. Upon entering that node (Figure 2: d)), a waiting time of three minutes is introduced by the contained code (Figure 2: e)). Behind the scenes, the token is moved into a passivated list, in which it stays for the given duration (Figure 2: f)). A scheduler has to be implemented such that it can activate processes again, also using events as criterion (to be discussed in due course). As next step, the goal, and therefore the token, is advanced to triage (Figure 2: g)). The contained code in that node (Figure 2: h)), assigns one of three priorities randomly to the token. It examines the next node, which is treat. Each node on the floor plan is implemented as a server of limited capacity (treat: three places, of which none is currently available). triage determines that instead of advancing to the next goal, the token has to go to wait1, until treat becomes free. The token walks over to wait1 (Figure 2: i)) and becomes passivated waiting for treat (Figure 2: j) → f)). As has been mentioned before, the scheduler has to reactivate that process token, when treat’s capacity is below its maximum. Technically, treat broadcasts a “freed event” which the scheduler utilizes to that end. Once the token has entered treat (Figure 2: k)) and waited for the treatment time (Figure 2: l)) again by being sent to the passivated list, it advances to the end-node exit (Figure 2: m)), where the process ends.

It is crucial to mention that several dynamic actions can happen within a node. For example, treat could have determined that additional intermediate steps (x-ray > treat) would be necessary, based on some probabilistic criterion. Vice versa, some elements of the process chain could be removed, in case the treatment concludes that those steps are obsolete. This could be again the case for probabilistic model.

As a next step, we describe the inner mechanics of the simulation core from an implementation standpoint. We have two parts that are combined: First, the code that is executed within nodes (which we call code graph for reasons we will explain in due course), second, the algorithm that takes care of transitioning between nodes. In both cases, it is the scheduler that handles all of these types of executions and, furthermore, is responsible for passivation/activation (based on time and events).

Figure 2: The activity chain of a token through the floor plan.

1) Code Graph

In contrast to a variety of programs that let the user write textual code, we have a graphical editor for the same sake (see Figure 3). Even though this does not free us of the programming language itself, it does provide the benefit of a “drag and drop” development of the dispatching code, in which an instruction can only be dropped if it can be accommodated by the target block. Such an avoidance of errors is significant for our intended audience, which consists of hospital planners (i.e., architects, process designers) and not programmers. It is based on a similar approach by the CMU’s Alice Team [4].

Figure 3: Dispatcher code graph, visualized as hierarchy of “blocks within blocks”
Interestingly, a block-based program forms a hierarchy, which can also be interpreted as graph. For example, the program in Figure 3 is analogous to the graph shown under Figure 2h. We use a graph engine acting on that level to actually execute us the code. Internally, each token carries an execution pointer to the current node of this code graph. The token itself furthermore contains the necessary data (values the computation in the nodes need). The scheduler has the task of advancing the execution pointer, based on the result of the current node. For example, the expression “if free? treat is true then” can return either true or false, which is also the name of the transition in the code graph used for getting to the node of the code.

The overall responsibility for advancing the execution pointer lies with the scheduler. More precisely, this manages all tokens and their respective state (data, execution pointer, execution history). It encompasses two waiting lists - one for tokens waiting for a timestamp, one for tokens waiting for an event to occur. Once the execution arrives in a node that requires passivation (based on one of the two the mentioned criteria), the token is moved to these lists, to be activated in the future. In our model, we use a fixed-step progression of one second, plus events that are fired when resources become free, nodes are entered and left and so on; however, this is no necessity - we could have also used a scheduler with a variable time step, i.e. future event list for time, plus an additional one for events. Since events can only be generated by tokens that wait for a certain time, we could have first activated time-based tokens, then event-based tokens until all signals are consumed. However, we have not done so, because we want to be able to animate the passage from one node in the floor plan to another - hence we progress in seconds.

2) Transitioning between Nodes of the Floor Plan

The transitioning algorithm is, in the simplest case, a linear interpolation between the source and target node of a token. However, at this level, we might also explicitly deal with the floor plan and physical movement of each token (shown as an agent), in the following fashion:

- Using the layout of the hospital exemplified in Figure 1, we can perform a physical simulation [5] that simulates each individual’s passage through the floor plan for high-density situations. To a certain extent, this approach is useful for simulating crowded waiting areas, entry zones and the like. It must be mentioned, however, that these algorithms are more targeted at egress planning than early hospital design, which is our context.

- Even if we do not model the physical transition of each token through the floor plan, we can make it walk along the circulation (basically a graph giving walkable pathways). An approach that automatically calculates such a transition sequence along a circulation graph (also taking walls and entrances that lie along the way into account) has previously been presented in [6].

C. Reverse Engineering Hospital Processes out of visited Nodes

After having simulated the patient pathways in the described manner, we want to reverse-engineer processes out of the occurring activity chains.

Simply assigning a utilization to each node would not serve the purpose, because only the capacity of each node would be displayed (by a counter that increases by one for each token (agent) passing by). Our goal is to display the whole processes and not only the visited nodes. Therefore, agent histories are used. An agent history is a string storing each of the patient visited nodes. Those histories are then (mentally) laid on top of each other to get a look on the occurring pathways.

An algorithm is used to extract the “real” processes out of these pathways. To get a clearer look what is meant by “real” processes, an example is discussed shortly. The activity chain (each visited node) of an agent with for example id=2 stored in the agent’s history is: arrive1 > register > triage > wait1 > treat > x-ray > treat > exit. The agent with the id=3 has an activity chain such as: arrive1 > register > triage > treat > x-ray > treat > exit. The difference of these two activity chains lies in one waiting node: the agent with id=3 advances from triage directly to treat and does not has to wait. Due to the fact that those two activity chains have different strings stored in the agent’s history, they would further count as different processes, although their (for the modeler) important pathway is the same.

Therefore, the nodes are distinguished between significant and insignificant nodes. The latter one is a node representing waiting in a waiting area, because no significant activity is done here. Usually waiting areas in hospitals are not from particular importance to the modeler due to the fact that they exist in every hospital. In Figure 2 these insignificant nodes would be wait1 and wait2. Furthermore, there are some nodes for which the user can decide himself if they are significant or not, like triage. On the one hand triage is a waiting node too, but on the other hand it requires additional items (such as staff that is triaging the patient). The modeler can decide of its importance and therefore of its significance.

Furthermore, the user has to define a threshold that determines the amount of patients having used this pathway, used for qualifying the pathway as a being a “process” or not. That can be a real number of total patients, or a percentage of patients. This is important, because otherwise each (for insignificant nodes reduced) activity chain would qualify as a process. An activity chain that represents a specific rare behavior would then be a process too, i.e. the very rare case of a patient suffering from ruptured appendix after being treated and therefore not ending his pathway in an exit, but in being hospitalized.

After having set the threshold and distinguished between the significant and insignificant nodes, the insignificant nodes are eliminated by the algorithm from the agent history. The remaining activity chains now qualify for a “real” process if their utilization is above the threshold. To determine the cardinality for each group of remaining
activity chains, they need to be sorted from longest to smallest by length and also by content, so that they can be grouped by the equal ones. Now it is easy to count the amount of grouped activity chains (sequentially executing the list of sorted activity chains and increasing a counter) and assigning each group of chains its cardinality. Those which are above the threshold are the required processes.

Next the visualization of these processes is discussed shortly: To avoid cluttering these processes have to be laid on top of each other, which makes it difficult to distinguish between processes that are part of other processes together with depicting their cardinality. The pathways of processes are depicted by edges connecting the significant nodes of the activity chain on the floor plan. To display the cardinality of these processes a simple chart of color range is used, for example by depicting often occurring process by a darker color.

The main problem arises if there are processes that are part of other processes. Let’s assume there is process1: arrive1 > register > triage > treat > x-ray > treat > exit, seen in Figure 4 (a) as yellow line, because its cardinality (how often the process occurs) is not that high, and process2: arrive1 > register > triage > treat > exit seen as purple line in Figure 4 (a) having a higher cardinality than that of process1.

The length of those two processes differ (and for the sake of the example also their cardinality), but process2 is (although being an independent process) a part of process1. To lay these two processes on top of each other would lead to not visualizing process2, because it would be totally overlaid. Therefore, we display longer processes (more visited nodes) different from shorter processes (less visited nodes) by adding additional space to the width of displayed activity chain for each additional visited node. In Figure 4 (b) this circumstance can be seen: The process initially gets the width of the start-node and for each additional visited node of the activity chain, the width of the visualized process gets additional space resulting in longer activity chains having a thicker displayed width. Therefore, the yellow line of process 1, inheriting 7 visited nodes appears thicker than that of process2, only inheriting 5 nodes.

Cluttering can be avoided with this kind of visualization of processes, although the cardinality (how often a process is simulated) and every process, no matter if it is part of another process or not, can be displayed accurately.

IV. DISCUSSION

At the core of our approach lies the concept that sequences of treatment steps are dynamic, in contrast to processes, which are static (at least when it comes to the choice of a next node). Characterizing further, our model builds up processes based on the observed data (treatment sequences), while traditional processes are prescriptive and “state what shall be done”. We have argued that, even process definitions in the latter sense are produced during planning, these may fail to capture the “real-world” behavior of patients undergoing treatment, and may thus be less effectively employed in simulation.

Technically, we see our contribution in the dispatcher facility, implemented as a scheduler acting on a “code graph”: Apart from distributing tokens to next nodes (which is trivial), we have closely looked at the mechanisms for supporting passivations/activation within a node. This step represents a challenge, as (in our case: Java) does not support passivating a program and activating it again, lest we turn to more techniques such as one thread per agent (a no-go, considering performance). If we represent the “code” as graph and the execution pointer, we can get around this constraint: The token containing both the agent’s data (i.e. the context, in programming lingo) and the execution pointer (the state) is written into a list (passivation). The scheduler can then activate the program again based on either time-based or event-based criteria, by simply removing the token from the list - in which case the execution resumes exactly where it has left off.

A further contribution lies within the actual reverse-engineering of static processes out of the dynamic activity chains during treatment. On the surface, it would look as though this step could also be done quite easily without our approach, if all treatment steps are known. For example, one might bring up a spreadsheet processor apply the steps listed under Section III.C. However, as has been said earlier, the code can alter the treatment chain as it sees fit - inserting
additional steps or removing them. A typical case for that might be the refurbishment of a hospital, in which the newly planned operational concept differs from the old one: Given the patients of the old hospital and their treatment chains therein, a node code can perform a mapping onto the new operational concept, by inserting, altering or deleting parts of the chain. Such an approach was successfully used by the authors for the case of an 800-bed clinic near Vienna [7], which is also our test-bed for the simulation. Because of the possibility for mapping and other dynamic behavior within nodes, we can safely state that this is not possible lest we resort to an approach as presented herein.

V. CONCLUSIONS

We have presented a model in the context of hospital planning, for situations in which a “flowchart-like” process description is to be simulated on top of a floor plan. Trivially, such depictions suffer from visual cluttering caused by overlapping edges. But, an even more serious issue is that such static process graphs fail to capture the dynamic nature of treatment, in which patients cross individually from node to node, instead following a preset route. To cope with these two problems, our model gives up the notion of edges altogether, replacing them with nodes that can dispatch patients to next nodes dynamically. “Processes”, i.e. static sequences of activities, are derived by a-posteriori analysis of all agent histories. As result, we are able to model treatment chains adequately, while at the same time making it easier for planners to work on the according simulation models (only nodes and dispatcher code is entered).

ACKNOWLEDGMENT

Special thanks to ZIT – The Technology Agency of the City of Vienna, a subsidiary of the Vienna Business Agency for supporting us on the project Modular Dynamic Planning and Simulation for Health Care Facilities (MODYPLAN).

REFERENCES

Paper: From Quantities to Qualities in Early-Stage Hospital Simulation

In the context of early-stage simulation, we often deal with “unsharp” input data such as approximate zone locations within a Schema, from which we compute quantitative results such as path lengths for each agents. However, these are neither correct nor meaningful for the user, who rather thinks in terms of unsharp qualities - e.g. “near” and “far” in a relationship matrix. In this paper, Wolfgang E. Lorenz and I have been thus mapping the quantitative results computed by an ABS back into linguistic terms (i.e. qualities) using fuzzy sets\textsuperscript{80}. As effect, simulation results can be presented “in the language of planners” and compared to preset requirements in that same language.

Authorship Information. Wolfgang E. Lorenz focused on the architectural view of “qualities” and helped diversify the term, while I focused on the actual details of the approach\textsuperscript{81}. The accompanying implementation in Netlogo was written by myself.

Main Contributions.

- Argues that early-stage simulation based on “unsharp” input data must not be interpreted quantitatively, but qualitatively.
- Does so by mapping of quantities to linguistic terms (coined as “qualities”) through fuzzy sets.
- Automatically compares the so-gained qualitative results to the requirements (see Figure 1 of the paper), using a showcase implementation to substantiate its claims (Figure 3 in the paper).

Type of Work. Peer-reviewed paper, accompanying showcase implementation.


\textsuperscript{80} L A Zadeh. Fuzzy sets. Information and Control, 8(3):338–353, 1965

\textsuperscript{81} I estimate that I have authored roughly 70% of the paper
ABSTRACT
Simulation in early stages of hospital planning might not live up to its full potential: Results come as quantitative (crisp) values, whereas early conception specifies requirements in a more qualitative (think: linguistic) fashion. Because of this gap, it is no wonder when planners cannot easily interpret what is computed. In our most recent work, we have addressed this issue through fuzzy analysis: Crisp simulation results are mapped back into linguistic terms, which can then be compared to the requirements set by the planning team. On the one hand, this enables us to communicate “in the right language”, on the other hand, we may use this comparison for automatically raising warnings, in case there is a mismatch between the two.

Keywords: Fuzzy Logic, Hospital Simulation, Early-Stage Design.

1. INTRODUCTION
“Linguistic” descriptions of architectural space are ubiquitous in early stages of planning: Requirements are formulated both narratively and with the help of diagrams showing adjacency relationships, preliminary space layout (“schema”) and so on. Simulation acting at this stage can compute flows of material and building users, resource utilization, space usage and so on, as has previously been shown in Wurzer, Lorenz, and Pferzinger 2012. However, its utility is sort of limited as a design tool, since what is computed is quantitative, but what planners want is qualitative. Consider, for example, the adjacency matrix shown in Fig.1a, especially the relation between “Trauma” and “OT” that is specified as “near”. Let us also assume we let a simulation act in the accompanying space layout, by recording the average path length of agents crossing between these two spaces (resulting in a number). The problem for planners is now how to connect this to the specified requirements (“near”). Which leads us to our

List of Contributions:

- Through fuzzy analysis, we may map quantitative values computed by the simulation back into linguistic terms, which are easily
understood and compared to the requirements (see Section 3.1.1: Mapping Quantitative to Qualitative).

- We may also automate the checking of computed results against the requirements, establishing simulation as a design tool rather than only for analysis and optimization (see Section 3.1.2, “Automated Requirements Checking”).

- There are situations in which there is no 1:1 mapping between computed results and a linguistic term, in which case we may use Fuzzy Inference to take multiple results into account. A discussion of this, together with some background on why such cases can be quite common, is given in Section 3.1.3 (“Mapping Multiple Quantities”).

In the current state of the research, we are applying this approach to an early design of an 800-bed clinic, where we are involved in the pre-planning phase. We are convinced, however, that the ideas we bring forward are a contribution also beyond the borders of hospital design and simulation. For example, buildings which have a comparable degree of formalization - airports, schools, even prisons - necessitate an intensive and constant communication between “planners” and “the client”. Like many other approaches that seek to reduce the complexity of this task, our ultimate goal really lies in the reduction of the communication gap in between the involved disciplines.

2. BACKGROUND

Fuzzy Architectural Spatial Analysis is an upcoming field that describes spatial organization through membership to fuzzy sets - i.e. memberships to linguistic terms in the range [0,1]: For example, a room might be "spacious" (0.8), "average" (0.6) and "small" (0.2) at the same time, albeit to a different degree.

Such fuzzy memberships have been previously used for mapping the perceived qualities of architectural space (Ciftcioglu and Durmisevic 2003), most notably: the formation of spatiality within a given building layout (Arabacioglu 2010). Interestingly, all of the approaches dealing with fuzziness in architecture have so far concentrated on form as primary driver for analysis. In early-stage planning, however, there is no such thing as a definite form; the given schemata and relationship diagrams are rather conceptual: They describe the rules that govern space layout, not the actual geometry that will eventually replace every “rectangle” that stands for a space (see Fig. 1a). This, however, comes in a very later stage called “Form Finding”.

2.1. Related Work

In the architectural domain, Fuzzy Logic (Zadeh 1965) has been widely used for evaluation of the perceived qualities of a given design (De Vries and Steins 2008; Holicky 1999; Durmisevic, Ciftcioglu and Sariyildiz 2001; Durmisevic 2002). However, hardly any work on early-stage applications exists, one notable exception being the fuzzy form generation approach presented Koutamanis (2007). To us, it seems that the obsession with form distracts from early-stage applications, most notable, functional design (often also called conceptual design or pre-design).

2.2. Concepts used in Fuzzy Analysis

Fuzzy analysis is especially good at handling the linguistic terms. In fact, it is one of the only methods that can comfortably cope with their uncertainty, building on expert knowledge entered as membership functions. More formally, these regard the set of crisp input values $X$ that come from simulation - e.g. average distances that agents have to cross between elements of the schema - and map these to a set $A$ of linguistic terms (e.g. distance={"near","neutral","far"}, also see Fig.1b) that are represented by a membership function

$$\mu_{A_i}: X \rightarrow [0,1]$$

(1)

to determine the degree of membership for that linguistic term, e.g. "near" (0.6), see Fig.1b. Trivially, membership 0 means no member and 1 full member of a linguistic term. A full elaboration on the application of this is given in Section 3.1.1, “Mapping Quantitative to Qualitative”.

We can use a threshold $a$ above which regard the linguistic term satisfied (a-cut). In this case, the membership degree can be omitted (see lower part of Fig.1b, in which only “far” survives the a-cut). The so-obtained linguistic term can be compared to a linguistic constraint, e.g. an adjacency relation that was defined between two functional areas such as “near” defined in Fig.1a. A further elaboration of this is given in Section 3.1.2, “Automated Requirements Checking”.

Sometimes, it is not enough to interpret membership functions directly. Cases where multiple linguistic terms are combined via rules to derive a conclusion are comfortably handled via fuzzy inference: For example, a mechanism commonly known as Mamdani’s direct method can consecutively apply a set of rules of the form $IF$ condition $THEN$ conclusion, or, in mathematical terms:

$$Rule_i:$$

$$IF x_i \ is \ term \ in \ A_i \ [ \ AND \ y_i \ is \ term \ in \ B_i \ ] \ OR \ y_i \ is \ term \ in \ B_i \ ]$$

$$THEN \ z_i \ is \ term \ in \ C_i$$

(2)

where $x_i$, $y_i$ and $z_i$ are variables and term is a membership function. $A_i$, $B_i$ and $C_i$ are sets of linguistic terms as described above. Also note that the AND and OR parts are optional and mutually exclusive.

The derivation of the rules proceeds as follows: In the first step, called fuzzification, the membership degree of every condition term $\mu_{A_i}(x_i)$ and $\mu_{B_i}(y_i)$ is computed. The second step then evaluates the IF-THEN-rules using the computed membership degrees:
• Rules consisting of a single condition IF $x_i$ is term in $A_i$ simply evaluate to their membership degree $m_i = \mu_{A_i}(x_i)$.
• Rules containing a logical AND, i.e. IF $x_i$ is term in $A_i$ AND $y_i$ is term in $B_i$, are evaluated to the minimum of the left and right part of the condition, i.e. $m_i = \min(\mu_{A_i}(x_i), \mu_{B_i}(y_i))$.
• Rules containing a logical OR are evaluated to the maximum, i.e. $m_i = \max(\mu_{A_i}(x_i), \mu_{B_i}(y_i))$.

For every rule, the computed membership degree $m_i$ is then used to clip the membership function $\mu_{C_i}(z)$ of the rule's conclusion part. All values above the line given by $y = m_i$ are cut off, giving the new membership function $\mu'_{C_i}(z)$:

$$\mu'_{C_i}(z) = \min\left(m_i, \mu_{C_i}(z)\right) \forall z \in C_i$$

(3)

As third step, all cut membership functions are aggregated by determining their union:

$$\mu_C(z) = \max\left(\mu_{C_1}(z), \mu_{C_2}(z), \ldots, \mu_{C_n}(z)\right)$$

(4)

The last step, called defuzzification, generates a single crisp number out of the aggregated function $\mu_C(z)$ by finding its center of gravity, i.e. the point on the x-axis where a vertical line would slice the aggregate function into two equal areas:

$$\text{COG} = \frac{\int_0^b \mu_A(x)xdx}{\int_0^b \mu_A(x)dx}$$

(5)

3. ELABORATION

Examples of qualitative requirements that are specified in early planning are found in the adjacency matrices or bubble diagrams (White 1986):

• **Adjacency matrices** (see Fig.1a) specify the relations between functional areas. The semantics of this are manyfold, as architects have always used these in a variety of connotations, for example: Spatial proximity (“near”, “neutral”, “far”) or interaction between two units (“high”, “average”, “low”). Observe also that such relations may be either positive or negative, apart from having a degree.

• **Bubble Diagrams** are a variation of adjacency matrices, in which each function becomes a node of a graph and each relation becomes an edge between two nodes. Therefore, what has been said earlier also applies to this type of diagrams.

The question is now, how to take these into account when interpreting the results of a simulation, which (in our case) runs within the architectural schema (also refer to Fig.1a): This diagram shows the preliminary space layout, consisting of an arrangement of spaces. The area that these spaces take up is typically given in the space allocation program (which is yet another form of requirement - albeit quantitative) that comes with the competition or tender documentation.

3.1.1. Mapping Quantitative to Qualitative

We simulate the flow of patients, staff and material through the preliminary hospital schema, computing the utilization of resources (e.g. waiting seats) and spaces (e.g. the whole waiting area). In more detail, patient arrivals and functional areas visited are identified via the Hospital Information System (HIS) of the current hospital, in order to be able to allocate agents within planned new schema of the same hospital (an 800-bed clinic close to Vienna/Austria). In the new schema, every function can be limited in capacity, so that we can simulate resource allocation and queueing. As results, we gain:

• The utilization of each function over time. We can fuzzify this as “low”, “average”, “high”, even though also comparing against the capacity might be a better measure (see 3.1.3, “Mapping Multiple Quantities”).

• The number of patients crossing between two functional areas (taking intermediate goals into account). Fuzzyfied, this gives the interaction (“high”, “average”, “low”), which can be compared with the adjacency matrix.

• The average path length for a transition between two functional areas can be used in its fuzzified form for obtaining proximity (“near”, “average”, “far”) as experienced by a patient. Note that, again, intermediate goals are taken into account.

The examples stated above are only examples for quantities to be mapped into qualities which can be compared to the preset requirements. For a much more elaborate list of these in the context of planning, we forward interested readers to Durmisevic, Ciftcioglu and Sariyildiz (2001).

What still needs to be discussed, however, is how we come to the membership functions in the first place. We have observed that there is no “standard” guidance on this topic, since for example the meaning of “near” or “far” varies even among hospitals. For example, an acute hospital will likely be very strict in what it means by “near” and “far”, while a general hospital might balance out the categories a bit. In line with literature, we thus argue that this is really expert knowledge, which must be sampled per project (which is what we did, even though this was not easy and certainly not a
topic of common agreement among the professionals either - we had to find a balance).

### 3.1.2. Automated Requirements Checking

In addition to producing linguistic results, a consistency checker can automatically compare what stands in an adjacency matrix versus the results obtained as described in Section 3.1.1. However, this comparison must take the membership degree into account (or, it may use $\alpha - cuts$, as described earlier, for filtering). It must also know the semantics of the linguistic terms. Consider, for example, the four example cases given in Table 1 concerning proximity between two functional areas: In the first case, the required proximity is not stated; hence, it does not matter what the simulation result is - no warning is raised. In the second case, “near” is required but “far” was computed - clear case for a warning. In the third case, proximity “near” versus “average”, the case is not so clear: It really depends on the project if a warning should be raised. The same goes for case four, in which “average” is demanded but n/a is produced (e.g. because the number of agents crossing between the two functional areas is too insignificant to conclude anything).

<table>
<thead>
<tr>
<th>Spatial Proximity</th>
<th>Required</th>
<th>Simulation Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>“near”</td>
<td></td>
</tr>
<tr>
<td>“near”</td>
<td>“far”</td>
<td></td>
</tr>
<tr>
<td>“near”</td>
<td>“average”</td>
<td></td>
</tr>
<tr>
<td>“average”</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example of linguistic comparison

### 3.1.3. Mapping Multiple Quantities

There are cases where one quantity alone is not sufficient for obtaining a quality. Consider, for example, the newly introduced quality “collaboration” in Figure 2: Let us say there are only two membership functions for this, “good” and “bad”, where only “bad” is shown in the example. Then, one rule for obtaining “bad” out of values for proximity and interaction would be:

\[
\text{IF proximity is far AND interaction is high} \quad (6)
\]

\[
\text{THEN collaboration is bad}
\]

In practice, all combinations of proximity and interaction would be specified as rules in this fashion, before evaluated by a fuzzy inference system as presented in the fashion described under Section 2.1. As has been noted in literature (Kosko 1996), however, combining all input variables with each other leads to an exponential growth in fuzzy rules, also known as the \textit{curse of dimensionality} (Bellman 1961). Furthermore, a verification concerning the consistency of the rules becomes an issue, not only because of the high number of these but also because of the information loss that occurs through the application of the AND operator (i.e. minimum function). If several rules need to be satisfied at the same time, the accumulated imprecisions can effectively outweigh the inference model, making it ineffective. Another limitation of Mamdani inference is that it does not allow for making a difference in relative importance among rule premises. For example, one notes that the degree by which adjacencies are deemed important varies - White (1986) terms some demands as ‘vitaly important,’ and others as ‘desirable,’ alternatively he refers to them as ‘crucial’ or ‘helpful,’ indicating this common difference.
performance assessment should take this information into account, which is a challenging issue in general, and remains unresolved with respect to Mamdani approach.

Several alternative approaches have reported: For example, neuro-fuzzy networks introduce machine learning as means to select appropriate rules, at the cost of transparency. Supplemental to this, Genetic Algorithms (GAs) can also help in this respect (Ciftcioglu, Sariyildiz and Bittermann 2007). Second, using the alternative Takagi-Sugeno inference model (Takagi and Sugeno 1985) can decrease (but does not alleviate) the mentioned shortcoming. Third, the shortcomings in relative importance among the linguistic terms have been put into the linguistic comparison (Section 3.1.2.) rather than being part of the inference, for pragmatic reasons: One might also introduce weights (as in tree-based neuro-fuzzy approaches).

4. IMPLEMENTATION RESULTS

Currently, we have two implementations: The basic flow simulation (written in Microsoft Visio) that simulates the movement of patients between functional units. Each functional unit can consist of multiple functions, which can either be unbounded (resource with unlimited capacity) or bounded - in which case we simulate queuing. In all cases, functions record their utilization, agents remember their agent history and so on; we can draw a lot of data already from this model, alas, there is no notion of actual circulation or way system within the building. Thus, this simulation cannot offer insights into the path lengths used by each individual, which is the basis e.g. for estimating transitioning lengths. We have opted to write a second implementation of a more academical nature, which tries to present the ideas given in this paper and extends on the missing circulation/way network part of our base implementation. The program (which was written in NetLogo) is given in Figure 3:

- Schemata can be generated automatically or imported from a sketch (sic!). The first functionality is purely exploratory - it helps in finding a basic decomposition of a clinic. Once this is known, a manually drawn sketch can capture the essence of what an architect wants to have better, and thus we have provided a vectorizer/tracer that can infer schemata from hand-drawings. Similar techniques have been employed in Architecture for some time now, see for example Koutamanis (1992).
- Functions are attributed to spaces randomly from a preset pool, in order to be able to vary not only the basic spatial structure but also the functional assignments within.
- After manual positioning of entrances, a flow simulation is performed. Agents are given a chain of functions to visit (goals). Crossing a function increases its utilization. Secondly, if coming from a previous function, we record an interaction between both units. These interactions can then be checked against a preset adjacency matrix (see middle part of Figure 3), highlighting areas where the linguistic specification (“low”, “average”, “high”) is not met.
- The interaction check uses the presented fuzzy concept for three single membership functions. We map the quantitative value for interaction between each two functions. For example, we might have an interaction of 10 between OT and ED, which gives low=0.3, average=1.0 and high = 0.2, depending on the choice of membership functions used. The preset is “low”, thus we take the low membership of 0.3 and apply a threshold=0.5. Since 0.3 < 0.5, we may not conclude that the OT and ED have a “low” interaction. Thus, we highlight the corresponding entry in the adjacency matrix red (middle-left part of Figure 3).

Figure 3: Screenshot of the NetLogo implementation featuring generation of floor plans and evaluation of interaction by comparing simulation results with the preset adjacency matrix.
5. SUMMARY AND FUTURE WORK

In this paper, we have argued for benefits of fuzzy logic as means to map the results of a simulation performed in the context of early-stage hospital design back into the language of the requirements, in our case: adjacency matrix and bubble diagram. As contributions, we have targeted three fields: (1.) The mapping of quantitative values to qualitative statements, (2.) automated checking of qualitative results against requirements and (3.) mapping multiple quantities to qualities.

Summing up, our approach works fine in the case presented - adjacency requirements versus flow simulation within the architectural schema, but is clearly limited in handling a larger set of quantities being combined, because of inherent complexity issues inherent to our underlying inference system.

Addressing the issues in an extended model is clearly on our wish list, respective future work items having been highlighted in the text; however, these require extensive effort not only with regard to the underlying technical concepts, but also when considering our target audience: Our implementation is currently intended for clients (hospital holdings, clinics) and planners (architects and process planners), such that we can give them a design tool which is simple and offers some (limited) capabilities for process simulation. In the light of this, it is vital for us that what the software does is rather unsurprising, as the users are quite unfamiliar (if not reluctant) with regard to the technical concepts in the background. It is thus no surprise that we have first concentrated on the direct case, where the 1:1 relationship between the simulation’s result and the resulting quality is obvious. Adding the possibilities for multiple quantities to be combined must be equally comprehensible.

ACKNOWLEDGMENTS

Siren image in Figs. 1, 2 by Sebastien Durel is being used under cc-by-attribution, cc-non-commercial, cc-no-derivative, cc-share-alike license. We wish to thank our reviewers for their inputs, especially concerning the use of multiple quantities. We further wish to thank the city of Vienna, which financed parts of this research work under the ((MODYPLAN)) project.

REFERENCES


One of the key benefits of early-stage simulation is the ability to perform capacity planning within the preliminary schema. In simulation terms, this relates to seeing functions as resources having a certain capacity, i.e., the number of agents able to utilize them in parallel. The utilization of functions is usually shown in the way of an graph (see Figure 39) which charts utilization over time: A utilization above the capacity line is said to be over-utilized (leading to the formation of queues), the converse is under-utilized.

In an ideal case, utilization and capacity lines are identical. However, such a planning goal is hard to achieve due to a number of reasons: First of all, there might simply not be enough resources (think: costs) to satisfy all requests. Second, and more important, resources are not isolated but closely coupled through the building user’s process: A queue in front of one functional unit will lead to delay the utilization of all consecutive units which are in the user’s process. In line with that thought, the usual way to establish what bottlenecks were causing disturbances was to repeatedly run a simulation and to closely monitor the utilization graphs. In this paper, which I wrote together with Wolfgang E. Lorenz, we have been seeking to avoid this cycle of “guess, re-run, verify” by introducing a visualization on top of the Schema in which we can see causalities underlying observed utilization problems without having to restart the simulation.

Authorship Information. Wolfgang E. Lorenz concentrated his efforts on extracting utilization classes from the utilization graphs, while my part was the overall visualization of causal chains. The accompanying implementation in Netlogo was written by myself.

Main Contributions.

- Argues for the depiction of utilization chains in analogy to debugging, showing the run-time dependencies between the different resources.

- Introduces an interrogatory interface in which a simulation can be asked at a given instant (1.) why a resource is utilized (2.) who is utilizing a resource and (3.) when that happens.

Type of Work. Peer-reviewed conference paper, showcase model in Netlogo. Contribution to STABLE HBO (p. 188) and MODYPLAN projects (p. 189).


Figure 39: Utilization graph
Causality in Hospital Simulation Based on Utilization Chains

Gabriel Wurzer and Wolfgang E. Lorenz

Vienna University of Technology
Treitlstrasse 3 (Architectural Sciences / 259.1)
Vienna, Austria, 1040
{firstname.lastname}@tuwien.ac.at

Keywords: Agent-Based Simulation, Analysis, Debugging.

Abstract
The operation of complex buildings (e.g., airports, hospitals, industrial facilities, penitentiaries) is commonly simulated forward in time: agents arrive and perform their prescribed tasks, utilizing resources and space as required. When trying to understand the model's state at a certain point in time, say, "why is this resource over-utilized?", one must either guess or run the simulation again to determine what the cause is. Our contribution lies in the introduction of causal chains into the workflow of an agent-based simulation, so that an end user (in our case, process planner and hospital architect) can get a further insight into the intermediate simulation result at a certain point in time, without having to re-run the simulation.

1. INTRODUCTION
In the past half year, we have been simulating a large central operations theater (eight operation rooms minimally, the total number dependent on simulation output). With this work, we regularly had to hunt down errors and inconsistencies. Even when having full access to the model's code (white box approach), this was already quite a challenge, as there are many causes for an observed behavior (activities of an agent, signals, queuing and prioritization, etc.). For someone without access to the code (i.e. black-box approach, typically end-users), finding the erroneous spot might be even more troublesome: one could either guess (basically using trial and error) or resort to more sophisticated protocol-based methods such as (Rogsch and Klingsch 2011). Clearly, it is hard to argue for the trustworthiness of a model under such circumstances. Our major goal is thus to give end-users a greater flexibility in exploring a model during its development phase.

2. END-USERS NEED MORE THAN RESULTS
Process planners and hospital architects (who are our end-users) are not confident in simulation results lest they can understand how they were computed. For example (see the left side of Figure 1), they will want to see not only a utilization graph but also dependencies between different resources. It is common to shift this responsibility to the visualization (see the right side of Figure 1), meaning that one has run a simulation multiple times in order to understand the relationships between the patients and resources fully.

What if we could instead stop the model at a certain point in time and find out the chain of events that led to the current state, without having to restart? This is the core idea and main contribution of our paper, which we detail under Section 4. Summarizing briefly, we might then ask—for a specific space under scrutiny—the following questions:

- Why are resources utilized (causal chains, Section 4.1)?
- Who is using certain resources (role-centric, Section 4.2)?
- When are resources utilized (time-centric, Section 4.3)?

It is our hope that, using such functionality, users will be less tempted to call for “realistic visualizations” for proving
the credibility of a simulation but can instead resort to proper argumentation.

3. BACKGROUND

3.1. Simulation Model

We use an agent-based model in which each agent follows a fixed sequence of functions to visit (that is, the medical pathway, which might consist of functions ARRIVAL > [HOLDING AREA] > PATIENT TRANSFER SYSTEM > OT > RECOVERY). This is similar to a schedule-calibrated occupant behavior simulation (Goldstein et al. 2010) or User Simulation of Space Utilisation (Tabak 2008). We use real (but anonymized) pathways exported by the Hospital Information System (see Wurzer et al. 2012; Glock et al. 2013 for details) or generate these by using “standard sequences” from which we eliminate steps based on probabilities, also determining step durations using min/max service times per function.

Functions form the dynamic aspect of our simulation: they are resources which can hold a finite number of agents at a time (capacity) and ultimately determine when an agent is able to proceed with its process. Thus, one might call our model a more discrete/client-server-based approach than agent-based approach, even though we also use agents for their ability to navigate in space (along the circulation, in the preliminary schema of the hospital, using a pedestrian movement model if required). Each function is situated in a space (e.g. OT1), which is the basic planning unit that our end users are after. There are two modes in which functions are acquired. First, an agent can utilize a function directly because it is in his process (function as activity). Second, a function is utilized as queue when another function is unavailable (e.g., preoperative holding area).

Histories are recorded for both agents and spaces: agents record at which time which function was sought or acquired, or a queue was entered, whereas spaces record which agents have visited or queued for a contained function. Using these basic recording mechanisms, we can later infer a causal chain.

3.2. Related Work

Techniques that can deliver the functionality we propose have already been available for some time in the context of debugging. An illustrative example of interrogative debugging techniques, (Ko and Myers 2004) have proposed a system that can interrogate a program over “why” or “why not” a certain program state was reached, by recording program states in between. In contrast to this approach, we need not “infer” causality (cf. Pearl 2009; Hluch 2014), as this is explicitly given in the medical pathway that each agent follows (see Section 3.1).

Visually, our approach uses Sankey diagrams (cf. Tufte 1983, p.176) to depict flow between spaces. A similar technique that depicts temporal events from medical records has been previously presented by Wongsuphasawat and Gotz (2012), albeit with no connection to the building layout on which our approach superimposes the visualization. Such an approach can be seen as Focus+Context technique (Card et al.1999) where a space under scrutiny is shown in full detail together with flows from other spaces leading there (giving context).

4. CONTRIBUTION

4.1. Why are Resources Utilized?

The answer to this question lies in all spaces that an agent has crossed before coming to a space under scrutiny. As we have recorded all utilizations for these spaces, we may thus speculate if these have led to a bottleneck (also refer to Figure 2). If agents are utilizing a space below its capacity, no queuing occurs (labeled green in the left part of Figure 2). If, on the other hand, agents queue for that space (labeled red, see again the left part of Figure 2), there is a potential for that space to have become a bottleneck.

![Figure 2](image-url) Utilization of a function. (left) Utilization with queuing depicted as red. (right) Classification into underutilized (yellow, <20% usage), well-utilized (20%-100%) and over-utilized (>100% = queueing).

The latter aspect deserves some attention, as there are two possible interpretations. Either we say that a space for which queuing occurs [even once!] can be the source of a bottleneck (absolute utilization). Or, we disregard minor queuing activities and focus on whether queuing occurs “most of the time” (relative utilization). In that context (refer to the right side of Figure 2), a space may be “under-utilized” (< 20% utilized), “well-utilized” (20-100%) or “over-utilized” (queueing occurs, thus >100%).
Figure 3. Causal chain depicting “why?” the space on the lower-right has been utilized (red = over-utilization, yellow = under-utilization, green = well-utilized). Actual screen shot from our simulation.

Figure 4. Utilization from a patient-centric perspective (“who?”). Patient flows depicted as colored edges (red = acute, green = planned, blue = day-clinic patients). Thickness of edges corresponds to patient volumes. Relative utilization of spaces shown as color-coded circles (red = over-utilization, yellow = under-utilization, green = well-utilized). Circle radius shows patient volume having crossed that space. Actual screen shot from our simulation.

4.3. When are Certain Resources Utilized?

This information is readily available in the utilization graphs (see the left of Figure 1); however, anchoring it to the chain of events may make patterns and dependencies in the temporal domain more obvious. We superimpose temporal utilizations over the causal chains (see Figure 5), by inscribing usage plots into circles showing relative utilization. A benefit of having such visualization is that planners can get a feeling for circles showing relative utilization without having to re-run the simulation.

4.2. Who is Using Certain Resources?

To answer questions on utilization for a specific role (e.g. “are acute patients causing a bottleneck?”), a separation of causal chains by role is performed (refer to Figure 4). Colored edges now show patient type (red = acute, green = planned, red = day-clinic patients) and volume of patients crossing a space under scrutiny. As before, we show the relative utilization of the space as a circle (yellow = under-utilized, green = well-utilized, red = over-utilized), which we scale according to the patient volume. Large volumes of a specific patient type and visible utilization problems along the causal chain can help to pinpoint a bottleneck in the process for a certain group of building users.

5. DISCUSSION

Our approach is targeted at planners wishing to ask “why”, “who” and “when” types of questions during the development of their model—either for analysis or for debugging purposes—leading to some confidence in the simulation results. The underlying model consists of agents visiting their respective medical pathways (or causal chains, as we would say). However, there is no need to have fixed sequences of visited functions as a basis for our approach -
it will just work fine with any other agent-based simulation for which agent histories and spatial utilization are recorded.

Another point worth discussing are the results of our underlying model (i.e., OT utilization, number of patients, room configuration and so on). We have intentionally left these out, since what we are interested in is tool support rather than a concrete case, even though our simulation was initially built with a 1100-bed clinic in Vienna in mind (but has since been extended so as to become more general-purpose OT planning software). Arguing further, we are only interested in preliminary phases of building design, where space programming takes place. In contrast, most papers in the same field target optimization of hospital units in the later phases of design, where concrete results can be given because the main decisions in space programming have already been made.

The chosen visualization types—"why”, “who” and “when”—come from actually working with practitioners while developing our model. To be fair, we have yet no indication if our approach can be considered a “success” or “failure” in terms of aiding the design team for the general case. Proof or rejection of this statement would require us to test our model with one group using the visualization and one without—a task we have to leave for future work.

6. CONCLUSION

We have presented a visualization of utilization chains in agent-based models, which lets planners ask “why”, “who” and “when” a certain space and its contained resources were utilized. Interrogating a model in this way can help users gain confidence and hunt down errors during the development of a simulation.

Acknowledgements

This work was sponsored in part by the ZIT MODYPLAN and the FFG STABLE HBO project. We wish to acknowledge fruitful discussions among the project members, especially with Anita Graser, Barbara Glock, Richard Wurglitsch, Niki Popper and Rudolf Linzatti.

References


This short paper addresses the circulative network in a building, both in early-stage planning as well as in later stages. It assumes that the performance of a building concept in the context of evacuation depends on the redundant design of evacuation routes (multiple hallways leading to multiple entrances). To test such a performance, we first transform the building into a graph of functional areas and exit nodes. We then iteratively take away one edge (standing for a route between two areas which is made unusable, e.g. by fire) and simulate an evacuation using the pedestrian dynamics algorithm by Blue and Adler for every hour of the day. The latter is important, since there occupancy of a building changes over time (e.g. offices full in the morning but empty at midday when people go for lunch), leading to different evacuation times. The delta between the minimum and maximum evacuation time is what we call “sensitivity” of the circulation against congestion and blockage.

Authorship Information. The work on the paper was led by myself with inputs from all co-authors. The accompanying implementation in Netlogo was written by myself.

Main Contributions.

- Transforms a floor-plan back into a bubble diagram of functional areas and accesses, connected by edges which stand for the circulation.

- Simulates evacuation based on occupancy at different hours of a day. Infers and visualizes sensitivity against congestion from minimal and maximal evacuation times. Computes sensitivity to blockages in the same fashion, by iteratively leaving circulation edges away during the simulation.


Additional Graphics. The paper is a bit short on graphics substantiating the approach. Therefore, I herein include additional material that from the same simulation model:

- A floor-plan with its circulative network forms the basis for the simulation (see Figure 40). In that floorplan, we simulate egress such that (1.) each of agent determines the route to the nearest exit using a waypoint graph, resulting in a list of waypoints which every agent (2.) processes using the Blue-Adler pedestrian algorithm.

- On a higher level, the building concept is seen as a graph of nodes standing for functional areas connected by edges standing for the circulation (see Figure 41). This graph records the throughput of agents (thickness of nodes and edges) and also visualises the sensitivity against blockages (red: sensitive, green: not sensitive, gray: no definite answer possible) which is computed by taking the delta between minimal and maximal egress times per hour and edge/node.

![Figure 40: Floor-plan with route network as basis for the ABS.](image)

![Figure 41: Bubble diagram of functional areas with superimposed sensitivity (node and edge colors) and throughput (node and edge sizes).](image)
In a final step, the sensitivity against blockages is computed by leaving one circulation edge in the bubble diagram away, thereby making that passage unusable. The overall sensitivity against blockages is again depicted by node and edge colors (see Figure 42). It is always important to relate the sensitivity to the average throughput (see line thickness in Figure 42): Edges that are sensitive but have little throughput (FA1-H, FA2-H, H-EA3) are not grave, since the flow volume passing through that is little.
Sensitivity Visualization of Circulation under Congestion and Blockage

Gabriel Wurzer, Matthias Ausserer, Harry Hinneberg, Christa Illera and Andrea Rosic

Vienna University of Technology
Vienna, Austria
Corresponding author: wurzer@iemar.tuwien.ac.at

Abstract In the context of circulation design for large buildings (e.g. hospitals, airports), the question of sensitivity of the path network against congestions and blockages naturally arises. To date, the answer to this question would require planners to use a simulation package, which is, however, almost never done in the early stages of building design. We therefore propose a novel visual planning tool that enables architects to estimate the impact of disturbances on the building circulation without having to use a simulation package. Our approach is integrated into a common CAD system and visualizes changes in the path-time relationship of adjacent functional areas under the effects of impeded accessibility.

Fig. 1. Based on an attributed floor plan (left), we perform a visualization of sensitivity against congestion and blockages (right). Nodes are used for functional areas and edges for circulation. The colour of each edge gives the sensitivity.

Introduction

Our planning tool is a graph which visualizes functional areas as nodes and connections between functional areas as edges (refer to Fig. 1, right). A functional area is a collection of pedestrian sources and thresholds (either leading to the next functional area or being exits). Each pedestrian source models the usage of a space in a time span t0 to t1. Usage data is of high interest in complex buildings which
are designed around pre-determined processes (e.g. hospitals, airports). Our goal is to try to visualize flows along the circulation (both in and between functional areas), the target audience being architects who want to get “insight, not data” (as once proposed by Hamming).

Our program is integrated with a CAD system. As starting point, we take a basic CAD drawing that depicts walls and thresholds as input. This drawing is imported as raster image with a resolution of 0.5m, and must then be attributed with functional areas, pedestrian sources and threshold points in a pre-step (Fig. 1, left). A cell-space simulation then computes flows from the sources to the exits in the set time span, giving us:

- the number of people crossing each circulation between two functional areas
- the duration of travel along this circulation

This information forms the basis for our visualization, which will now be described.

**Visualization of sensitivity against congestion**

The thickness of each edge (see Fig. 1, right) depicts the throughput of a circulation, i.e. the number of people crossing it in the set time-span. The same goes for the size of each functional area node, which tells the amount of people that have entered it. We have opted use the absolute value of the throughput, scaled by a user-defined constant, which gives us good results.

The duration of travel along the circulation is depicted as edge color. However, we do not use this value directly, but first look at the whole time-span that is set. It is the difference between the maximum travel times that is taken gives us a measure of congestion in both functional areas (nodes) and the circulation in between them (edges), which we map to a red-green color gradient. In this sense, attaining “thick green lines” (high throughput, little sensitivity against congestion) is the preferable goal. The mapping of differences in travel times onto colors is obtained by taking ranges and mapping these to discrete colors (classification of absolute values).

**Visualization of sensitivity against blockages**

In a pre-step, the simulation runs through the set time span and records, for each functional area node and circulation edge, the maximum travel time. Then, the algorithm subsequently removes an edge from the circulation graph and performs the simulation again. The difference between the recorded and the current maximum duration time gives a measure of the sensitivity against blockages in the chosen time span, which is again mapped to a red-green color scale in the same fa-
The throughput is visualized in the same way as before, as thickness of circulation edges and functional area nodes.

**Underlying simulation**

Our simulation uses a raster image that was exported from the CAD drawing as input into a cell-space simulation algorithm, which performs the actual work of generating the data that is to be visualized. Furthermore, we require that, for each pedestrian source, there is usage data (in the form of a spreadsheet table). This usage data tells the simulation engine how many agents to generate at a given time in which pedestrian source area (see Fig. 2). The agents are then simulated using the model of Blue and Adler [1] which is extended with a higher-level exit route choice function, in order to support choosing exits on the bases of functional areas. Furthermore, the extension is also responsible to disallow access to functional areas that are currently marked as “blocked”.

Each performed simulation run measures when and how many agents cross area borders. These resulting values are recorded for later use in the visualization. When crossing thresholds, agents must choose their next target using our extension algorithm that considers adjacent functional areas. If there is such an adjacent functional area, the agent resets his internal clock and crosses the functional area to find an exit. In due course, the number of agents crossing the circulation is incremented by one. Upon reaching the exit, the agent records the total time of travel for the circulation he has just crossed, and the choosing of the next functional area continue. If there is no further functional area to go to, the agent is taken out of the simulation (the exit can then be considered as safe area).

**Previous Work**

Our work employs usage data of functional areas to aid the planning process. Previous work in this context recorded and simulated building user’s activities [2] in order to assess the building design’s performance. Extending CAD systems with user activities has also previously been researched in [3][4][5]. Furthermore, our work focuses on providing meaningful visualizations of simulation data, which have been considered in [6][7]. From the view of pedestrian dynamics community, architectural considerations have been previously brought forward by [8][9].
Conclusions

We have brought forward the idea of a novel diagram that lets an architect assess the effects of congestions and blockages on the planned circulation. Our concept is integrated into a regular CAD system and can be used during the early stages of building design. We are confident that, using our approach, architects can further improve the circulative system (i.e. add redundant paths, or increase capacity) in order to design safer.

We are aware of several points that our approach does not address and that require a justification. First of all, our underlying simulation is rather simple (for example, we have not considered taking any response times into account). In the context of this paper, simulation is merely a basis for gathering data, which is to be fed into a higher-level visualization. It is clear that more elaborate forms of simulation algorithms that have been in existence for a long time could be added. As our prototype will be published under an open-source license, we encourage interested researchers to do so.

References

Additional Scientific Works

Reports on work which was scientific, but did not quite fit into the subject of agent-based planning. One of these topics is community building, establishing a web-based word processor, while the other is on my application of ABS for archaeology.
Paper: ProceeDings - A web-based word processor

One of the most annoying aspects of conference organization is the need to produce a proceedings book out of the authors’ individual papers, given some of the following problems:

- Inconsistent formatting or broken template styles coming from copy-and-pasted content.
- Incomplete provision of citation information.
- Figures with too high or too low resolution.
- Broken formulas.

To help the proceedings publication process and provide a consistent layout for all eCAADe conferences, I have therefore programmed an online word processor, coined “ProceeDings”\(^{55}\) (see 43), which addresses the aforementioned problems. The implementation was successfully used for the production eCAADe 2014 proceedings books - more than 1200 pages totally, and can therefore be said to have stood its test. Even though great for the scientific community (and especially for the organizers), I am aware this is of zero scientific significance. However, because we could record the process of paper preparation of each individual contribution of eCAADe 2014, we were able to gain insights into the behaviour of our authors - which might indeed be valuable for future hosts.

Authorship Information. The authors contributed equally to this paper. ProceeDings written by myself and will be released as open-source software next year.

Main Contribution.

- Provides insights into the behavior of authors in the architectural community. Mines into the provided content, identifying predominant types of paragraphs used. Looks at hand-in behavior in relation to the deadline (and its extension).

Type of Work. Peer-reviewed paper, implementation in PHP.

In recent years, the question of how to teach programming to architecture students has often led to uncertainty (Edelko, Czibulka and Martin, 2010; Brazil, 1998; Byers and Theimer, 2009; Barry, Batta and Avrun, 2010; Fricker, Wartburg and Hosenstaff, 2008). Advocates of functional programming rely on those who prefer imperative programming, people who prefer one programming language over another, and everyone is at odds with one-self on how to meet the goals being set forth for their courses. There is a strong wish for sharing insights on how to teach architecture students programming, which has lead to a variety of publications on the topic. One thing largely missing from the discussion is, however, a fresh look at the subject from a student's side. In the course of a programming workshop, we have therefore been assessing the subjective impressions of students, and have compared these to the actual presented content. Being a case study, we see our contributions both in the conducted work (see “Case Study”), as well as in the discussion of the results and actual consideration points which we believe can help fellow lecturers improve their courses (see “Discussion”). Given the perspective that we would like more teachers to conduct student-centered assessments, we are looking forward to a format which we believe can serve as practical assessment basis, in the form of questionnaires to be handed to the students (based on an Excel chart).

Figure 43: Editing a paper in ProceeDings.
A web-based word processor automating the production of conference proceedings

Gabriel Wurzer\textsuperscript{1}, Bob Martens\textsuperscript{2}, Thomas Grasl\textsuperscript{3}
\textsuperscript{1,2}Vienna University of Technology \textsuperscript{3}SWAP
\textsuperscript{1,2}http://www.tuwien.ac.at \textsuperscript{3}http://www.swap-zt.com
\textsuperscript{1,2}\{gabriel.wurzer|b.martens\}@tuwien.ac.at \textsuperscript{3}tg@swap-zt.com

In this paper an online editing system for eCAADe papers is presented, which is also the technology behind this volume. On the occasion of the eCAADe 1999 conference in Liverpool, a novel layout for the proceedings was developed. In the course of forthcoming annual conferences, this became the distinctive "look and feel" for eCAADe papers. Due to the complexity, professional typesetting was required for and the authors were disconnected from the publication and layout stage. This paper elaborates on the development and implementation of a web-based tool, which takes care of the typesetting and delegates this activity to the authors. Neither software installation is required, nor specific training must be completed in advance. On top of this the degree of homogeneity can be raised significantly, thus supporting the editors in charge to concentrate on the task of harmonising the publication content.

Keywords: Word Processing, Proceedings Preparation, Cloud Computing

OVERVIEW
To the annoyance of both authors and editors, word processing packages are still in their infancy when it comes to guarding a publication template against modification and improper use. Modification, i.e. inserting or altering template styles, is a 'feature' that is often seen when copying and pasting between documents or from the web. Improper use of a template, on the other hand, is a failure to meet the semantics of the entered content itself - think, for example, of entering references incorrectly. On top of this, the authorship is in charge of the final layout and obliged to stay within the given page-limit. Until now, the workflow suffered from a sharp cut between the word-processing-stage and the loosely coupled layout software stage.

During the past three years, we have been developing a web-based word processor that copes with the mentioned problems: It enforces eCAADe's own conference proceedings style and makes use of structured content with added semantics. The resulting contributions can then be automatically compiled into a proceedings book, or exported into \LaTeX for further editing.

We will start with a brief outline of our solution, but then immediately come to our main contribu-
tion: The development process behind such a massive web application. Decisions taken in that phase are often far from obvious, which is why we would like to share some of our insights with soul mates who also want to bring their app to the web or into the cloud. We will conclude with a brief statistical overview of some of the data collected during the submission process. This contribution targets hence the wider eCAADe2014 conference theme of "Fusion" in the sense of a collaboration tool aiming at data integration.

BACKGROUND AND RELATED EFFORTS

When speaking about the editing of conference proceedings, managing the time-line is the dominant issue. In the given area of CAAD the (printed) proceedings ought to be finished by the beginning of the conference; a post-conference publication is no option, neither is preponing the deadline for submission. This means that within a timeframe of approximately 12 weeks (or so) roughly up to 150 papers with a significant amount of figures/tables/images need to be edited. Therefore, it makes sense to delegate a justifiable amount of work to the authors supporting the homogeneity of each individual contribution. An electronic publishing environment should be able to secure unintended violations of implemented rules/restrictions. It should be mentioned that data entry via a structured web mask is mandatory. The papers/contributions are to be centrally stored and perpetual backup routines must be implemented.

A first thought which instantly comes up is that a gaggle of editors must have been tackling these issues over and over. Possibly there exist different views on the level of consistency and accuracy to be achieved. It could, however, also be that some of the work involved is regarded as unavoidable and divinely ordained, be it alone the persistent use of upper and lower case in the title and headings. Indeed, wide-ranging reflective publications are numerous [such as Russey et al. 1995; Fredriksson 2001] and even a series of conferences has been dedicated to this topical area [such as King 2000; www.elpub.net].

When searching for similar tools predominantly a variety of tools can be retrieved, allowing to handle a collection of already created PDF-files. ProceeDings focusses on the contrary on the production of the publication entry itself, carefully respecting the predefined publication guidelines. As a matter of principle, content management systems (for websites) would allow to settle a similar task within a system of distributed roles. However, this would in most cases require a training, whereas ProceeDings claims to be almost self-explanatory.

A solution like ProceeDings does not make the guild of editors jobless as there is still a need for taking care of the content of the publication as a whole. Especially the setup of coherent sessions and their composition in (parallel) sequences is rather demanding.

PROBLEM: FORMATTING

It is not the first time that the eCAADe has created a web-based platform for its community: The Cumulative Index on CAD (Martens and Turk 2000), for example, was founded as paper archive for education and research in architecture and urban planning. We have targeted the same community with our web-based word processor, however, it is more than likely that we will find us serving a greater range of fields in the future.

There were two main drivers that led to the development of the tool:

- We were unsure what to do about Microsoft Word's persistent tendency of breaking the formatting of a document. Locking the template had only mixed success, and likewise did Springer's Manuscript template (which uses Visual Basic for Application Macros for taming Word to some extent, but: The user has to enable Macros and copy-and-pasting from the web automatically allocates new paragraph styles...).
We felt that a general-purpose word processor is ill-suited for the conference’s needs, as the user enters text but the program does not enforce semantic and structural rules (see next section) that lie at the heart of every paper. \LaTeX{} does (or better: can do, if instructed) all of this, however, it is not common in the field and requires a certain degree of a learning effort. Having a larger audience switching from Word to \LaTeX{} was, as can be imagined, also not feasible.

**SOLUTION: SYNTACTICALLY DUMB, SEMANTICALLY RICH**

Microsoft Word’s problems in keeping formatting intact is the result of mixing text semantics (heading, paragraph, bullet list,...) and styling (font family, size,...). Copy-and-pasting brings out the worst of both worlds, in the sense that the text is pasted using the current paragraph format, which is automatically updated to reflect the pasted text’s styling. Discarding styling information altogether and pasting only the text would be the obvious solution, however, there is no way of forcing this behaviour without user intervention, e.g. by storing settings together with the template.

Pasting “only text” does do away with superfluous
uous styling, however, we completely loose text semantics - the paragraph format, bold, italic, and so on. In our web editor, we have therefore devised a two-fold editing strategy that enables us both.

**Semantically rich.** We regard each contribution as a set of paragraphs containing plain text (see Figure 1a). Paragraphs are context-aware: For example, a heading cannot be moved further than the list of keywords. One may not add an abstract block in the body of the paper, or anything else but references in the references section.

**Syntactically dumb.** Paragraphs are edited - one at a time - simply by clicking them. In the simplest case of a “paragraph” format, the application then shows a text box where one can enter or paste plain text (see Figure 1b). For paragraph types that acquire structured information, such as the list of authors or a reference entry, we present not one but many plain text boxes in which the user types in the needed information. Formatting beyond the paragraph level is done using a wikitext-like syntax, for example *bold* or /italic/. What wikitext markup is available in each paragraph ultimately depends on its type: For example, headings offer nothing but plain text, while paragraphs furthermore enable **bold**, *italics* and *equations* done in this manner.

**TECHNICAL OUTLINE**
Our tool is composed of four technical layers:

- **Editor Front-end**: We chose to implement a “Web 1.0” application which is based on classical request/response cycles, utilizing PHP (served by nginx/php5-fpm) for generating the HTML presentation (styled by CSS). Optional functionalities such as the real-time preview of formulae are handled via JavaScript libraries (jQuery, MathJax) which degrade gracefully in case of missing browser support. As a matter of fact, the application displays even on outdated browser versions (we tested e.g. Internet Explorer 6 and the text browser Lynx [1], which dates back to 1992).

- **Editor Back-End**: Instead of a classical database, we store our documents as files. The reason behind this is that load tests showed database performance to drop when memory is constrained, whereas files-based access remains fairly stable even if the server is under heavy load. Furthermore, we utilize a memory database (memcached) for fast access to back-end data and session management.

- **Printing**: Documents are converted into **\LaTeX** for printing. The actual generation of a PDF is a very costly task, and therefore, we have utilized a flexible “printer queue” (utilizing Beanstalk for PHP) where multiple other servers could help in generating a print preview of the currently edited document. On the other hand, having too few resources to serve all print requests will not overload the server, since all clients are queued and only some are served at a time.

- **Book Compilation and Metadata creation**: The final compilation looks at a spreadsheet stating (1.) in which volume is (2.) which session, containing (3.) what contributions. It then re-renders all PDFs in that order, giving the correct page numbers and also assembling the list of contents in the process. The resulting PDF are the proceedings, but not the final product: We still have to do an extraction of metadata (e.g. authors, title, references) and creation of metadata for indexing, in for example CUMINCAD, automatically.

All these parts work together so that authors can collaborate in producing a consistently formatted proceedings that saves the editor valuable time which can now go into the actual editing process. Scientifically though, these points are hardly of concern. Indeed, what we are more interested in is how the community works with our tool, and when this work is delivered. An analysis of this is brought in the next few sections.
LESSONS LEARNED

As eCAADe 2014 was the first time that a web-based editing tool was used, we coined it ProceeDings\textsuperscript{Beta}. Due to the constant correspondence with the authors and monitoring the layout of the whole proceedings ourselves, we are able to give some insights on what worked and what did not.

What worked

Inserting well-behaved text. Pasting paragraphs from the clipboard as plain text and formatting that with markup worked like a charm. The only exception were inline equations, which presented an own problem (chars that are known to Microsoft Word but not to Unicode, WingDings and the like).

Warnings. Most authors took a great deal of pride in having their paper warning-free: For example, the system would complain about too short or too long abstracts, and people strongly responded to that. That (in the end) the system was so liberal as to allow submission of every paper, regardless of warnings, was not of concern. The people simply wanted to help getting their work right, and this is one of the most assuring things which we encountered during review of all the final submissions. Some exceptions (which we have to deal with, of course) were the references, where the system would sometimes complain wrongly about a URL being wrong, when in fact they had only given an additional [accessed 2nd June 2014]. We are delighted by this ability to judge the system by ones personal experience in scientific writing - some authors also left use NOTE paragraphs explaining what they want to appear in the paper and why they could not accomplish - and thus this is one of the points which we build on for the next version, exposing e.g. more tools that we as editors had in figure positioning and paragraph indentation ("dont-indent" my paragraph, please).

Selecting the right kind of paragraph for the job. For people not concerned with structured text, terms such as "heading 1, 2, 3", "authored book" and "edited book" might not be very descriptive. So it shook us when we realized that people actually understand far more about paragraph formats than we had anticipated: After all, we had not given out a template in \LaTeX instead of the web-based system because of fear that this would be not understandable. The impression that we got is that even though \LaTeX might be too much to stomach for the whole community, the general concepts are clear for everyone - and this includes structured markup such as the ones mentioned.

What did not work

A paper is not an image gallery. It would be a lie to say that most authors added figures accompanying the text. In fact, it was the other way round: The text would accompany the figures! Given the very limited abilities of \LaTeX in positioning the graphics, this was as much of a pain for us as in layouting as has been for the authors in editing. The future perspective on this is clear: (1.) Constrain the use of images to cases where they are visualizing the text (every image needs to be cross-referenced, as mentioned in the User Guide), (2.) constrain the number of images to a minimum, we think of 5-8 at the moment and (3.) give more options for positioning the images in the text, accompanied by a clear description over how \LaTeX will attempt to position them.

Figure positioning. Figure positioning deserves some more attention: \LaTeX will position a figure either "right here" in the text or let it "float":

- "Right Here" means that images will be one column wide. If there is not enough space vertically in this column, \LaTeX will shift them to the next column (or even the next page, 1st column), leaving an ugly hole in the text. Authors need to know that they need to close these holes by inserting a "here" figure where it has enough space, at best in the middle of a column, surrounded by lots of text.

- "Floating" means that \LaTeX will put a figure either at the top or at the bottom of the next opportunity. That again means: When a figure is "beyond the top" of the page, it will insert
it at the bottom of this page or at the top of the next page. When the layouter has already crossed the "the bottom" of the page, it will insert it on "the top of the next page". In practice, this means that all floating pictures need to be defined well before their insertion position, so that the layout algorithm has them at hand when going through the text. Arguably, this is quite counter-intuitive. However, this is nothing that we can work around, as \LaTeX{} is built like that.

The mentioned points are even more enervating when preparing for different kinds of output media: For example, an eBook has the requirement that figures always appear exactly in the spot where they are mentioned, i.e. "here". For a printed proceedings, however, we may additionally use "floating" figures, which may need to be defined before the spot where they are actually referenced. Essentially this is the divide between structure (as in eBooks) and layout (as in printed proceedings). It will be our task to think about ways in which we can bridge this gap, providing more options for positioning (e.g. figure "on an own page", which was used during editing) and also for vertical spacing before and after figures.

Formulae. Formulae were the main cause why a paper did not print properly. This is no coincidence, but the result of two diametrically opposed policies on dealing with erroneous input: The web-based formula viewer (MathJax) would simply ignore all offending markup and display what it could make sense of without complaining, while the printing algorithm (\LaTeX{}) would immediately stop and report an error.

Errors produced by \LaTeX{} are handled by ProceeDings such that it shows an error page. It does, however, neither know what happened nor where the error lies (i.e. no parsing of error text, yet). Therefore, we display a generic error page that gives some hints over what could have gone wrong, but is no use when it comes to hunting down specific errors in a formula - leading to a lot of despair and troubleshooting via mail. In further detail, our analysis shows that there were three separate cases that led the printing algorithm to fail:

- **"Unicode" formula input into web editor:** What most people do is paste a formula from Word into a paragraph. Technically, this is no formula, but merely a set of characters that are hopefully \LaTeX{} - and the system will appropriately save them as normal text. In some cases, however, characters that Word pastes are simply not Unicode - they are symbols in Windows encoding! So this process can fail terribly, if we have no clue what this symbol is (remark: this is likely, as we have one programmer (currently writing this) against the rest of the Microsoft world). A better way would be to input a formula markup, presented in the next bullet point.

- **"Formula" input into web editor:** With the help of surrounding #, an author can insert an inline formula (the other option would be to make a standalone formula paragraph, which does not need that). Some authors have taken this hurdle, but kept pasting Unicode into them. This can go well - the formula is laid-out in the formula font instead of the text font - but this can also fail (when the character is not defined in the formula font).

- **Formula defined in ASCIIMath but \LaTeX{} is too dumb to digest:** For those authors that did embrace the formula syntax (technically: ASCIIMath syntax, an easy way to enter even the most complicated formulae), there were two further hurdles: Either the formula was not entered correctly or the the converter to \LaTeX{} simply produced erroneous results; the web editor would always produce a result, in the sense of "I am happy with what you enter; I will typeset what I understand and skip the rest"; but \LaTeX{} would not work this way - it would crash! So this went definitively wrong, such things should never happen.
Summarizing, we should definitely parse the error text given by \LaTeX\ and show the author what the error is, and where it lies in the paper. Another lesson learned: Do not use the conversion from ASCIIMath to \LaTeX, use the image produced by the web-based formula viewer, at least for non-inline equations. In that way, what you see would truly be what you get.

*What did work, but most authors say it didn’t*

References were a source of constant dispute between the editing team and the authors. As must be said, we had lack of support in importing from EndNote or BibTex, which is a shame fully taken. As a result, the authors had to re-enter every reference again, which was frustrating. However, the process of having to review every reference again according to eCAADe’s needs has proven very effective in assuring quality. In fact, it is clearly one of the conferences where quality of references is of the highest standards, since we enforced semantic rules rather than only taking “the data” that authors would provide (for example, journal articles require a volume and a page) which is far beyond what people would normally give us. Furthermore, it allowed authors having different citation management systems to produce one homogeneous reference section.

**A BRIEF LOOK AT ABSTRACT SUBMISSION**

ProceeDings was used for the editing of full papers, while OpenConf has been employed for abstract upload (Word or PDF). Thus, strictly speaking, this part should not appear at all in our paper, as it is something we are not concerned with. However, it might nevertheless put some contrast on our later analysis of the the final submission (see next section).

When looking at the number of submits to the abstract submission site, we can note that these are roughly four times of abstracts handed in. Further analysing when the submits happened, we can see that typical eCAADe submitter is:

- **Well-behaved**: The peak of the initial abstract submissions is *eight days before the initial abstract submission deadline* on 3rd February (there was an extended submission deadline on February 10th, which led people to pause for two days before resuming a steady stream of submissions).

- **Occupied during the week**: Most submissions happen on Mondays and Fridays between 12 und 16 hours UTC.

- **Expecting an extended abstract submission**: There are *as much submission in the extended deadline period as before*: this means that people really assume that there will be an extended abstract submission deadline, and use this to correct their paper.

These three points are of course a very subjective interpretation - the analysis of the final submission (see next section) tries to put some more scrutiny in so as to further narrow down what a conference organizer can expect, on a statistical basis.

**FULL PAPER SUBMISSION**

The following is a statistical overview of the full papers submitted for eCAADe 2014 (sidenote: 164 paper were initially accepted, 148 were present at the time of the extended submission and 127 remained in the final proceedings; either the authors failed to complete their work or withdrew their paper after submitting; the following data is a snapshot as per 18th June, two days after the extended deadline).

If ever there was a proof for a tendency to procrastination in academia it is shown in figure 2. The graph displays when authors started uploading their papers (positive y-axis) and when they finally submitted the paper (negative y-axis). The timeline along the x-axis starts at the beginning of April, which is when the first invitation to log-in to the system was sent out, passes the initial deadline (D) on the 9th of June and ends just after the extended deadline (ED) on the 16th of June. Work on the papers did not start to increase until about two weeks before the deadline, with a sizable portion not starting until after the
Figure 2
Timeline showing the dates the authors started and finished uploading their data. The initial deadline (D) and the extended deadline (ED) are overlaid.

Figure 3
Number of revisions and number of items per paper. Papers are sorted by the order work on them was started. The moving average of each measure is overlaid.
initial deadline had passed. It can also be seen that the email sent out on the 5th of June announcing the deadline extension caused a sudden drop in activity. Procrastination can be seen even more clearly with the paper submissions. Most waited until the extended deadline and some continued to work even beyond the deadline.

However, the graph in figure 3 clearly shows that the individual preferences in time planning have little impact on the length of the paper. Here papers are sorted by the date the authors started to upload their data and displayed along the x-axis. The positive y-axis gives the number of revisions, the number of times the authors made a change like inserting, changing or deleting data, on the negative y-axis the number of items (paragraphs, headers, images, etc.) is displayed as a measure of the overall length. Both measures are overlaid by their moving average over 20 papers. While there is a clear tendency to fewer revisions by papers started later, the overall number of items remains fairly constant.

Figure 4 shows the distribution of items over all papers. The items were grouped into content items (paragraphs, lists, formulas and algorithms), headers, graphical items (images and tables) and references. Papers were then divided into 16 categories according to their keywords and the item distribution was calculated for each category (Figure 5).

CONCLUSION AND OUTLOOK
The implementation of ProceeDings in the framework of eCAADe 2014 has delivered a treasure trove of experience which will be used for further developments. The tool was able to automate the production of eCAADe proceedings, starting from an initial list of accepted publications (coming from OpenConf) and ending with PDFs ready to be delivered to print production.

During the whole process, the editors still have an important role: As the technicalities of the paper layout are dealt with, they can direct their attention towards the content as such, i.e. focus on editing. In that context, ProceeDings allows to monitor ongoing developments within the community of submitting authors at an early stage, i.e. before the (final) submission and eventually to deliver feedback. Till a relatively late stage, authors can be involved in the publication process and the final outcome can be made available at any time.

The number of opportunities to (unintentionally) overrule the template is shrinking and especially ongoing live experience will accelerate the improvement of the interface. Most importantly, the users can take care of the layout themselves: You-Get-What-You-See (YGWYS) instead of You-Will-Sometimes-See-What You-Will-Get (YWSSWYWG). In this regard, the pre-publishing option (preprints) might gain interest. The extraction of (coherent!) metadata, required for indexing etc. has to be highlighted as well. It is unlikely that ProceeDings will encompass the abstract/paper submission and review stage. Here, a number of well-functioning (open source) environments has been made available for a longer period of time.

ACKNOWLEDGEMENTS
ProceeDings was programmed by Gabriel Wurzer over the course three years. Bob Martens took the tool into the eCAADe council, from which we got a first “go” in getting it so far that it would be ready for eCAADe 2015. Luckily, Emine Thompson gave us the chance to test-drive the system even earlier, as publication platform for the eCAADe 2014. We are extremely greatful for this decision, since it involved the risk that some functionalities would still be beta-quality. The paper importer proved to be
such a case, thanks goes to Marie Davidova and Martin Tamke for pointing out multiple issues. Likewise, Thomas Grasl and Rudi Stouffs gave their input on multiple features either missing or totally buggy, as did a multitude of authors through our bug reporting system. Thanks goes as well to Gregor Hartweger, Andrea Wölfer, Rudolf Scheuvens and Georg Penthor: The first three made it possible that we got our own server hosting the system (memory and CPU power: sufficient), and the latter one made sure we could relocate our whole set-up into a new building when the department was getting refurbished. From the editorial side, colleagues Wolfgang Lorenz and Gerda Hartl helped tremendously in layouting the proceedings. Speaking of layout, we also want to thank Henri Achten for showing us how to do this with style, based on his team’s great work with the eCAADe 2012 proceedings. Our main volume of thanks goes to all authors, who did most of the work: It is on their shoulders that we all stand. Last but not least, here is to you Martin Winchester, thank you for the OpenConf support which you do every year, and for the chance to pull an analysis of the abstract submission.

REFERENCES


Book: Agent-based Modeling and Simulation in Archaeology

Because of the organization of a conference on Agent-Based Simulation in Archaeology (AIA 2011, http://aia11.nhm-wien.ac.at/) and numerous publications on ABS of prehistoric mining in the Hallstatt region (Upper Austria), I was invited together with colleagues Kerstin Kowarik and Hans Reschreiter, both in the Natural History Museum Vienna (NHM), to compile a book on the topic for the Springer GIS Science series. The publication process, started in 2012, took two full years and led to a book encompassing a State-of-the-Art view on ABM in that context. For that book, I have also summarized the current state of simulation research in Hallstatt (together with Kerstin Kowarik and Hans Reschreiter) and given a characterization of ABS in Archaeology (together with Martin Bicher and Felix Breitenecker).

Citations.

Part II

Didactic Qualification
Teaching Experience

**Lectures.** Being a originally a computer scientist, I have been teaching programming and simulation for architects and urban planners since 2007, together with my colleague Wolfgang E. Lorenz who is an architect and has ever been my companion in science and teaching (see Table 8 for a summary; German course names have have been translated into English by myself).

Interestingly for me, recent work has been also in the field of agent-based design, i.e. the generation of form through ABS (together with Sigrun Swoboda and Christoph Degendorfer; see Figure 44), which is a novel field in architecture: Based on a 3D matrix of values (containing e.g. sun and shading in each cell, think: site analysis), agents walk through space and react to this context, building up form. The result is either performance-driven or form-driven, depending on the extent with which the agents react to their environment\(^8\). Such an approach might also bring previously not design-savvy people (to which I also count myself), such as mathematicians and computer scientists, into the architectural field.

![Figure 44: Agent-Based Design](image)


<table>
<thead>
<tr>
<th>Course Name</th>
<th>From</th>
<th>Until</th>
<th>At</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fundamentals in Computer Science</td>
<td>2007</td>
<td>2008</td>
<td>Vienna UT</td>
<td>Building Science</td>
</tr>
<tr>
<td>Build The Code - Programming for Architects</td>
<td>2009</td>
<td>2012</td>
<td>Vienna UT</td>
<td>Architecture</td>
</tr>
<tr>
<td>4D Simulation (Module 10)</td>
<td>2009</td>
<td>2011</td>
<td>Vienna UT</td>
<td>Urban Planning</td>
</tr>
<tr>
<td>Programming for Architects (Module Algorithmic Planning and Analysis)</td>
<td>2013</td>
<td>ongoing</td>
<td>Vienna UT</td>
<td>Architecture</td>
</tr>
<tr>
<td>Topics in Algorithmic Planning and Analysis (Module Algorithmic Planning and Analysis)</td>
<td>2013</td>
<td>ongoing</td>
<td>Vienna UT</td>
<td>Architecture</td>
</tr>
<tr>
<td>Spatial Process Modelling, Simulation and Visualization (Module 10)</td>
<td>2013</td>
<td>ongoing</td>
<td>Vienna UT</td>
<td>Urban Planning</td>
</tr>
<tr>
<td>Design Studio “How to be Posh” - generative design of flagship stores from IKEA to Prada</td>
<td>2013</td>
<td>2013</td>
<td>Vienna UT</td>
<td>Architecture</td>
</tr>
<tr>
<td>Design Studio “Brickster Style” - Design of a cultural center in Vienna Meidling</td>
<td>2014</td>
<td>2014</td>
<td>Vienna UT</td>
<td>Architecture</td>
</tr>
<tr>
<td>Design Studio “Flying Bricks” - Generative Brick Design</td>
<td>2014</td>
<td>2015</td>
<td>Vienna UT</td>
<td>Architecture</td>
</tr>
</tbody>
</table>

Table 8: Courses taught
Workshops. I have been invited to workshops in the context of programming and simulation for architects as well as simulation for archaeologists. The latter demand by archaeology is no surprise, since the social sciences have embraced the subject of ABS since a long time - assumably because of the ease of expressing artificial societies (see Figure 45) in a direct but nevertheless formalized way. Table 9 shows some of the workshops I held. It is notable that some of these led to further research, resulting for example in the paper “How to Teach Architects (Computer) Programming” which I have reprinted from p. 177 onwards.

<table>
<thead>
<tr>
<th>Workshop Name</th>
<th>Year</th>
<th>At</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the Code Workshop Istanbul: Introductory Programming (VBA)</td>
<td>2010</td>
<td>ITÜ Istanbul</td>
<td>Architecture</td>
</tr>
<tr>
<td>AIA 2011 - Simulation Workshop (NetLogo)</td>
<td>2011</td>
<td>Natural History Museum Vienna</td>
<td>Archaeology</td>
</tr>
<tr>
<td>Build the Code Workshop Bialystok: Introductory Programming (VBA)</td>
<td>2012</td>
<td>Bialystok Polytechnica</td>
<td>Architecture</td>
</tr>
<tr>
<td>eCAADe 2013 - Introductory Simulation Workshop for Early Design (NetLogo)</td>
<td>2013</td>
<td>CVUT Prague</td>
<td>Architecture</td>
</tr>
<tr>
<td>Topoi Winter School 2014 - Introductory Simulation Workshop (NetLogo)</td>
<td>2014</td>
<td>FU Berlin</td>
<td>Archaeology</td>
</tr>
<tr>
<td>eCAADe Regional Workshop - PacMan, meet Architecture (NetLogo, Agent-Based Design)</td>
<td>2014</td>
<td>Bialystok Polytechnica</td>
<td>Architecture</td>
</tr>
<tr>
<td>SCS 2014 - Artificial Societies Workshop</td>
<td>2014</td>
<td>UAB Barcelona</td>
<td>Archaeology</td>
</tr>
</tbody>
</table>

Table 9: Workshops held
Paper: How to Teach Architects (Computer) Programming - A Case Study

Since I am originally coming from computer science, I have been teaching computer programming for architects together with my colleague Wolfgang E. Lorenz, who is an architect. Apart from our own lecture “Build The Code” in Vienna, we have also been giving introductory workshops on the subject. During one of these workshops, given in the Istanbul Technical University (ITÜ), we have been sampling the progress of our students using mood charts, a tool for capturing not only the comprehensibility of our lecture but also the students’ personal feeling towards the presented material. Our interest was primarily on investigating at what pace a programming lecture has to progress in order to be “digestable”, if you will. The study was triggered by a previous conference contribution of our colleague Vassilis Bourdakis (of eCAADe), who was asking about “proper” ways to teach programming.

Authorship Information. The authors contributed equally to this paper. The accompanying implementation of Mood Charts was written by myself.

Main Contributions.

- Identifies topical areas in introductory programming for architects that need to be brought with ample amounts of time in order to be comprehended correctly (e.g. arrays, see p. 54 of the paper).
- Affirms the need for breaks and awareness of receptiveness at different times of the day when giving lectures.

Type of Work. Peer-reviewed paper, implementation of mood charts as a web application.

How to Teach Architects (Computer) Programming

A Case Study

Gabriel Wurzer¹, Sema Alaçam², Wolfgang Lorenz³
¹Vienna University of Technology ²Istanbul Technical University
¹{wurzer|lorenz}@iemar.tuwien.ac.at ²semosphere@gmail.com

Abstract. Computer programming in architecture seems to be commonplace throughout the eCAADe Community. Yet, a critical evaluation of a programming course as seen from a student’s side is still missing. During a week-long programming workshop in a fellow university, we have been assessing subjective parameters such as mood, quality of presentation and comprehensibility, comparing these to the actual topics that were covered at this instance. Our results contribute to understanding architecture students in their quest towards algorithmical thinking. We are convinced that the discussion given in this paper will help other teachers to further increase the quality of their lectures. Furthermore, the structure of our approach may serve as basis for further research into recording student behavior during programming courses.

Keywords. Teaching, Programming, Assessment.

INTRODUCTION

In recent years, the question of how to bring programming to architecture students has often led to uncertainty (Leitão, Cabecinhas and Martins, 2010, Bourdakis, 2010, Boeykens and Neucker-mans, 2009, Burry, Datta and Anson, 2000, Fricker, Wartmann and Hovestadt, 2008): As advocates of functional programming rally with those who prefer imperative programming, people who prefer one programming language with those that prefer another, and everyone is at odds with oneself on how to meet the goals being set forth for their course, there seems to be a strong wish for sharing insights on “how to teach architecture students programming” which has led to a variety of publications on the topic. One thing largely missing from the discussion is, however, a fresh look at the subject from a student’s side. In the course of a programming workshop, we have therefore been assessing the subjective impressions of students, and have compared these to the actual presented content. Being a case study, we see our contributions both in the conducted work (see “Case Study”) as well as the discussion of the results as actual consideration points which we believe can help fellow lecturers improve their courses (see “Discussion”). Given the perspective that we would like more teachers to conduct student-centered assessments, we also bring forward a format which we believe can serve as practical assessment basis, in the form of questionnaires to be handed to the students (keyed by us as mood charts).
RELATED WORK
During the last decade, it has been discussed how far programming courses support the architecture students’ way of thinking. Researchers state that it improves understanding of descriptive geometry when it comes to representations and relations between the entities (Bourdakis, 2010, Burry, Datta and Anson, 2000, LaBelle, Nembrini and Huang, 2009, Kajima and Panagiotis, 2008). Typically, programming in the architecture domain is performed by writing code (e.g. Visual Basic for Applications, ANAR+ for processing). However, visual programming approaches that promise to fit the graphical mindset of aspiring architects better are also gaining ground, e.g. Bentley’s Generative Component and Grasshopper for McNeel Rhino 3D. The reason for this is reported to be that algorithmical thinking requires a problem-centered approach, while architecture usually takes a solution-centered strategy (Lawson, 2005).

In our case study, we focus on programming by writing code, targeting the question of teaching quality as seen by architecture students in a workshop environment, using a questionnaire-based approach as method. To the best of our knowledge, this has so far not been conducted, although we can find examples of qualitative thoughts for example in (Knight and Brown, 2010). In detail, we will focus on the different subjects to be covered in such a course, relating the sequence given and the time spent on each subject to the comprehension and perceived quality of presentation. It is safe to say that our report is not generalizeable (given the number of participating students as well as the form of survey); we see our contribution as a pre-study, on which further work can be based. Therefore, we also cover the shortcomings and address fields where more effort should be invested when performing a full-fledged study (see “Case Study”).

WORKSHOP DESCRIPTION
The programming workshop was given in November 2010 at the architecture faculty of Istanbul Technical University over the duration of five days. The content of the course was on programming using Visual Basic for Applications in AutoCAD for the sake of form generation, i.e.:
- To learn basic constructs of an imperative programming language and use them to generate geometry.
- To develop algorithmical thinking by solving a programming assignment in a studio-like context, in order to learn how to generate form and automate daily CAAD work routines.

There were 18 students attending, half of which already had or were in the process of being in a basic course on PASCAL. As lecture material, a specially-produced booklet covering all presented topics was given to the students, allowing them to revise the subjects taught and catch up once needing to be absent for some hours. Furthermore, a description of the programming assignment was handed out. The chosen topic was on automated building using a grid-based house-generator, which would call up each student group’s program in order to build a house.

Due to the constrained nature of our schedule, it was decided to give the introduction to programming in only two days, after which practical work on the programming assignment would begin. For a programming lecture, this rather unusual approach boils down to imposing a very intensive work program in a short time frame, and clarifying topics still unclear to a student in a face-to-face manner, in parallel to actual work being already done by all others.

The topics covered were the same ones that are usually given in a standard programming lecture. However, to meet the time constraints, care was taken to bring two or more topics at the same time (e.g. learning the basic program structure while getting introduced to debugging):
- Debugging a “Hello World” program, variables (only numbers at first), functions with and without return value, parameters, calculations with numbers, nesting of calls
- AutoCAD drawing library (using Object-Oriented Programming without actually saying so), Booleans and conditions, arrays (static and dynamic)
• Implementation of a staircase generation program using loops, creation of a user interface to enter parameters for the staircase program, strings and string operations, introduction to Object-Oriented Programming

The final goal and motivation of the workshop was to present the results of the programming assignment to the dean of Istanbul Technical University’s architecture faculty, as well as to conduct the case study on perceived quality of each topic given.

CASE STUDY
The survey on teaching perception was conducted using a questionnaire-based approach. At the beginning of each day, the students were given a sheet in which they could chart general mood, comprehension of the presented material and perceived quality of presentation over time, using a five-fold scale (excellent, good, fair, satisfactory, poor). In order to emphasize deviations from the average case (‘fair’). The survey sheets, also keyed as mood charts, were completely anonymous - participation was conducted on a voluntary basis. Students were told to sample at the end of each topic, and encouraged to take additional samples as they saw fit. Furthermore, the noting of sentiments on the graph was explicitly welcomed (e.g. “11:15 - boring!”), although very seldom seen. At the end of the day, mood charts could be thrown into a bin (filled or unfilled). The results were then analyzed and interpreted by correlating the presented topics (of which time and duration was known) to the mood, quality of presentation and comprehension at the given instance (see Figure 1).

Figure 1
Analysis of „mood charts” created during the introductory programming course. Mood, comprehension and quality of presentation are charted over time. The covered topics are furthermore entered at the time of their appearance. The results clearly show a change in all supplied measurements from very excellent marks in the morning (color-coded as green) to poor marks in the afternoon (red).
For the three days of programming that followed the two-day introductory course, only mood was measured at discrete times (morning, midday, evening). The reason behind this was that, since all students had to work individually, there was no presentation given and therefore also no comprehension to measure. The assignment itself was presented at the start of day three, implemented on day 4 and the outcomes presented on day 5.

**Observed attitude towards the presented material during the workshop**

The first half of day 1 was occupied with “diving in” and getting fully focused on programming. Presentation was seen excellent, comprehension was generally good. However, as can be seen from the mood chart, the rapid flow of information lead to a steady decline in mood, from good (morning) to fair (midday). We see this as being related to the presentation of functions and parameters, which was given using the concept of “Input-Process-Output” (IPO) and then mapped to either function (with output into a variable) or procedure (without output). As mental bridge, a relation to visual programming using flowcharts and Rhino Grasshopper was given, which worked remarkably well. The relation to mathematics (e.g. function sin(x)), however, did not have a beneficial contribution to the understanding.

After lunch break, the mood returned to normal values. The presentation of arrays was hard to digest, it seems, and thus led to a dramatic decrease in comprehension and to a decline of mood in the afternoon. As soon as practical work with arrays using the AutoCAD drawing library was done,

![Mood charts for Day 3, Day 4, and Day 5](image)

**Figure 2**
Upper part: Mood during the programming assignment. Even though the point samples at morning, midday and evening are of limited significance, one can see that the trend in Day 4 is significantly different than any of the other days in the sense that the mood constantly gets better. Lower part: (left) Photo taken during workshop. (right) Example of a grid-based house generation algorithm used in the assignment.
the mood settled on an average level. The introduction of multi-dimensional arrays which followed immediately afterwards without giving practical examples was seen as inadequate, with mood and comprehension decreasing to poor at around 4pm. The last lecturing hour, in which Booleans and conditions were to be introduced, was not digestible any more. Therefore, only an overview followed by the day’s summary was given, with the intention of deepening the knowledge in a recapitulation in the beginning of the next day.

Day 2 started with practical exercises (drawing geometry), in which conditions and Booleans were brought in. This approach worked remarkably well, with all measured parameters being excellent until midday. The introduction of dynamic (i.e. growable) arrays led to a mood shift at around 12am, with the presented content being generally understood, but not well-received due to the lack of benefits for the program written. After lunch break, the introduction of loops required the re-writing of a previously written program and led to a drop in all sampled values. Either the adaptation of a program was not sufficiently presented, or the introduction of new content was not adequate. Between 3pm and 4pm, an overview over Forms and Strings was given. Even though this subject was, in our view, already adapted to the declining attention and mood, it was nevertheless not well received, with mood values being at their minimal peak of the day. This afternoon decline leading to a no-go situation at 4pm is to be investigated in further courses. After a 15-minutes break, a practical example on forms as well as the end of the day being in close reach, the mood values increased again.

In day 3, we did not gather enough measurements in order to be able to discuss them here. From our observations, we saw a generally bad mood, connected to the rapid pace of the previous days, which we had to amend by giving individual help to the students, leading to a significant improvement over day 4. In day 5, we observed a decline in mood due to the hand-in stress.

Lessons learned
From our observations during the workshop, there are several points that are worth consideration in subsequent courses:

• The explanation of functions with and without parameters, with and without return value benefits from a clear analogy to an IPO metaphor, if which grasshopper is an excellent visual example.

• Simple data types (strings, Booleans, numbers) can be handled in a straightforward manner. Arrays, however, are hard to explain - especially in the multidimensional case. It is probably hard to imagine the necessity for collections of values, which would otherwise have been taught in a course dealing with algorithms and data structures (definitely also a point for further emphasis in a programming lecture for architects). Objects are naturally observed as “packages of functions and variables”, with no further explanation needed. This might be as to the nature of VBA, which exposes all functionality of AutoCAD as objects.

• As opposed to a bottom-up lesson, which tries to present all topics independently, we have experienced that is does sometimes make sense to present a seemingly complex topic right away and then decompose it into its constituent parts. In this way we were able to present many concepts in parallel, without losing reference to the overall picture.

Comparing to a full-fledged study
We are aware that the number of participants in our study is statistically insignificant. The results nevertheless reflect, to a large extent, our own insights gained through years-long teaching practice, and can be used as a guidance when setting up a larger study. Points worth considering when doing so are:

• The prior knowledge of the students was not asked in a structured manner (rather by raising hands). It might be important to sample, for example, who has had experience in each of the presented topics
before actually beginning the course.
• For comparison reasons, comparison groups will have to be set up.
• A fivefold scale offers the opportunity to state average levels. It might be interesting to exclude this possibility, thus forcing the student to be explicit.
• The assumption that each topic can be sampled immediately afterwards (i.e. the topic as cause, the comprehension as effect) is naïve. It might be interesting to see when each topic has settled (using questions targeted not only at the last topic given, but at a variety of topics).
• As side note, we must mention that the students were of different interests and knowledge concerning programming, which led to segregation. It might be easier to hold a workshop in an environment where all students have the same knowledge (e.g. by the way of establishing programming in a fixed place in the curriculum).

DISCUSSION
During the teaching process of computer programming in architectural design education, the feedback from the students, their motivations and their learning curves should be taken into consideration by the lecturers. There should be more explorative studies about what kind of strategies in teaching computer programming are crucial for introductory courses.

From our collected data, we can see a direct connection between mood and comprehension, while the quality of presentation is separate and declines slightly from the morning to the evening. Topics that are new or hard to digest must be therefore brought in the morning. If this is not possible, more time than usual has to be invested in the afternoons. In general, recapitulations and exercises are of utmost importance, due to the nature of architectural work (solution-centered). However, the format of a workshop might not always allow for such deepening of knowledge, because of time constraints. Therefore, establishing an educational scheme that covers all programming topics while at the same time being time-efficient and qualitatively useful as seen from a student’s side is a field worth researching in.

REFERENCES
Bourdakis, V 2010 ‘Designing Interactions: A step forward from time based media and synthetic space design in architectural education’, Proceedings of eCAADe 2010, Zurich, Switzerland, pp. 151-156.
Part III

Appendix
**Projects**

*Even though* not counting towards my academical or didactic qualification, my work has always been done in the context of projects, some of which where even didactic in nature. Being half-time employed in the university and the other half in the healthcare industry (more specifically eHealth; until 2012) sometimes gave me the opportunity to mix concepts from both fields - which is for example where the idea to use Hospital Information Systems as data source for simulation came from.

*Early-Stage Simulation Landesklinikum Wiener Neustadt*

LK Wiener Neustadt, an 800-bed clinic in Lower Austria, is to be rebuilt some time in the future (no official starting date has still been named). However, before that happens, the client wanted us to produce a simulation within the schema he had produced in interviews and workshops, in order to (1.) be able to visualize treatment chains occurring in the current hospital being executed in an envisioned hospital, (2.) have a tool with which simple capacity planning can be made, as input for a tender, (3.) be able to present the building concept in animated form to a wider public. In order to satisfy these requirements, we have written an own ABS in Visio (or Visual Basic for Applications, to be precise), which has the advantage that simulation results can be saved together with the schema and, furthermore, can be printed at arbitrary resolutions. The project was a huge success and was presented at press conferences and disseminated as DVD to all stakeholders, which is especially encouraging.

**Duration.** 2 Months (starting second quarter 2012).

**Client.** Landesklinikenholding Niederösterreich, LK Wiener Neustadt  
**Partners.** medipro (lead), Vienna UT

*STABLE AIR*

As part of an effort to bring leading-edge science into the industry, the Austrian Research Promotion Agency (FFG) financed a *teaching project* in which the scientific team presented simulation techniques
for airports. The project was not aimed at producing a fully-fledged simulation, however, but to foster interest in the subject and to offer both scientists and industry partners an initial incentive for getting into touch with each other.

**Duration.** 6 Months (starting second quarter 2013).

**Partners.** DWH Simulation Services (lead, research and industry), Vienna UT (research), Destion (industry), AIMS (industry)

*STABLE HBO*

Similarly to STABLE AIR, a teaching project for hospital simulation was financed by the FFG.

**Duration.** 6 Months (starting third quarter 2013).

**Partners.** DWH Simulation Services (lead, research and industry), Vienna UT (research), SOLVE Consulting (industry), Wolfgang Bayer Consulting (industry)

*MONARCA*

In my job as e-Health consultant for Systema Human Information Services, I have been working on a EU FP7 project named MONARCA - MONitoring, treAtment and pRediCtion of bipolAr Disorder Episodes. The project was occupied with assessing the current state of a bipolar patient via cellphone sensors (e.g. movement sensor, location), to process this data and to come up with predictions on the development of his or her episode (leading to suggestions being displayed, messages sent to the advising psychiatrist and so on). My role in the project was the modeling of the
high-level architecture (systems and their connections) as well as the integration with Electronic Health Records (EHRs).

**Duration.** 3 Years (starting February 2010).

**Partners.** CREATE-NET (lead), Aipermon, BITZ, ETH Zurich, IT University of Copenhagen, Meditrainment, RegionH Psychiatry Denmark, Psychiatric State Hospital Hall in Tirol, Scuola Universitaria Professionale della Svizzera Italiana, Systema Human Information System, Universität Bielefeld, Universität Passau, Universität Kaiserslautern.

**MODYPLAN (ongoing)**

In its “Smart City 2012” call, the City of Vienna’s Center for Innovation and Technology (ZIT) granted us a project for producing a planning in which preliminary hospitals can be designed as schemata and simulated for capacity planning. As goal, the project sets out to integrate different clinics and simulate the integrated concept based on treatment chains.

Integration within a clinical region has been an ongoing in the Vienna area, which has recently drawn together its resources in a newly-built “Krankenhaus Nord” (North Hospital; see Figure 48). Likewise, we want to be able to outsource or integrate different hospital areas (technically: zones) for being able to simulate this shift. The project is still ongoing until end of February 2015, which is when we will release this simulation as commercial application (see Figure 49). Instead of bringing a more elaborate description, I include a descriptive paper (p. 191) submitted for MathMod 2015 (accepted 10.01.2015 but not published, and hence not counting for...
the evaluation of the habilitation).

**Duration.** 3 Years (started February 2012).

**Partners.** DWH Simulation Services (lead), Vienna UT.

Figure 49: MODYPLAN: Schema editor and ABS of treatment chains.
**Paper: MODYPLAN: Early-Stage Hospital Simulation**

Instead of a project description, I here reprint the accepted version of a paper which we sent to MathMod 2015 (due to be published end of February). The paper argues for the embedding of the agents’ control logic into zones, showing the exact pseudocode that replicates our application. Quoting the paper (p. 3), the reason for putting the control logic into the different zones is “that there are a variety of options to choose from. For example, the [zone] can have a waiting [sub-]zone where the queuing actually takes place. It could have a prioritization (e.g. Manchester Triage) and so on”.

**Authorship Information.** All authors contributed equally (directly or indirectly) in this work: I compiled all previous papers and inputs given during the project into this contribution, so one could say that this is the “overall summary” of our MODYPLAN implementation.

**Main Contributions.**

- Presents a simulation model in which the agents’ goals take over control once being reached. This approach makes it easy to model complex behavior inside zones, while leaving the agent description be a simple treatment chain.

- Provides extensive pseudocode in order to aid re-implementation (p. 3 of the paper)

- Discusses benefits and observed drawbacks of our approach (p. 5 of the paper).

**Type of Work.** Peer-reviewed paper, implementation as commercial program MODYPLAN.

**Citation.** Wurzer, G., Lorenz, W., Rößler, M., Hafner, I., Glock, B., Bruckner, M., and Popper, N. (in press). MODYPLAN: Early-Stage Hospital Simulation based on Treatment Chains, *Proceedings of MathMod 2015*, Vienna, 6 pages
**MODYPLAN: Early-Stage Hospital Simulation based on Treatment Chains**

Gabriel Wurzer*, Wolfgang E. Lorenz.*

*Vienna University of Technology, Architectural Sciences (E259), Vienna 1040, Austria (Tel: 01-58801-27201; e-mail: <firstname.surname>@tuwien.ac.at).
**DWH Simulation Services, Vienna 1070, Austria (e-mail: <firstname.surname>@dwh.at) and Vienna University of Technology, Analysis and Scientific Computing (E101), Vienna 1040, Austria (Tel: 01-58801-10115; e-mail: <firstname.surname>@tuwien.ac.at).

Abstract: Discrete-event simulation of hospitals typically specifies flow by means of a process graph through which patients are routed. While this is generally fine for models in which processes are clearly defined, e.g. smaller units such as emergency departments, it falls short of capturing the fact that a patient can in principle go from one unit to any other unit if medical procedure requires it. This problem becomes even more evident when modeling a whole hospital, at which level specifying each individual treatment through directed edges and nodes becomes unviably complex. In the past three years, we have been developing a combined hospital simulation/space design tool in which processes are defined individually by patient, as sequence of visited units imported from the Hospital Information System (treatment chains). The major advantage of this approach lies in the fact that we can now design a future space layout in which assumed capacities (staffing, equipment, required space) can be tested against the recorded patient volume, using an Agent-Based Simulation to re-enact each treatment chain. In contrast to other hospital simulations, this approach targets early stages of architectural conception, during which the actual building structure is elaborated. Using the method, we can compare and contrast different layouts during that stage, making simulation a driver for design rather than a tool for late optimization within the final floor plan.

Keywords: Hospital planning, design tool, treatment chain, Agent-Based Simulation, space layout planning.

1. INTRODUCTION

In this paper, we draw together the lessons learned during the development of our combined hospital simulation/space design suite called *Modular Dynamic Planning and Simulation for Health Care Facilities*, (MODYPLAN) which is based upon the idea of schedule-based simulation (see section 2) using the actual treatment data of a clinic, even though in anonymized form. One of the main realizations leading to the development of the tool was that the hospitals are not conceived from scratch - they are based on pre-existing spatial templates (e.g. Neufert 2000) which are adapted to fit the expected patient volume and specialization. There are, nevertheless, a variety of possible options to choose from, leading to the questions of “How to compare two spatial layouts?”, which is where simulation comes into play: By letting agents simulate their individual schedule (coined as treatment chain, see section 3), they utilize spaces which have a finite capacity. By recording this utilization, space occupancy and agent histories among different layouts, the program can compare the “performance” of each of them.

From a project view, we have just finished the core implementation and are currently modeling a hospital as a test case. Section 4 draws together some insights from this process as well as on the visualization which we are currently completing.

2. RELATED WORK

Schedule-based simulation has been used for some time to simulate occupant behavior in buildings, mostly for obtaining the predicted energy demand. For example, Page (2007) models the presence of building users in each space and correlates that through a set of stochastic models to each individual type of energy-consuming activity (e.g. light switching, window opening). Tabak (2008) concentrates on surveying and classifying office tasks in order to be able to accurately model them. One of the advantages of his approach is that it takes interactions between agents into account (e.g. shared activities such as meetings). Goldstein et al. (2010) generate fictional schedules that reproduce recorded activities (the authors call this “schedule-calibrated” simulation). In more detail, the authors present a stateful approach where each activity can be based upon past behavior. This is to some extent comparable to having a memory within the simulated entity (i.e. in a token, process execution or agent) to base decisions on.

Schedule-based simulation with special regards to the clinical domain has been addressed to a much lesser extent. On a regional level, Miksch et al. (2014) have presented an Agent-Based Simulation framework for simulating medical service provision in the context of epidemiology. However, their model is purely abstract, with no reference to a space layout. Likewise, Einzinger et al. (2014) have simulated whole
health care systems using routine health care data from reimbursement claims as a basis. On the local level of an individual clinic, we are so far the only group that has been doing schedule-based simulation, to the best of our knowledge (Wurzer, Lorenz and Pferzinger 2012).

Typically, hospital simulations use process notations (e.g. Simeone et al. 2012) or other forms of flow graphs (Friesen and McLeod 2014) to supply the logic behind the simulated patient flow. They are applied in late stages of planning, i.e. within the finished floor-plans, for detailing of processes and quantification of resources. By contrast, our method is applied in early phases of hospital planning, for establishing and comparing several preliminary space layouts (of which several variations may exist, see Wurzer 2013). To be fair, these space layouts are still very approximate, and thus running simulations in them merely gives qualitative rather than quantitative results (Wurzer and Lorenz 2013). Nevertheless, applying simulation early has the advantage that the design can be changed easily, in order to avoid planning errors.

3. PROPOSED METHOD

Our method is based on treatment chains (figure 1a), i.e. sequences of functional units that a patient visits during his clinical encounter. Each Functional Unit (Figure 1b) is a capacity-constrained resource which receives these visits (immediately or after queuing). The service duration is given in the treatment chain (e.g. in figure 1a: 4 time units in Neurosurgery followed by 30 time units in Rehab).

Functional Units have named purpose (“function”, e.g. Examination), a boundary (closed polyline) and can be nested, thus forming a spatial hierarchy. They are the elements of the space layout, i.e. named spaces augmented by their role as resource. On a regional level, we may model each clinic as Functional Unit having its departments as sub-units (e.g. in figure 1: sub-units Neurosurgery, Orthopedics and General Surgery in unit North Tyneside, Neurehabilitation and Neuropsychiatry in Walkergate Park).

![Fig. 1. (a) Treatment chains as list of Functional Units to visit. Durations spent in each Functional Unit are given in parentheses. (b) Functional Units as resources having a finite capacity, after which a queue is formed.](image)

For each of the functional units, utilization and occupancy of its underlying space is recorded over time (the latter through a raster of a given resolution, e.g. 0.5m). Each agent, acting on its individual treatment chain, furthermore records durations spent queuing, receiving service and spending time transitioning. Functional Units furthermore directing the agents furthermore write the intermediate goals into the agents’ histories. All of these values are later used in the comparison of different planning scenarios (see section 4.2), i.e. different configurations of Functional Units in which the same treatment chains are simulated.

3.1 Processes moved into agents

If we consider a process simulation based on nodes and directed edges, we have a class structure as shown in figure 2a: A process is specified by nodes and connecting edges, or more precisely: Subclasses of Node which have a specified behavior, as for example a DecisionNode that chooses an outgoing edge for the process Execution, possibly based upon data that is contained in Execution:context (think of this as a key-value container).

Given this structure, the previously stated problem (patients sent from one unit to any other unit if required by treatment) maps to the following two issues:

1. In order to model processes realistically, one would need to introduce additional edges that lead to nodes invoking a process in another unit of the hospital. Referral processes such as ‘emergency treatment’ will hence fan out to include every possible department. Likewise, the department processes must also introduce edges for back-referral.

2. Within each process changed in this respect, the decision logic must now take the additional edges into account. There are two places in the class structure that are affected, first: the data contained in Execution:context and second: the node’s decision logic called upon by execute().

![Fig. 2. Class diagram for (a) process simulation (b) agent-based simulation based on treatment chains.](image)

Both issues make the processes difficult to read: Additional edges clutter the process description; additional logic that concerns another process is hard to understand. Furthermore, process descriptions are cumbersome to change: In the worst case, one has to go through each process to see whether it is affected. Such changes are, however, very frequent in early
planning, where hospital units are created, split, merged and so forth.

“What if one could free the process of explicit transitions and shift this responsibility to another entity?”, was our primary thought when looking at this problem. In an agent-based simulation, the Execution becomes an Agent (see figure 2b). It is typical that each agent has a list of consecutive goals to which it moves, where each of the goals carries some additional significance (e.g. waypoint, service point where interaction happens). In our model, the treatment chain becomes this goal list. Furthermore, each goal references either a FunctionalUnit, (a Space with a polyline boundary which is also a capacity-limited Resource within the simulation) or a Waypoint (a point, not shown in figure 2b).

Each Agent starts at the first goal, which is typically the entrance of a hospital. In case this entrance is not present in the chain, one can always prepend it (preprocessing). Goals coming after that are the actual visited Functional Units, except the last one, which is the exit. Below algorithms give the functions used by agents to process their chain:

```javascript
function Agent:start() {
    if goals not empty {
        fetch next goal and duration
        position = center of goal
    }
}

function Agent:simulate() {
    if controller is set {
        controller.move(myself)
    } else {
        myself.move()
    }
}

function Agent:move() {
    if goals is empty {
        remove myself from simulation
    } else {
        if goal refers to a FunctionalUnit {
            fu = get FunctionalUnit for goal
            if inside fu.boundary {
                fu.handleAgent(myself)
            } else {
                moveTowards(fu.center)
            }
        } else { // goal is a Waypoint
            wp = get Waypoint for goal
            if position is wp.center {
                wait for Agent.duration
            } else {
                moveTowards(wp.center)
            }
        }
    }
}
```

The agent always starts at the center of the first goal and moves, with each step of the simulation, towards the of the next goal. In case the goal is a waypoint, then the agent moves to its center and fetches the next goal. If it refers to a FunctionalUnit, we first move the agent until it reaches the boundary and hand control over via the `FunctionalUnit:handleAgent(Agent)` function, which resets the state of the agent and sets its controller to the FunctionalUnit’s Behavior, which from now on governs the movement of the agent:

```javascript
function FunctionalUnit:handle (Agent) {
    agent.state = INITIAL STATE
    agent.controller = my behavior
}
```

The default behavior, Queue, lets the agent queue at the boundary of the Functional Unit until it is free. Then, it moves the agent to its center and lets it wait for the given duration. As convention, this is always 0 for entrances and exits (first goal and last goal of a treatment chain), since we want arrivals and exits to happen instantaneously. As last step, the Agent’s controller is cleared and it resumes normal operation using the next goal:

```javascript
function Queue:move(Agent) {
    fu = get FunctionalUnit for myself
    if Agent.state is INITIAL STATE {
        fu.reserveFor(Agent)
        Agent.state = ACQUIRED RESOURCE
    } else {
        if Agent.position is fu.center {
            wait for Agent.duration
            fu.releaseBy(Agent)
            clear Agent.controller
            Agent.fetch next goal and duration
        } else {
            moveTowards(fu.center)
        }
    }
}
```

The reason for outsourcing the Functional Unit’s behavior is that there are a variety of options to choose from. For example, the Functional Unit can have a waiting zone where the queuing actually takes place. It could have a prioritization (e.g. Manchester Triage) and so on, which are scenarios that are hard to describe for the general case but easy to implement in an own sub-class of Behavior which pulls the strings.

![Fig. 3. Control logic of the method: An Agent moves until it reaches its goal FunctionalUnit, at which spot the Functional Unit’s Behavior takes over until resource utilization has been simulated.](image-url)
in control. By entering the Functional Unit, control passes to its Behavior until the agent’s utilization of the Functional Unit has been fully simulated. The nice thing about this approach is that Function Units can now be characterized independently of their process, i.e. have agents say what is utilized and have spaces state how this is happening. The resulting design may be saved as a template (Christiansen and Bruce 2008), to be reused in other designs that are subject to different treatment chains.

3.2 Functional Unit Hierarchies

Functional Units are composed into a hierarchical space layout where each sub-unit is completely contained in its parent unit. The hierarchy enables us to specify capacities either implicitly or explicitly (refer to Figure 4a):

- Implicit Functional Units specify a capacity greater than one, signifying that there are multiple resources that can be served simultaneously (left part of Figure 4a). Often though, planners want to see these resources explicitly - as they mostly represent spaces. Therefore (see right part of Figure 4a), users can place k Functional Units with the same function, which is the same as specifying one Functional Unit of capacity k. The difference between the two options is that, visually, the agents move to the center of each Functional Unit, and hence the sub-units can be thought of as individual “rooms” serving a common purpose. As caveat, the implementation of FunctionalUnit:getBoundary() must either return its actual boundary when capacities are implicitly stated, or the parent boundary for the explicit case. The reason is that the agent has to stop at some point at the border of a Functional Unit, which is the parent’s boundary in that case.

- At run-time, each agent must resolve the name of its goal so that it can get the corresponding Functional Unit. For example (see Figure 4b), consider a clinic having an Emergency Department (ED) and a Neurology (N), both with sub-units for Examination (EX). If an agent’s treatment chain contains “EX”, then name resolution will give different results depending on where the agent stands: We first look at sub-spaces, then consider the higher hierarchical levels until a Functional Unit with the correct function has been found (in all other cases, we raise an error). For Neurology, this would choose the examination room in that area, while the Emergency Department would give its examination room to the agent. If it is required to switch between departments, one must prepend the department name (here: N > EX > ED > EX).

3.3 Comparing different layouts

In order to compare different layouts, we rely on three types of data:

- The utilization of each Functional Unit (see left in figure 5a) leading to an utilization graph (right in figure 5a), e.g. as plot over time or as piechart showing the share of time that the Functional Unit was overutilized.

- The occupancy of each Functional Unit (left in figure 5b), increased when an agent enters the boundary. This can be shown e.g. absolute or according to the increase per hour (see right in figure 5b).

- Agent histories record the time spent utilizing, transitioning and queuing (left in figure 5c). Additionally, each agent has a specified type (e.g. trauma patient, regular patient). Histories can be grouped by that type, in order to arrive at the maximum queuing times before Functional Units or the maximal treatments.

In order to compare different layouts, we use side-by-side view in which Functional Units can be dragged in from two different layout trees (refer to figure 6): The left tree depicts the current layout, the right a comparison layout. The user can choose to drag in units from the same tree to also perform a comparison in one layout.
4. DISCUSSION

The presented simulation has been implemented as software and is currently being tested with sample clinics. We herein wish to share some interesting observations that we have made during that phase.

4.1 Importing Data

Treatment chains are imported from Hospital Information Systems, or rather: they are exported as spreadsheet and interpreted by our application. An example for this is given in table 1: Here we have two anonymized patients given by id 1234 (a trauma patient) and 2345 (a regular patient). The first has the treatment chain [Entrance]>1ED2ADM(1 minute)>CXRAY(0 minutes)>[Exit] and the second one [Entrance]>1ED2ADM(0 minutes)>[Exit]. The problem is here that we have only limited information concerning the durations spent in both cases. We know for sure that the patient 1234 has spent 1 minute in 1ED2ADM (which is the Emergency Departments second admission counter), but we do not know how long he was in CXRAY (central radiology), since there is no following timestamp on which to base durations on. The same is true for the second patient, who has only one goal in his/her chain.

**Table 1. Sample data in a typical format**

<table>
<thead>
<tr>
<th>timestamp</th>
<th>id</th>
<th>unit</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-09-01</td>
<td>1234</td>
<td>1ED2ADM</td>
<td>TRAUMA</td>
</tr>
<tr>
<td>13:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014-09-01</td>
<td>2345</td>
<td>1ED2ADM</td>
<td>REGULAR</td>
</tr>
<tr>
<td>13:31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014-09-01</td>
<td>1234</td>
<td>CXRAY</td>
<td>TRAUMA</td>
</tr>
<tr>
<td>13:32</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We have learned that this is not the exception but the rule for such data. In order to compensate these “zero durations”, we have included a field in FunctionalUnit that gives the typical average duration of service, which is used in such a case. Also, the duration between two timestamps may include waiting times; we have included a Boolean override duration flag in each Functional Unit, for cases where this is typical (classical example: x-ray examinations take anything between 3 to 5 minutes, but waiting times may take hours).

4.2 Comparison between two layouts

We have designed different visualizations for being able to depict the recorded simulation histories stated earlier (see section 3.3). The design options were shown to planners, in order to have an expert opinion on their value and be able to implement one option fully. We are currently in the phase where we have completed several of these interviews, and wish to share some preliminary results of these.

At first, we had an “overall” comparison that computed a score for the topmost Functional Units, taking all subunits into consideration. This was, however, not informative, since planners want to be able to “drill into” the data and consider sub-units individually. Also, the expectation of always comparing two different layouts was wrong, surprisingly, as planners also wanted to compare the performance of functional units of the same layout. Thus, we now aim at giving both options in one common comparison interface (sketched in Figure 6).

We also found that individual treatment chains were of less interest; what is considered most important are the spaces and their utilization (queues!), followed by occupancy. An extension to visualization of causalities (think: who traveled through a specified Functional Unit, where was he before and where did he go afterwards), in line with Wurzer and Lorenz (2014) is being considered at the moment.

4.3 Relationship with other tools on the market

Our tool is insofar unprecedented as there is currently no simulation based on treatment chains, and no simulation serving as driver for preliminary design (i.e. before and after the tendering phase of an architectural project. Some examples of competing applications are: Programs that offer “sketch-like” Computer Aided Design functionalities (e.g. the ONUMA System [http://www.onuma.com] which provides a web-based editing environment that can be used for early design) or discrete simulation packages that can be used for modeling passages between hospital service units (e.g. FlexSim Healthcare [http://healthcare.flexsim.com]). However, both types of systems do not form a closed cycle - one can import the spatial layout into a simulation but the reverse way of using the simulation results (e.g. space utilization and occupancy for determination of space sizes) is not possible. This is where we see the value of our own software.

5. CONCLUSIONS

We have presented a schedule-based hospital simulation method which relies on treatment chains stemming from a Hospital Information System. In contrast to classical process-based simulation, we have no notion of edges; the transitioning between nodes is rather based on an agent’s prescribed schedule, making it easy to capture the real flow between different units, departments or even hospitals. The approach has been implemented as our own simulation suite, ((MODYPLAN)), which we will bring to the market in spring of next year.
ACKNOWLEDGEMENTS

This paper was financed by the ZIT MODYPLAN project, under the call Smart Vienna 2012.

REFERENCES


www.healthcaredesignmagazine.com/article/template-change [accessed 1st October 2014]


Appendix A. AUTHORS BIOGRAPHIES

Gabriel Wurzer has done his Ph. D. on Process Visualization and Simulation for Hospital Planning (Vienna UT, 2011). His research in architectural sciences focuses on tool support for early-stage planning of complex buildings.

Wolfgang Lorenz has a Ph. D. degree in Architecture for his thesis on Analysis of Fractal Architecture using the Box-Counting Method (Vienna UT, 2014). He is lecturer on programming for architects and simulation.

Matthias Rößler received his Master’s degree on Coupling of Thermodynamical Systems (Vienna UT, 2012) and is currently working on simulation models for DWH Simulation Services as well as on clinical information systems.

Irene Hafner has done her Master’s degree on Co-Simulation using the Building Controls Virtual Test-Bed. Her current research concerns cooperative and multi-rate simulation, healthcare and production planning.

Barbara Glock has gotten her Master on A System Dynamics Model of the Prevalance of Obesity in Austria (Vienna UT, 2014). Her current work is in the field of coupling different modelling methods.

Martin Bruckner is a specialist on Agent-Based Modeling currently working for DWH Simulation services. He has completed his PhD in that same subject and is currently in the way of defending it before Vienna UT.

Niki Popper has a degree in Mathematics and is CEO of DWH Simulation Services, a professional simulation company based in Vienna. He is also lecturer in mathematical simulation at Vienna UT.
**Activity**  An single observable event within a behavior or process of an individual.

**Agent-Based Simulation (ABS)**  A simulation type in which individuals (persons, material goods, particles and so forth) are modeled by simple rules concerning their movements and decisions taken. Typically, these agents act in a spatial environment (e.g. a floor plan, schema, city map) which they take into account when moving and deciding (interaction agent - environment). A further interaction type in ABS is between agents, e.g. the infection process in epidemic models. Apart from rules for agents, it is also common that there are rules governing the environment, typically defined as set of square cells in which the agents move (see Cellular Automata).

**Architecture, Engineering and Construction (AEC) Sector**  Synonymous for the architecture industry, encompassing the whole project cycle from planning until completion of a building.

**Adjacency Matrix**  A matrix containing adjacencies between functions or functional units, e.g. distant, neutral, far, not applicable. Adjacency are a special case of general relationship matrices, which can describe any quality of relation.

**Behavior**  An observable manner in which an individual interacts with his environment (e.g. movement, decisions, reactions to external stimuli, etc.).

**Bubble Diagram**  A transformation of a Cellular Automata into a graph, which contains a node for each function and an edge for each relationship between two functions.

**Cellular Automata (CA)**  A simulation in which an environment is modeled as set of cells, each of which is governed by transformation rules: For each instant $t_i$, the pattern of a cell and its surrounding neighbors leads to a new configuration at $t_{i+1}$. A classical example for this is Conway's “Game of Life”\(^9\), which has two states for a cell: “alive” and “dead”. There are four rules governing the transformation, (1) a live cell with fewer than two live neighbours dies, (2) a live cell with two or three

live neighbours stays alive, (3.) a live cell with more than three live neighbours dies and (4.) a dead cell with exactly three live neighbours becomes a live cell. There is a multitude of resulting spatio-temporal patterns that either “flow” through the cell-space, remain stable (“still”) or are annihilated after a few simulation cycles.

Cellular automata are typically used as discretized environment for Agent-Based Simulations. In contrast to Conway’s Game of Life, each cell may hold a variety of properties (e.g. walkable? <true/false>; elevation <number>; cell type <string>). Update rules are optional - in the simplest case, the cells stay static and are read by the agents passing by, e.g. for navigation. The normal case is that agents interact with cells by changing their value. This can also happen the other way around - cells can change properties of agents, which raises the question of “who” is the agent in that case. It gets even more interesting if cells transform themselves over time (think e.g. of grain growth, i.e. an update to a quantity held in a cell).

The distinction between “movable” agents and “inert” cells is, in my view, somewhat superficial: In my view, both are agents, since they can both execute rules. The distinction between cells and agents should rather be made regarding the ability to move.

Circulation Describes the way-system in a building, usually by means of line segments.

Complex Buildings Buildings which operate 24/7 and are planned around the prescribed routines of their building users, e.g. hospitals and airports.

Design Brief A document stating the building client’s requirements and current state of the building spot, which is to be addressed by planning (usually in the form of architectural competitions).

Design Method A description of the process of planning, sometimes prescriptive, sometimes deductive.

Early-Stage Planning All activities of preliminary occupation with a building project, originating both on the side of the building client and that of the planner (i.e. architect, urban planner, process planner and so forth).

Floor-plan A description of locations and sizes of building spaces in their definite form. By contrast, a schema is a preliminary floor-plan showing only relative sizes and locations of spaces, which have no definite form.

Flow Diagram A diagram which depicts the flow of persons and material between areas of a building.

Form Finding The process by which a schema is transformed
into a floor-plan, in a designerly (or otherwise creative) fashion.

**Function** The purpose of a space in a building.

**Functional Planning** A planning method which derives functions from the building user’s activities, sets their relationships and derives spatial zones.

**Functional Program** A (usually tabular) enumeration of functional units in a building. Each unit may further be attributed with area (square meters) and multiplicities (e.g. 2 operation rooms). Furthermore, functional units might be aggregated hierarchically, in order to address different semantic levels of a building (e.g. whole building, building components, functional areas, functional units).

**Graph** A mathematical construct consisting of vertices (also called nodes) and connecting edges.

**Hierarchy** A compositional property of a system: Every parts consists of sub-parts and relationships between them. This definition can be recursively applied for sub-parts, until the atomic level where parts are not further divided.

**Hospital Information System** An software system for managing and storing patient records in a hospital.

**Process** A sequence of activities which building users perform in a building.

**Spatial Proximity** A special kind of adjacency which denotes the intended location of functions (e.g. near, medium, far or n/a), see Adjacency Matrix for a generalized description.

**Process-Based Planning** A planning approach in which the building’s design is subordinate to prescriptive work routines of the building users.

**Preliminary Floorplan** see Schema.

**Relation** A quality between two functions, e.g. near, far, neutral or n/a.

**Relationship Matrix** A matrix describing relationships (or qualities) between functions, e.g. adjacency or cooperation within processes.

**Schema** An occupational medium (2D plan) in early-stage design which shows building zones with their relative sizes and locations and the circulation.

**Semi-Lattice** A hierarchy in which parts of a system can overlap (every part can have multiple parent parts). Also see trees

**Space Allocation Plan** A table of spaces and their respective areas in a building. Also see functional program.

**Space Program** see Space Allocation Plan
Threshold  An entry and/or exit of a zone.

Tree  A hierarchy in which parts of a system cannot overlap (every part can have one parent part). Also see trees.

Zone  A bounded space within a preliminary building concept. The zone can either describe an area (if the boundary is purely conceptual, e.g. as in a waiting area) or a room, building enclose and so on (in case the boundary is really meant to be one).

Zoned Bubble Diagrams  A type of bubble diagram in which functions are grouped (which is called zoning).

Zoning  A group of one or more functions into a zone.
Annotated list of figures and fonts used

In the light of the copyright problems cited in “Notes on style and structure of this Habilitation” (p. 11), I have chosen to redraw every graphic I would have otherwise cited, that is: I tried to draw entirely new graphics that have no resemblance to their original works. However, since the diagram styles given by these authors are essentially what is to be cited, I am certain that I have failed in at least two respects:

• The resemblance to original work comes from the usage of a visual style or metaphor intended by the original author(s). Even if I redraw all content depicted within a diagram, some parts will always look as if copied from the original work, since I am dealing with an arrangement of rectangles, circles and connecting lines.

• On the other hand, the reader has to be able to rely on my credibility. Do I really quote Neufert correctly when I produce a diagram in his sense, with no intended resemblance apart from the symbols employed? This problem weighs a lot more than the previous one, but I have a proposition to make: In the now following table, I have meticulously compiled a list of all graphics cited, from where they have been taken, and how I have redrawn them. You are welcome to check for yourself whether I correctly capture the style given by an author (if you are a researcher), and have recreated the figure to an extent that qualifies as being inspired by rather than having copied from the original work (if you are a lawyer).

<p>| Figures used in this cumulative thesis (excluding those in published papers) |
|---------------------------------|-------------------|------------------|-----------------|</p>
<table>
<thead>
<tr>
<th>Figure description</th>
<th>Location</th>
<th>recreated from</th>
<th>changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of schema</td>
<td>Figure 1 on page 7</td>
<td>Neufert 1992, Bauentwurfslehre, p. 256</td>
<td>functions removed in this bubble diagram for a school.</td>
</tr>
<tr>
<td>Differences schema-floor plan</td>
<td>Figure 2 on page 7</td>
<td>own work</td>
<td>-</td>
</tr>
<tr>
<td>Injecting agents</td>
<td>Figure 3 on page 8</td>
<td>own work</td>
<td>-</td>
</tr>
<tr>
<td>Research groups</td>
<td>Figure 4 on page 9</td>
<td>own work</td>
<td>Used by permission of Wikimedia Commons: World Map by Jose Carlos Garcia Lopez: CC by SA, added locations of important research centres</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Complex buildings</td>
<td>Figure 5 on page 15</td>
<td>own work</td>
<td>-</td>
</tr>
<tr>
<td>Early planning</td>
<td>Figure 6 on page 18</td>
<td>own work</td>
<td>-</td>
</tr>
<tr>
<td>Hospital planning</td>
<td>Figure 7 on page 19</td>
<td>Hardy and Lammers 1986, <em>Hospitals</em>, p. 16</td>
<td>simplified but left the overall process intact</td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site as active network</td>
<td>Figure 8 on page 21</td>
<td>White 2004, <em>Site Analysis</em>, p. 8</td>
<td>redrawn for Vienna</td>
</tr>
<tr>
<td>Location in Region</td>
<td>Figure 9 on page 22</td>
<td>White 2004, <em>Site Analysis</em>, p. 45</td>
<td>redrawn for Vienna Region, using map data from d-maps.com (free use)</td>
</tr>
<tr>
<td>Location in Neighborhood</td>
<td>Figure 10 on page 23</td>
<td>White 2004, <em>Site Analysis</em>, p. 46</td>
<td>redrawn for Schwedenplatz in Vienna, abstracted map data from OpenStreetMap (/non-accurate!)</td>
</tr>
<tr>
<td>Static Features</td>
<td>Figure 11 on page 23</td>
<td>White 2004, <em>Site Analysis</em>, pp. 69, 94</td>
<td>redrawn for Schwedenplatz in Vienna, photos shown are depicting view towards Stephansdom, along Ringkai and towards Prater (own work)</td>
</tr>
<tr>
<td>Views</td>
<td>Figure 12 on page 23</td>
<td>White 2004, <em>Site Analysis</em>, pp. 93-94</td>
<td>redrawn for Schwedenplatz in Vienna</td>
</tr>
<tr>
<td>Physical Forces</td>
<td>Figure 13 on page 24</td>
<td>White 2004, <em>Site Analysis</em>, pp. 64, 98-99</td>
<td>redrawn for Schwedenplatz in Vienna</td>
</tr>
<tr>
<td>Coupled System</td>
<td>Figure 14 on page 24</td>
<td>White 2004, <em>Site Analysis</em>, p. 60</td>
<td>redrawn for Schwedenplatz in Vienna</td>
</tr>
<tr>
<td>Distances and Travel</td>
<td>Figure 15 on page 25</td>
<td>White 2004, <em>Site Analysis</em>, p. 47</td>
<td>redrawn for Vienna Region, abstracted map data from OpenStreetMap (/non-accurate!)</td>
</tr>
<tr>
<td>Times</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Circulation</td>
<td>Figure 16 on page 25</td>
<td>White 2004, <em>Site Analysis</em>, pp. 57-58</td>
<td>redrawn for the inner city using abstracted map data from OpenStreetMap</td>
</tr>
<tr>
<td>Traffic Patterns</td>
<td>Figure 17 on page 26</td>
<td>White 2004, <em>Site Analysis</em>, p. 56</td>
<td>redrawn for Vienna</td>
</tr>
<tr>
<td>Zoning Regulations</td>
<td>Figure 18 on page 26</td>
<td>White 2004, <em>Site Analysis</em>, p. 49</td>
<td>redrawn using competition documents for “Praterstrasse 1”, <a href="http://www.architekturwettbewerb.at/data/media/med_binary/original/1209565796.pdf">www.architekturwettbewerb.at/data/media/med_binary/original/1209565796.pdf</a> [accessed 26-02-2013]</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------</td>
<td>------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Functional Derivation</td>
<td>Figure I on page 26</td>
<td>own work, with reference to White 1986, <em>Space Adjacency Analysis</em></td>
<td></td>
</tr>
<tr>
<td>Activity Specification (Schönfeld)</td>
<td>Table 2 on page 29</td>
<td>Schönfeld 1992, <em>Gebäudelehre</em> (2nd Ed.), pp. 157,17</td>
<td>simplified activity diagram for a single role instead of two, activities were garbled, departments also</td>
</tr>
<tr>
<td>Relationship Matrix (White)</td>
<td>Table 3 on page 31</td>
<td>White 1986, <em>Space Adjacency Analysis</em>, pp. 32-39</td>
<td>abstracted and simplified</td>
</tr>
<tr>
<td>Relationship Matrix (Schönfeld)</td>
<td>Table 3 on page 31</td>
<td>Schönfeld 1992, <em>Gebäudelehre</em> (2nd Ed.), p. 31</td>
<td>abstracted and simplified</td>
</tr>
<tr>
<td>Bubble Diagram (Neufert)</td>
<td>Table 4 on page 32</td>
<td>Neufert and Neufert 2000, <em>Architects Data</em> (3rd Ed.), p. 253</td>
<td>left concrete functions away but kept symbols; now shows only graphical depiction of flow rather than kitchen relationships</td>
</tr>
<tr>
<td>Bubble Diagram (White)</td>
<td>Table 4 on page 32</td>
<td>White 1986, <em>Space Adjacency Analysis</em>, p. 79</td>
<td>simplified and abstracted</td>
</tr>
<tr>
<td>Bubble Diagram (Schönfeld)</td>
<td>Table 4 on page 32</td>
<td>Schönfeld 1992, <em>Gebäudelehre</em> (2nd Ed.), p. 17</td>
<td>simplified and abstracted (originally this diagram is for a sports hall, removed that context)</td>
</tr>
</tbody>
</table>
functions removed in this bubble diagram for a school. Interestingly, this figure was later transformed into Neufert and Neufert 2000, Architecture’s Data (3rd Ed.), p. 314, which shows a non-nested bubble diagram.

- **Zoning (White)** Table 5 on page 34

  **Zoning (Schönfeld)** Table 5 on page 34

  Neufert 1992, Bauentwurfslehre, p. 256

  White 1986, Space Adjacency Analysis, pp. 103, 110

  Schönfeld 1992, Gebäudelehre (2nd Ed.), p. 31

  abstracted into a completely different diagram while keeping the same style.

  abstracted

  abstracted and garbled functions

- **Example Schema** Figure 20 on page 35

- **Nesting and Conceptual Containment** Figure 21 on page 35

- **Subordinate and Shared Zones** Figure 22 on page 35

- **Transitivity and Shared Function** Figure 23 on page 35

- **Schemata (Neufert)** Table 6 on page 37

  Neufert and Neufert 2000, Architects Data (3rd Ed.), pp. 302, 385, 570

  top to bottom:
  - schema with shared function has been simplified, functions garbled
  - bakery schema has been simplified and functions removed
  - “delivery” and “removal” diagram: all other functions have been garbled. entrance and exit locations and have been zones have been kept.

- **Schemata (White)** Table 6 on page 37

  White 1986, Space Adjacency Analysis, p. 122

  transformed into a simpler version

- **Schemata (Schönfeld)** Table 6 on page 37

  Schönfeld 1992, Gebäudelehre (2nd Ed.), p. 100

  abstracted and garbled functions
<table>
<thead>
<tr>
<th>Topic</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Schemata (Neufert)</td>
<td>Table 7 on page 38</td>
<td>Neufert and Neufert 2000, Architects Data (3rd Ed.), esp. p. 571</td>
</tr>
<tr>
<td></td>
<td></td>
<td>has been reduced to two levels, garbled functions but left “material” and “delivery”</td>
</tr>
<tr>
<td>3D Schemata (Schönfeld)</td>
<td>Table 7 on page 38</td>
<td>Schönfeld 1992, Gebäudelehre (2nd Ed.), p. 184</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reduced to two levels, simplified</td>
</tr>
<tr>
<td>Alexander’s Problem Decomposition</td>
<td>Figure 24 on page 40</td>
<td>own depiction inspired by Alexander 1964, Notes on the Synthesis of Form</td>
</tr>
<tr>
<td>Tree vs. Semilattice</td>
<td>Figure 25 on page 40</td>
<td>own depiction inspired by Alexander 1965, A City is not a Tree (Part I and II)</td>
</tr>
<tr>
<td>Activity using a Function</td>
<td>Figure 26 on page 41</td>
<td>own work</td>
</tr>
<tr>
<td>Non-atomicity</td>
<td>Figure 27 on page 42</td>
<td>own work</td>
</tr>
<tr>
<td>Capabilities vs. Functions</td>
<td>Figure 28 on page 42</td>
<td>own work</td>
</tr>
<tr>
<td>Ambiguities in relationships</td>
<td>Figure 29 on page 43</td>
<td>own work</td>
</tr>
<tr>
<td>Congruence</td>
<td>Figure 30 on page 44</td>
<td>own work</td>
</tr>
<tr>
<td>Consistency</td>
<td>Figure 31 on page 44</td>
<td>own work</td>
</tr>
<tr>
<td>Survey of design briefs</td>
<td>Figure 32 on page 46</td>
<td>own work</td>
</tr>
<tr>
<td>Big Bang Theory</td>
<td>Figure 33 on page 47</td>
<td>own work</td>
</tr>
<tr>
<td>ABS and emergence</td>
<td>Figure 34 on page 49</td>
<td>own work</td>
</tr>
<tr>
<td></td>
<td></td>
<td>from: Wurzer, Kowarik and Reschreiter (Eds.) Agent-Based Modeling and Simulation in Archaeology, 2015, Figure 3.9, p. 73, reprinted with kind permission from Springer Science and Business Media.</td>
</tr>
<tr>
<td>Environment</td>
<td>Figure 35 on page 50</td>
<td>own work</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Agent-Based Simulation Workshop at eCAADe 2012</td>
<td>Figure 36 on page 53</td>
<td>own photos</td>
</tr>
<tr>
<td>Screenshots of LK Wiener Neustadt Simulation</td>
<td>Figure 37 on page 65</td>
<td>own screenshots of the simulation, preliminary schema courtesy of LK Wiener Neustadt</td>
</tr>
<tr>
<td>Game Environment</td>
<td>Figure 38 on page 89</td>
<td>own work</td>
</tr>
<tr>
<td>Utilization Graph</td>
<td>Figure 39 on page 145</td>
<td>own work</td>
</tr>
<tr>
<td>Additional Graphics</td>
<td>Figures 40 on page 151, 41 on page 151, 42 on page 152</td>
<td>own work based on original design by Kamyar Tavoussi (wooden highriser), drawn by Sigrun Swoboda in AutoCAD and exported as image</td>
</tr>
<tr>
<td>ProceeDings</td>
<td>Figure 43 on page 160</td>
<td>own work</td>
</tr>
<tr>
<td>Agent-Based Design</td>
<td>Figure 44 on page 175</td>
<td>own work</td>
</tr>
<tr>
<td>Artificial Societies</td>
<td>Figure 45 on page 176</td>
<td>own work</td>
</tr>
<tr>
<td>Font name</td>
<td>Author</td>
<td>License</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>AnkeHand</td>
<td>Anke Arnold</td>
<td>Donation Ware</td>
</tr>
<tr>
<td>how cute</td>
<td>Kiana Janbakhsh</td>
<td>Public Domain</td>
</tr>
<tr>
<td>Permanent Marker</td>
<td>Font Diner, Inc</td>
<td>Apache License 2.0</td>
</tr>
<tr>
<td>Gel Pen</td>
<td>Shara Weber</td>
<td>Freeware</td>
</tr>
<tr>
<td>HelenasHand</td>
<td>Helena Jole</td>
<td>Freeware</td>
</tr>
<tr>
<td>Scrapbook</td>
<td>Two Peas</td>
<td>Freeware</td>
</tr>
</tbody>
</table>

**Fonts employed in this cumulative thesis** (excluding those in published papers)

**STABLE AIR**
- Figure 46 on page 188
- Own work, based on floor-plans of terminal C of Vienna International Airport, provided in the project

**STABLE HBO**
- Figure 47 on page 188
- Own work

**Integration of Clinics**
- Figure 48 on page 189
- Own work
- From the application for the MODYPLAN project

**MODYPLAN**
- Figure 49 on page 190
- Own work
- Screenshot

**Myself**
- Figure 50 on page 211
- Own photo

**Publication Graph**
- Figure 51 on page 212
- Own work
- Generated automatically from the Vienna UT publications database
Curriculum Vitae

Gabriel Wurzer, currently Researcher at Vienna UT

28.06.1980 Born in Vienna

1980-1983 Family moves to Munich

1983-1991 Family moves to Athens


1998-1999 Austrian Army

1999-2004 Study of Computer Science at Vienna UT. First publication leading to best presentation award at CESC

2002

16.11.2004 Graduation as Dipl.-Ing. (Master) with distinction (specialization on Computer Graphics, Thesis: “3D Regular Expressions” which I later published as a paper\textsuperscript{91})


• Multiple publications on simulation and visualization for hospital planning, egress and archaeology. Furthermore, publications in architectural tool support leading to Ivan Petrovic Award of the eCAADe in 2009.


10.05.2011 Graduation as Dr.techn. at Vienna UT


• Multiple co-advisorships (under supervision) for diplomas, mostly in simulation: Vassilis Kourkoutas; Matthias Auserer; Biljana Zaeva; Bernhard Platzer; Moritz Rosenberg; Benjamin Heinrich; Josef Öhreneder; Dimitris Kafetzopoulos; Patryk Wozniak; Benjamin Strassl. Furthermore technical advisorship during Ph.D. of Gerda Hartl (leading to IARIA SIMUL best paper award 2013 for our joint commuter mode-switching simulation).


\textsuperscript{92} own translation of German “Prozessvisualisierung in der Krankenhausplanung”
Publication Graph
Automatically retrieved from Vienna UT (based on the complete publication list at http://www.iemar.tuwien.ac.at/?page_id=434), see Figure 51:

Figure 51: Publication statistics for myself from 2002-2015. Automatically extracted from Vienna UT’s publication database.