Research focus

The purpose of this research is threefold. First, it proposes a practical framework for architectural and engineering design which, in collaboration with a human designer, can enable generation, evaluation, articulation, and effective communication of **Design Intelligence** (DI). Creating and applying such a framework on real-life design problems also affords an investigation on the nature of DI, its relationship to human and computational intelligence, and the role it can play in building a new design intuition that is grounded on design performance. Second, it attempts to bring powerful ideas and practices from the fields of Game AI and Procedural Content Generation and investigates the potential synergies and opportunities for knowledge and practice transfer between these domains and the domain of Architecture, Engineering and Construction (AEC). Third, it focuses on tethering the research and ideas researched onto practice by implementing a software version of the tool and applying it on different problems across the spectrum of activities in the AEC. These case studies will range both in terms of application, including urban design, building and layout design, and construction implementation (through the lens of offsite manufacturing) but also in terms of scale, from larger urban areas to single sites, whole floor plans to single dwellings, and from designing a single component to a kit of parts.

A Quick overview of the proposed generative design system

The problem of defining intelligence

The difficulty in even understanding how we could one day achieve general intelligence outside of humans is clearly evident in the difficulty to even define what intelligence is. In the context of AI research, Legg and Hutter [LH07] summarized no fewer than 70 definitions from the literature into a single statement: "Intelligence measures an agent's ability to achieve goals in a wide range of environments."

In general we find two different characterizations in definitions of intelligence: one that focuses on task-specific skills and one that is focused on adaptation across many different environments and skills. The first of these is nicely summarized by Minsky's famous 1968 definition of AI: "AI is the science of making machines capable of performing tasks that would require intelligence if done by humans" [Min69], which focuses almost entirely on skills acquired at narrow tasks normally handled by humans and has since resulted in ".. developing artificial systems that perform these tasks without featuring intelligence". [HernandezO17] The second, contrary to the task-focused approach, posits that intelligence instead "lies in the general ability to acquire new skills through learning; an ability that could be directed to a wide range of previously unknown problems". [Cho19] This describes the mind as being more flexible and with the capacity to learn new skills through experience, experimentation, learning and failure.

While both approaches have their own issues with the almost impossible task of defining and formalizing intelligence, it is quite easy to induce that the second one lends itself to the problem of design intelligence and the road towards a generative design system that is able to to adapt and perform under different design constraints, in entirely new design domains and sets of problems. Essentially, the goal of such a system would be to handle design problems and situations that it has not encountered before with relative capability and quality results. One more detail for the design of such a system comes from the two most important theories of human intelligence ([BJ05][McG05]) which organize human cognitive abilities in a hierarchal fashion, with the strata of general intelligence at the top, broad abilities in the middle, and specialized skills and tasks at the bottom (Figure 1).

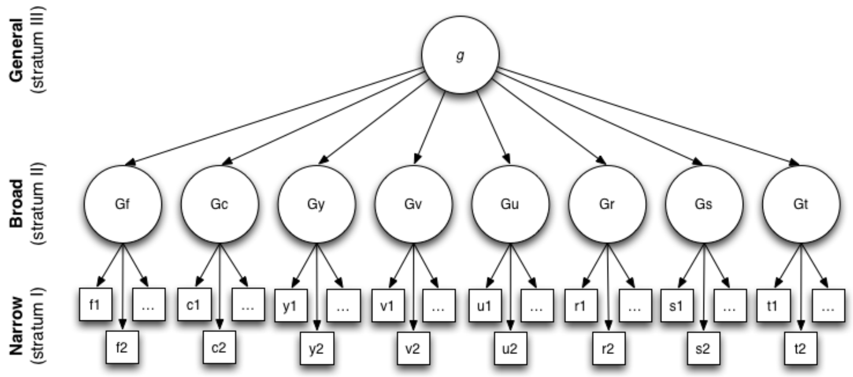
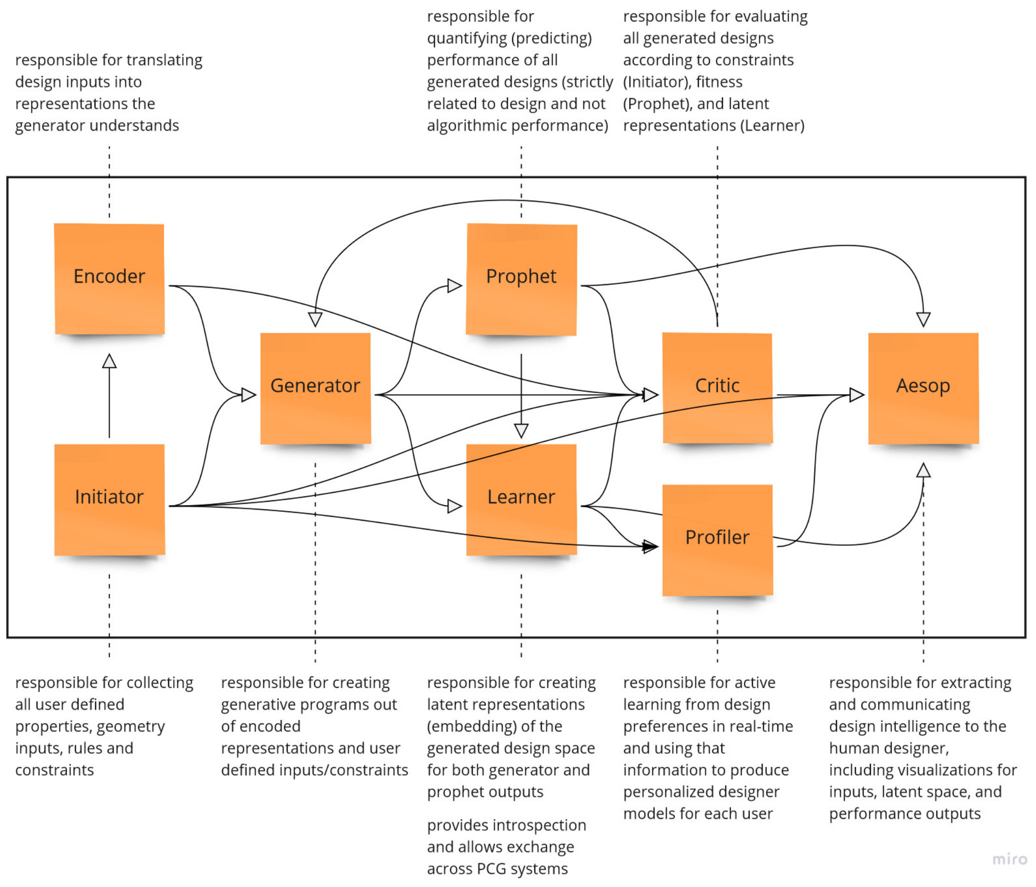


Fig. 1 CHC's three-stratum theory presented three levels of cognition: narrow abilities (stratum I), broad abilities (stratum II) and general abilities (stratum III).

A modular, hierarchical system

This assumption that information and intelligence follows a modular-hierarchical structure is a guiding principle for the design of the proposed system. The vision is that of a mixed-initiative interface that works along with human designer (stratum III) and on a range of different design domains and problems. Tackling these complex tasks can be made possible by a hierarchical and modular structure, where each module is a specialized and task-specific node (stratum I), and each iteration of the system is composed and (re)configured by different organizations of these modules (stratum II) in order to tackle a large diversity of complex design tasks. DI therefore is thought of as a meta-learning process taking place between the human designer and the tool as she interacts, explores, (re)combines, and (re)composes configurations of different modules specialized in specific design tasks, such as searching, learning, encoding, generating, manipulating, quantifying, and communicating designs and design performance. The mixed-initiative aspect of the system becomes crucial and acts as an evolutionary shortcut to Chollet's idea of a meta-learning algorithm which is responsible for the operation of such a system. Human intelligence, from the side of the human designer, makes the whole system work.

Another way of understanding these modules is to view them through the roles they each play in the functioning of the system and in the design process itself. This also provides a useful heuristic for naming these modules according to their function (Figure 2). The Initiator is responsible for capturing design inputs and constraints from the human designer in an easy and intuitive way and formatting them in a way that they can be used by the other modules of the system. The Encoder is responsible for encoding input parameters to design representations (genotypes) that the system can use to evolve, visualize and evaluate individual designs. The Generator is responsible for creating generative programs out of encoded representations and user defined constraints. These programs will then produce a collection of alternative designs to be evaluated and explored. The Prophet is responsible for quantifying design performance for each Generator output through the use of surrogate, pretrained deep learning (DL) models. The Learner is responsible for distilling the structures that are embedded in the collection of generated designs and does that by creating latent representations (embeddings) of all Generator and Prophet outputs. This allows the Learner to examine both input and performance latent spaces. The Critic is responsible for evaluating every design along with its quantified performance and selecting interesting or well-performing designs according to criteria set by the user. The Profiler is responsible for learning from the human designer's preferences, especially focusing on the relationship between inputs/constraints, design space, and performance. Finally, Aesop is responsible for extracting and communicating design intelligence to the human designer about the current design task, extracted from the data produced by the system.



responsible for translating design inputs into representations the generator understands
responsible for quantifying (predicting) performance of all generated designs (strictly related to design and not algorithmic performance)
responsible for evaluating all generated designs according to constraints (Initiator), fitness (Prophet), and latent representations (Learner)
Encoder
Prophet
Generator
Critic
Aesop
Initiator
Learner
Profiler
responsible for collecting all user defined properties, geometry inputs, rules and constraints
responsible for creating generative programs out of encoded representations and user defined inputs/constraints
responsible for creating latent representations (embedding) of the generated design space for both generator and prophet outputs
provides introspection and allows exchange across PCG systems
responsible for active learning from design preferences in real-time and using that information to produce personalized designer models for each user
responsible for extracting and communicating design intelligence to the human designer, including visualizations for inputs, latent space, and performance outputs
miro

Fig. 2 Roles and modules of the proposed design generation system and the relationships between them, arrows denote data transfer.

The modular nature of the system can manifest in a hierarchical manner, at three different conceptual levels, with potential practical implications:

- **At the level of the module**. The modules themselves can include multiple algorithms, techniques, models, methods, and code, depending on the task. For example, the Generator module can use different generative techniques, the Critic module can use different evaluation methods, the Aesop module can use multiple visualization strategies for the user to select. Thus, each module is in itself modular in a way, a very important property for addressing a diversity of complex tasks. This can potentially allow he designer to create recipes, specific combinations of modules, methods, and representations, for specific tasks.
- **At the level of the system**. The modules can be combined in different ways to create different versions of the system that can deal more efficiently with different tasks and goals which allows for hierarchical adaptation, at the system and module level, to different problem requirements. From this perspective, the system also implies an open-endedness in its structure with the potential of new modules, specialised in additional, new tasks, added by human designers in the future.
- **At the level of design**. It is also possible to use the system at different levels of the hierarchy of design. In fact, the case studies that the research will involve are designed to reflect exactly this plasticity of application. The system will be applied to urban design, massing design, floor plan design, layout design, and finally component design. At each different level of this hierarchy, from component to layout, from floor plan to building massing and urban design, the system will potentially explore the use of different representations, generative methods, evaluation processes, and fitness functions. Finally, since performance at these different levels is highly coupled, information exchange between them with regards to performance will be explored and the impact on solutions at individual levels evaluated.

Apart from being the central components of the proposed system, these modules and their function also guide the required literature analysis for the proposed research. This will need to cover issues and ideas concerning design representation and encoding, generative models and techniques, representation learning, evolutionary computation, search and optimization, latent space exploration and model interpretability, preference learning, gaming AI, mixed-initiative design, UI/UX design and visualization. The research questions along with a brief review of the state-of-the-art in these fields, highlighting the important ideas and research that will be useful for the development of the system, will be detailed in the next sections.

References:

[BJ05]

Thomas Bouchard Jr. The structure of human intelligence: it is verbal, perceptual, and image rotation (vpr), not fluid and crystallized. *Intelligence*, 33:393–416, 07 2005. doi:10.1016/j.intell.2004.12.002.

[Cho19]
François Chollet. On the measure of intelligence. 2019. arXiv:1911.01547.

[HernandezO17]
José Hernández-Orallo. Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement. *Artif. Intell. Rev.*, 48(3):397–447, October 2017. URL: https://doi.org/10.1007/s10462-016-9505-7, doi:10.1007/s10462-016-9505-7.

[LH07]
Shane Legg and Marcus Hutter. A collection of definitions of intelligence. 2007. arXiv:0706.3639.

[McG05]
Kevin McGrew. The cattell-horn-carroll theory of cognitive abilities: past, present, and future. *Contemporary Intellectual Assessment: Theories, Tests, and Issues*, pages, 01 2005.

[Min69]
Marvin L. Minsky. *Semantic Information Processing*. The MIT Press, 1969. ISBN 0262130440.

Transdisciplinary approach

The proposed research relies on finding fertile ground on the intersection of several different domains of knowledge and practice which are adjacent, in some ways, both due to the problems they try to solve but also the ideas they use for solving them. A brief description of these domains follows below.

The first domain is **Computational Architecture**, a subdomain of Architectural Design that focuses, among other things, on developing computational tools, workflows, and algorithms, that can generate and evaluate design artifacts in an efficient and diverse manner. The field spans most areas of design, ranging from the urban level all the way down to the construction component level and relates to practical areas of design and construction such us manufacturing, robotic fabrication, and modular design.

The second domain is **Machine Learning** (ML) and its subdomain **Deep Learning** (DL). DL has had incredible growth, research activity, and practical successes in the last decade. DL models play an important role for this research, in different ways:

- allowing real-time evaluation of performance through surrogate modelling,
- enabling an encoding of the design space at different moments of the design process,
- and assisting with design exploration, visualization and finally extracting Design Intelligence (DL).

The DL models that will be developed and used in this research come from the areas of **Computer Vision** (CV), specifically from the research domain of Domain Adaptation, of **Graph Theory** and **Graph Neural Networks** (GNNs), and the more diverse domain of **Self-Supervised Learning** (SSL) which includes new ideas and techniques for learning through representations without labels, an important attribute for a generative design system.

The third domain is that of **Evolutionary Computation** (EC) with a focus on the newly established field of **Quality Diversity** (QD). EC began in the late 1950s and 1960s along with the larger availability of computer programs that allowed the testing of models of evolutionary processes and includes important EC paradigms such as evolutionary programming (EP), evolution strategies (ES), and genetic algorithms (GAs). The main focus is on simulating various aspects of evolution and while the techniques developed have differences in their implementations and insights, they all have one fundamental commonality: “they each involve the reproduction, random variation, competition, and selection of contending individuals in a population” (ref: Handbook of EC). QD, a much younger sub-domain of EC which started with the seminal work on Novelty Search by Kenneth Stanley (ref: novelty search), focuses on the ability of natural evolution to produce a diverse collection of organisms, in our case designs, that are all well-performing in their niche. QD therefore “flips the script”, changing the focus from traditional evolutionary approaches that look to find a single high-performing solution towards design space exploration, identifying viable and well-performing solutions at every niche of the design space.

The fourth domain is **Gaming AI**, which includes a variety of sub-domains of interest: **procedural content generation** (PCG), **mixed-initiative design** (MID), **player experience** (PEX) and user **preference learning** (PL). The work done in Gaming AI is both highly interesting and applicable for the AEC, something which is becoming evident by the growing interest and number of applications in the field which take advantage of gaming technologies and tools (e.g. VR/AR, generative design, visualization, and simulation). An advantage of intersecting with the domain of Gaming AI is that a lot of the research being done not only has practical implications for design but also typically includes a practical implementation, case studies showcasing these new ideas in practical applications (in new games or gaming environments). Additionally, a lot of insights can be gained in areas important for design, such as generative design and its role in creating content of all kinds, evaluating expressivity and diversity of the design artifacts generated, understanding the behavioral space of designs and their relationship with designers and users, and developing interfaces that enable an intuitive MID process with a human designer.

In a true transdisciplinary approach, the above domains cannot be thought of in isolation from each other nor can they simply be connected at obvious points of intersection, where specific design challenges for the AEC are found. Instead, they need to be embedded and intertwined within the very structure and dynamics of the proposed generative design system and together create a tool that offers a seamless design workflow and user experience. For that reason, the next few sections will review relevant works and ideas from these domains, from the point of view of the system and its modules, and will detail both how these relate to each module and how they allow for relationships and information transfer to form between modules.

Initiator

The Initiator module is the user interface (UI) that enables most of the mixed-initiative aspects of the system. To do that, the module needs to have access to the various geometrical components used for generating design outputs and to the content, and its properties, produced by the Encoder, Generator, Critic, and Aesop modules. The role of the initiator is crrucial as it is the way through which the computational design process is conveyed to the human designer. A poorly implemented UI can obfuscate even the most elaborate AI running in the background and confuse the designer in terms of what the system is doing behind the scenes. The Initiator must be able to perform mundane but time consuming tasks with ease, such as allowing users to import and load past work in different file formats as well as save their work for another design session with this tool or in the next tool in the architectural design pipeline. Similarly, the Initiator should allow the human designer to manually draw or define elements of the design language (chunks, tiles, shapes, components), allow for simple geometry manipulation (rotate, scale, translate), geometrical operations (intersect, difference, union) and editing (assigning colors, textures, materials). In practice, such a tool would need to include a predefined database of these elements with available shapes and components that can be used off the shelf.

There is a long history of applied research on the role of the interface in the communication of information during the design process. Some of the earliest works were by Christopher Alexander with HIDESC3 [Ale63] in the 1960s and Nicholas Negroponte and Yona Frieman with Urban5 [NG67] and YONA [WN76] in the 1970s. Negroponte's "Architecture Machine" [Neg75b] explores the issue of computer aided-architectural design, with emphasis on man-machine interaction, a model where the machine and its user develop a collaborative relationship and establish a creative and educational dialogue. One interesting idea in this line of work was the goal of enabling the machine to be aware of the user's preferences, creating the ground for what they called mutual and successful "interruptability". Another important aspect of the role of interfaces is the vision by Yona Friedman [BLA76] of "democratizing" design, to free the user from the "patronage" of the architect, to enable "non experts" to make their own designs, as they are the ones who better know their needs and desires and, most importantly, bear the risk of failure. (ref: architecture by yourself paper). User participation becomes a major focus on research after that, including the idea of design amplifiers [Neg75a] that constitutes the interface between the infrastructure and the user's ever changing needs and the ideas by Guy Weinzapfel and Negroponte in which explores an extreme case of an "architecture machine" that handles a unique, ever changing and completely personalized problem of design [WN76]. In all of this work, the machine is conceptualized as a compact entity, whose ways of operation remain obscure to the user. where a communication happens in a higher level language, through a well designed interface. They also point towards the implications of creating a user programmable architecture machine, for which the modular-hierarchical system described earlier is a good archetype.

Despite this work being done mostly between 1960 and 1980, it's difficult to say that any substantial steps forward have been done in the last 30 years or so. Design interfaces in practice have seen little evolution and almost completely lack MID elements. Most interfaces are tied into a design software, used by practitioners, rather than influenced by new theories and ideas about generative design, human-computer interaction (HCI), and mixed-initiative design (MID). Instead we turn to the Gaming AI field where substantial research and innovative applications have been developed the last decade. Sentient Sketchbook [LYT13] is a game level authoring that allows the design of map sketches via an intuitive interface, allowing for real-time feedback of evaluated metrics for each map (playability, balance) and provides suggestions of alternative map designs to its users. The tool automatically evaluates for different metrics and constraints and provides live feedback to the designer through its visual interface. Finally, using a variant of Novelty Search [LS08] the tool provides unseen, alternative maps, that achieve the designer's goals. The performance difference between the current and alternative designs is clearly displayed in the interface to reduce designer fatigue and allow for quick iteration. Tanagra [SWM10] is a mixed-initiative tool that allows designers to make and play platform levels that are playable, engaging and meet their expectations. The focus of the tool is on the fact that it is important for designers to be able to test their own designs. The design generation method is quite innovative, using a top-down approach that represents a level as a set of 'beats', smaller segments that subdivide the level by both space and time (required to traverse them). The tool also records designer preferences, based on the changes and inputs the designer did on the previously generated design, an important way of communicating information during the design process. Another interesting MID method is the occupancy-regulated extension algorithm [MawhorterMateas10], a general method that supports authoring from the designer without requiring her to know the rules of the game. The methodology, much like Wave Function Collapse, works on a pre-defined collection of chunks, geometric components that can be used to develop designs, and even proposes a methodology for design language extraction from real game levels by dissecting them into smaller images. Ropossum [SST13] is a tool for automatic design of complete and solvable content. It incorporates designer input through the creation of complete or partial designs, automatically checks for playability and optimizes a given design according to playability score. Ropossum includes a physics engine and playability constrains and like the previous tools allows for partial design generation with designer input, enabling a MID approach.

From the examples above we can clearly see that the focus of most modern and classical tools allowing the designer to influence the process of generation in real-time, which is the core of a MID approach. All tools focus on creating valid designs, something extremely important for architecture and engineering. Finally, they all work to develop practical tools that work in real life, a crucial aspect of bringing modern, mixed-initiative, generative design forward. There are numerous challenges that arise in the scope of the Initiator. One obvious challenge is how to find an intuitive way for designers to communicate their constraints across different design spaces and tasks. The second challenge is to how incorporate needed features in a consistent UI, even across different configurations of the system modules. One the other hand, as the gateway of information to the system, the Initiator offers exciting data-collection opportunities. A database of constraint configurations for different design problems can easily be collected through the use of the system. The system can also potentially learn each user's typical constraints, and value ranges, for different tasks and quickly recommend set ups for future studies. Finally, the data collected at this stage is valuable information that should be communicated to the Critic and Aesop modules, provide a view into the relationship between constraints and performance, a crucial step towards extracting design intelligence.

References:

[**Ale63**]
C. Alexander. *HIDECS 3: Four Computer Programs for the Hierarchical Decomposition of Systems which Have an Associated Linear Graph*. Massachusetts Institute of Technology. Civil Engineering Systems Laboratory. Research report. MIT, 1963. URL: https://books.google.com.my/books?id=u7Z2NwAACAAJ.

[**BLA76**]
JUDITH R. BLAU. Toward a scientific architecture by yona friedman. the mit press, cambridge, massachusetts, 1975. 169 pp. (translated by cynthia lang.). *International Journal of General Systems*, 3(1):68–70, 1976. doi:10.1080/03081077608934740.

[**LS08**]
Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI*. MIT Press, 2008.

[**LYT13**]
Antonios Liapis, Georgios N. Yannakakis, and J. Togelius. Sentient sketchbook: computer-aided game level authoring. In *FDG*. 2013.

[**Neg75a**]
N. Negroponte. *Soft Architecture Machines*. MIT Press, 1975. ISBN 9780262140188. URL: https://books.google.com.my/books?id=sRVQAAAAMAAJ.

[**NG67**]
N. Negroponte and L.B. Groissier. *URBAN 5: An OnLine Urban Design Partner*. Cambridge, IBM Report. IBM, 1967.

[**Neg75b**]
Nicholas Negroponte. The architecture machine. *Computer-Aided Design*, 7(3):190 – 195, 1975. URL: http://www.sciencedirect.com/science/article/pii/0010448575900093, doi:https://doi.org/10.1016/0010-4485(75)90009-3.

[**SST13**]
Mohammad Shaker, Noor Shaker, and Julian Togelius. Ropossum: an authoring tool for designing, optimizing and solving cut the rope levels. In *9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 215–216. 2013.

[**SWM10**]
Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: an intelligent level design assistant for 2d platformers. In 01 2010.

[WN76](1,2)
Guy Weinzapfel and Nicholas Negroponte. Architecture-by-yourself: an experiment with computer graphics for house design. *Association for Computing Machinery*, 10(2):74–78, 1976. URL: https://doi.org/10.1145/965143.563290, doi:10.1145/965143.563290.

[MawhorterMateas10]
P. Mawhorter and M. Mateas. Procedural level generation using occupancy-regulated extension. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, volume, 351–358. 2010.

Encoder

The Encoder is the interpreter of the system, responsible of translating design inputs into representations that the Generator will understand. The Encoder's role is crucial since representation is perhaps the most important aspect of every generative system, as it defines what can be generated and determines whether effective search is possible [ST16]. The Encoder module is connected to the Initiator, Generator and Critic modules but has the strongest effect on the Generator and the generation process. Different types of representations lead to sometimes completely different generative schemes and output designs, so selecting appropriate representations is crucial.

When talking about representations, it will be helpful to define a taxonomy of content that the system can work with and the Encoder is able to represent. This taxonomy is specific to the AEC and closely relates to the hierarchical structure of the system we described earlier:

- bits: elementary units of designs, for example in the case of construction components these can be windows, walls, partitions, doors, furniture, etc.
- spaces: minimal structures that define areas that share similar properties, for example in the case residential typologies these can be the living room, dining room, bedroom, corridor, kitchen, bathroom, etc.
- context: anything relevant to the generation of designs that is outside of the design itself but impacts design performance and requirements, this can include location, orientation, climate data, building floor plan, building massing, plot boundaries, urban context, depending on the design problem.
- adjacency: graph data that captures relationships, connections, navigation and other aspects of space structure
- metrics: either direct (e.g. area, volume, cost, counts ,etc.) or indirect, simulated metrics such as daylight performance, natural ventilation, energy consumption, occupant comfort and more.

It is important here to remember that due to the hierarchical structure of the system itself every layer of the representation taxonomy is embedded into the layer after it. For example, spaces are made of bits, while a floor plan is made out of spaces and adjacencies. An adjacency is possible between bits, spaces and contexts while metrics can be quantified for a bit (e.g. number of instances for a specific component), for a space (e.g. daylight performance of a bedroom), or for a whole floor and context (e.g. wind and thermal comfort). The Encoder module is responsible for keeping track of all this information and communicating it to the other system modules that require it for their function.

There is a considerable history of research on representation, both in relation to architectural and engineering design and more more generally in the context of evolutionary computation (EC). One of the earliest works in the former domain is by Charles Eastman [Eas70], investigating the issue of representation in the design world. Eastman draws from insights collected by human designers as to what is important in a drawing. The survey highlighted the importance of clear demarcation of spatial domains, homogeneity in the way designs are represented and availability of all spatial information to the designer. Eastman goes on to give an early proposal of different ways of computational representations of space, focusing on the use of arrays and strings as data structures and on shapes, dimensions and adjacencies as elements. Perhaps the most thorough investigation of representation in design is by Woodbury [Woo91]. The paper takes an equally transdisciplinary approach in an effort to understand the concepts of representation, design space, and search, using a nexus of intellectual sources including linguistics, set and graph theory, cognitive psychology and AI. The work positions representation within a formalized system of design and search which includes the aspects of modeling, representations, expressivity, range, and operators. It also highlights the important relationship between representation and search, with properties of the former influencing properties of the latter. It goes on to detail two different approaches to design representation, an expert system approach that relies on prior knowledge [FUC+88] and shape grammars [SG71], with the latter having great potential for both representation and generative capacity. Woodbury's work provides the basis for a formal framework of generative design and has potential synergies with more modern developments in AI and specifically in Quality Diversity and the understanding of search and design space there.

While there is ample research in the AEC domain concerning these concepts we turn once again to Gaming AI and specifically procedural content generation (PCG) for practical examples of tools that were developed with the explicit purpose of generating alternative designs. Representation, as we expect, plays a major role in this work. Togelius et al. [TYSB10] provide a good introduction to how representation can be formalized, introducing the main formal distinction for representation, that of direct and indirect encodings. Yannakakis et al. [TogeliusPreussBeume+10] present an Evolutionary

Multiobjective Optimization Algorithm (EMOA) and propose a mixed encoding scheme which combines a direct representation, a 64x64 heightmap, only used during fitness and visualization and an indirect representation, a fixed-length array of real values between 0 and 1, which is used to efficiently search for new designs. The paper highlights an important area of investigation, the trade-offs between direct and indirect representations in different stages of the generative design process. Shaker et al. [ShakerNicolauYannakakis+12] propose a different approach to representation, utilizing Grammatical Evolution [Brown97] and shape grammars to generate Super Mario Bros. game levels. The levels are represented as set of chunks that include geometry and properties. GE allows for the incorporation of domain knowledge through the ability of human designer to design her own representations (chunks). The work highlights the potential of GE representations due to their compact descriptions and efficiency. Sorenson et al. (ref here) present a functional approach to representation where the whole level is an individual in the evolutionary process. They represent the level using a dirac function, with a 0 value everywhere (initial state) apart from where components are introduced where value is 1. Evaluation focuses on high-level semantic attributes, in this case anxiety and fun, which raises the important question of which similarly important, semantically meaningful parameters can be used in the context of architectural design, and how can they be represented as a generative process. Returning closer to the AEC domain, Hornby and Pollack [HornbyPollack01] introduce generative encodings for generating physical design, and instead of directly representing geometries with components and parameters they use rules to construct a phenotype. A special case of L-Systems [Jac94] is used for defining the parametric, context-free rules. This process produces very robust representations which allow for mutation and recombination to happen at multiple levels and in different ways. The way to allow for robust and expressive ways of generating alternatives in the highly constrained design tasks that are common in the AEC is an important area for further research.

As is evident from the diversity of approaches to representation, finding efficient ways to represent designs, for different tasks, is the biggest challenge of the Encoder module. Despite the research conducted in other domains, this has not been researched in AEC practice thoroughly. A further, domain-specific challenge is that architectural and engineering design typically involve different modalities of input data, such as text, images, textures, meshes, time series, vectors, and more. Any generative system that aims to develop design solutions that can be helpful in the real world needs to accommodate for these modalities and take advantage when possible of the information they offer during generation. Efficient preprocessing and encoding procedures for each of these modalities will be an important aspect of the Encoder module. Similarly to the Initiator, the Encoder module provides the opportunity to learn relationships between different types of inputs the user inputs and different design tasks the use tries to solve. This is because efficient representations that are fit for purpose will perform well and can be retained and reused. Automating this process, or at least abstracting it enough to make it intuitive and easy-to-use, is crucial to lower the barriers of entry for designers to use the system.

References:

[**Eas70**]
Charles M. Eastman. Representations for space planning. *Commun. ACM*, 13(4):242–250, April 1970. URL: https://doi.org/10.1145/362258.362281, doi:10.1145/362258.362281.

[**FUC+88**]
Ulrich Flemming, Ulrich, Coyne, Robert Glavin, and Timothy al. A generative expert system for the design of building layouts – version 2. *Artificial Intelligence in Engineering: Design. editor. John J. Gero. Elsevier (Computational Mechanics Publications), 1988. PP. 445-464 : ill. includes bibliography*, pages, 01 1988.

[**Jac94**]
Christian Jacob. Genetic l-system programming. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, 333–343. Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[**ST16**]
Noor Shaker and Julian Togelius. The search-based approach. In Noor Shaker, Julian Togelius, and Mark J. Nelson, editors, *Procedural Content Generation in Games*, pages 17–29. Springer, 2016.

[**SG71**]
G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In *IFIP Congress*. 1971.

[**TYSB10**]
Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation. In Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna I. Esparcia-Alcazar, Chi-Keong Goh, Juan J. Merelo, Ferrante Neri, Mike Preuß, Julian Togelius, and Georgios N. Yannakakis, editors, *Applications of Evolutionary Computation*, 141–150. Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[**Woo91**]

Robert F. Woodbury. Searching for designs: paradigm and practice. *Building and Environment*, 26(1):61 – 73, 1991. Special Issue Developments in Computer-Aided Design. URL: http://www.sciencedirect.com/science/article/pii/036013239190040I, doi:https://doi.org/10.1016/0360-1323(91)90040-I.

[**Brown97**]
K. Brown. Grammatical design. *IEEE Expert*, 12(2):27–33, 1997.

[**HornbyPollack01**]
G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, 600–607 vol. 1. 2001.

[**ShakerNicolauYannakakis+12**]
N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, and M. O'Neill. Evolving levels for super mario bros using grammatical evolution. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, volume, 304–311. 2012.

[**TogeliusPreussBeume+10**]
J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis. Multiobjective exploration of the starcraft map space. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, volume, 265–272. 2010.

Generator

The role of the Generator module is to automatically produce designs out of the encoded representations, user inputs and constraints. The ways to produce designs, uses, algorithms, and levels of control are numerous. In order to identify the possibilities of the Generator module it is helpful to consider properties of a taxonomy of generators and try to identify which of these properties are important, or possible, for the Generator module. We refer to the taxonomy of generators from the framework of Procedurally Generated Content (PCG) defined in [ST16] and try to adapt it to the AEC domain. The main practical choices detailed in the taxonomy are:

- **Online vs. offline**: a generator can produce content while the designer is interacting with the tool (online) or in batches during downtime (offline). Online generation "places two or three main requirements on the algorithm: that it is very fast, that it has a predictable runtime and (depending on the context) that its results are of a predictable quality" [ST16]. In principle, generation for a design tool should be online as the designer should immediately receive generated content as a response to their current action. In theory, however, an offline generator could run a priori for many hours, producing a large database of diverse possible designs and then, during the design process, the most appropriate design could be chosen from the database and shown to the user.
- **Necessary vs. optional content**: a generator can produce elements that are central to the design's function (necessary) or elements which are mostly decorative and could be either omitted or replaced by a different element without affecting the design's function (optional). In terms of design, an example of collection of necessary content is a building's layout where all rooms must be connected through doorways, the doors should be high enough for residents to pass through, and so on, while the doors' wood color or a room's wallpaper motif is optional content.
- **Random seeds versus parameter vectors**: the level of control that the generator has over the content affects both the type of designs that can be produced and the balance between computational and human initiative. Parametric architecture has already shown how a parameter vector can affect a design; however, the level of control of the generator remains pertinent. A well-defined, narrow set of design parameters can allow the generator to search for optimal designs without making large changes to a human-authored design, but this approach sacrifices diversity. On the other hand, allowing for more freedom, either through larger parameter spaces or from more flexible representations, can lead to novel and creative solutions that local search would not be able to discover.
- **Stochastic vs. deterministic generation**: the degree of randomness (stochasticity) affects the potential of the generator to discover new and creative solutions, but also contributes to the risk of dramatic changes that can move the solution far from the designer's intentions. Deterministic processes allow the designer to anticipate what the generator will do, and are thus easier to explain. They also might be more appropriate for transformations of one type of content into another (e.g. a hand-drawn sketch to a floor layout). On the other hand, stochastic processes allow the infusion of contingency and surprise in the generation process, a crucial element of a truly expressive generator.
- **Constructive vs. generate-and-test**: the types of algorithms in PCG are often distinguished between those that perform "operations, or sequences of operations, that are guaranteed to never produce broken content'" [ST16] (constructive) versus those that evaluate the quality of the produced content and re-generate a new solution if it is of insufficient quality (generate-and-test).

It is interesting to think of generative design in the context of the AEC through the above properties of the taxonomy proposed. Concerning the way designs are generated, almost by default, the offline method is utilized in generative design settings due to the way the popular design software (Rhino/GH, Revit/Dynamo, Forge) work. Given the expensive evaluation costs for most interesting metrics we want to quantify in the AEC, the offline setting does seem to be currently the best candidate for large scale design exploration, which is what the current proposed systems endeavors to develop, although online generation as a second step would be an interesting addition to the system allowing for mixed-initiative design elements, where the human designer investigates and alters the offline generated collection. When it comes to content, things are much clearer. For architectural and engineering design most, if not all, content produced during early concept phases is necessary. There are very few aesthetic elements that we would care to incorporate in conceptual design at that stage. In lieu of content generation, it's also hard to imagine many cases where random parameter vectors of any kind could produce meaningful and valid designs. Instead, we expect that at least several geometric primitives (points, lines, polygons, and so on) will have to be present in order to be able to generate and evolve complex designs. Things get more interesting when we consider the horizon of the generative and search processes. At this point in time, most design generation happens strictly in the deterministic domain, again mostly due to the restrictions imposed by the design software and partly by the paradigm of parametric design that is currently popular in practice. Some more stochastic approaches do exist, but those are scarce and typically found in the academic domain, with practical applications preferring instead ease of use and transparency of generative methods. Finally, for very similar reasons, the constructive approach is predominant in the AEC. In fact, determinism and constructivism seem to be the two philosophical pillars of the current state-of-the-art, practical generative design, something that severely limits the potential expressivity of generators and their ability to find truly new, interesting, and well-performing designs.

The most influential choice for the Generator of course is the one of the generative algorithms used to generate designs. There are multitude of generative algorithms that have been used for content generation, across many domains, with varying implementation details and implications for search. Jie Lim et al. [SJLW20] use Cellular Automata (CA) for the generation of mass-customized building blocks in Jakarta's kampungs (informal urban settlements). A combination of bottom-up rule sets for the CA and top-down strategies of development is used in a process that aims to improve the environmental performance of generated morphologies. Christiane and Thomas [HK05] use CA to generate high-density building form. In contrast to typical CA approaches, their research focuses on the re-modelling of an existing project, showing the value of CA on a more practical real-life design task. Koenig and Bauriedel [KB09] present a CA method to steer precisely the generation of settlement structures with regard to their global and local density as well as the size and number of forming clusters, allowing for precise control over the developing structure, an important functionality when using stochastic generation processes like CA.

Much more research has been done on evolutionary processes for design generation, and especially Genetic Algorithms (GA) in the AEC. Schneider et al. [SFKonig11] present a creative evolutionary methodology for layout problems that explores the design space across the dimensions of intuition and rationality in a variety of ways. The paper highlights the importance of identifying important behavioral descriptors of the design space that relate to the human designer. Byrne et al. [BFH+11] evolve designs by their ability to handle physical, structural, constraints in a process where the physical constraints interact with aesthetic considerations of the human designer. They combine the evolutionary approach with a design grammar that is capable of evolving aesthetically pleasing designs. In another Shape Grammar (SG) approach, Lipp et al. [LWW08] introduce a real-time interactive editing paradigm that allows the human designer to create rules in an intuitive manner. The work explores the important MID implications of real-time generation of such systems. Bielik et al. [BSK12] use DecodingSpaces, a graph-based analysis toolset, to generate a variety of different urban designs according to detailed design parameters and properties of the urban area. For the AEC domain, tools that enable the generation of an initial seed population of designs might prove to be critical for large-scale generative design. Recently, Nauata et al. [NCC+20] developed House-GAN, a relational Generative Adversarial Network model for graph-constrained layout generation which improved the state-0f-the-art in layout generation. They also provide an intuitive interface through which the designer can, in real-time, interactively draw adjacency graphs and see generated layouts. DL models will potentially be crucial for the Generator going forward, especially in design domains where evolution of a design is complex and cumbersome, and can potentially provide a streamlined way to generate design alternatives.

A lot of parallel research in all of these domains has also been conducted in the PCG and Gaming AI field (see for example [KaravolosLiapisYannakakis18], [SSN+], [LYT13], [SYT10] among many others) with a lot of interesting results and generative design approaches. But the most important, newly emerging, domain of research, focusing on a new approach of expressive generation of design alternatives with a focus on design exploration, is that of Quality Diversity (QD). The QD field was initiated by the seminal work of Stanley and Lehman [LehmanStanley11] on the idea of Novelty Search (NS), searching through a design space without objective. This has lead to an explosion of research around NS, which coupled to the idea of quality (in place of objective) creates the notion of Quality-Diversity. The goal of QD is to create a process that produces a diversity of organisms (or designs) that are all high performing in their own niche. The most successful algorithm that came out of QD is MAP-Elites by Clune and Mouret [MC15] which develops a collection of well-performing designs (elites) across a discretized behavioral space. QD and MAP-Elites revolutionize the way we approach design exploration and optimization in the AEC, by focusing on exploration while still finding both interesting and well-performing designs. Instead of one, or a front of, good solution(s) as with typical evolutionary approaches, QD finds a whole collection of designs which can have different characteristics and in fact could be useful in different design tasks and constraints. QD will be the focus of the generation-evaluation-selection process in the proposed system and will heavily influence the inner workings of the Generator and Critic modules.

The Generator, like all other modules of the system, comes with its own set of challenges and opportunities. A core challenge is whether it can consistently create content that is coherent, valid, original, creative, and performative. Architectural design has always been considered too difficult to do well in an automated way. This is where the QD approach can prove significant in allowing for proper exploration of diverse design spaces. Additionally, the need for consistency and computational speed is paramount. Furthermore, the level of control of a generator is also important, especially if a proper MID interface is to be developed. However, having more parameters that can affect the generator may also create broken content which increases the computational effort needed from the generator and mental effort by the human designer. Relatedly, efficient data pipelines and data-efficient representations need to be developed to allow quick transfer of data between the generator and other system modules. While individual designs are typically quite light-weight, having design spaces that range in the 100s of thousands of alternatives can become a bottleneck and has to be dealt with during system design.

Just like before, one of the biggest opportunities in the Generator module seems to be related with learning, in this case the being able to learn how to express specific architectural styles and typologies, and even specific styles of the human designer through the evaluation of his decisions. How to transfer that knowledge to other tasks is also an interesting research and might be a way to speed up subsequent design generation tasks. In combination with the Learner module, the Generator could use (DL) methods to find new ways to generate content. One such method is the “forward pass” method which involves the following process:

- Generate a collection of designs without performance evaluation.
- Train a model that learns an embedding of the latent space of the design collection generated.
- Use the pretrained model to guide generation in two ways:
    - unsupervised generation, where novelty can be defined as distance between designs in the latent space;
    - supervised generation, creating ‘nobs’ that allow the user to search across the dimensions of the latent space instead of the input space.

  The design of the proposed system enables the investigation of several of these ideas, some of which may become crucial for the generative design going forward.

References:

[BSK12]
Martin Bielik, Sven Schneider, and Reinhard König. Parametric urban patterns: exploring and integrating graph-based spatial properties in parametric urban modelling. In volume. 2012.

[BFH+11]
Jonathan Byrne, Michael Fenton, Erik Hemberg, James Mcdermott, Michael O’Neill, Elizabeth Shotton, and Ciaran Mcnally. Combining structural analysis and multi-objective criteria for evolutionary architectural design. In volume 6625, 204–213. 04 2011. doi:10.1007/978-3-642-20520-0_21.

[HK05]
Christiane M. Herr and Thomas Kvan. Using cellular automata to generate high-density building form. In Bob Martens and Andre Brown, editors, *Computer Aided Architectural Design Futures 2005*, 249–258. Dordrecht, 2005. Springer Netherlands.

[KB09]
Reinhard Koenig and Christian Bauriedel. Generating settlement structures: a method for urban planning and analysis supported by cellular automata. *Environment and Planning B: Planning and Design*, 36(4):602–624, 2009. doi:10.1068/b34025.

[LYT13]
Antonios Liapis, Georgios Yannakakis, and Julian Togelius. Enhancements to constrained novelty search: two-population novelty search for generating game content. In *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 343–350. 07 2013. doi:10.1145/2463372.2463416.

[LWW08]
Markus Lipp, Peter Wonka, and Michael Wimmer. Interactive visual editing of grammars for procedural architecture. *Association for Computing Machinery*, 2008. doi:10.1145/1360612.1360701.

[MC15]
Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. 2015. arXiv:1504.04909.

[NCC+20]
Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. House-gan: relational generative adversarial networks for graph-constrained house layout generation. 2020. arXiv:2003.06988.

[SFKonig11]
Sven Schneider, Jan-Ruben Fischer, and Reinhard König. Rethinking automated layout design: developing a creative evolutionary design method for the layout problems in architecture and urban design. In John S. Gero, editor, *Design Computing and Cognition '10*, 367–386. Dordrecht, 2011. Springer Netherlands.

[SSN+]
Mohammad Shaker, Mhd Hasan Sarhan, Ola Al Naameh, Noor Shaker, and Julian Togelius. Automatic generation and analysis of physics-based puzzle games.

[ST16](1,2,3)
Noor Shaker and Julian Togelius. The search-based approach. In Noor Shaker, Julian Togelius, and Mark J. Nelson, editors, *Procedural Content Generation in Games*, pages 17–29. Springer, 2016.

[SYT10]
Noor Shaker, Georgios Yannakakis, and Julian Togelius. Towards automatic personalized content generation for platform games. In *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010*. 12 2010.

[SJLW20]
Varvara Vasilatou Song Jie Lim and Shih Hsin Wuu. The use of ca to generate informal architectural systems. In *SimAUD 2020, Society for Modeling & Simulation International*. 2020.

[KaravolosLiapisYannakakis18]
D. Karavolos, A. Liapis, and G. N. Yannakakis. Using a surrogate model of gameplay for automated level design. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, volume, 1–8. 2018.

[LehmanStanley11]
J. Lehman and K. O. Stanley. Abandoning objectives: evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

Prophet

The Prophet module's role is to quantify design performance for all generated designs, that are meant to be evaluated. Prophet utilizes pretrained, surrogate deep learning (DL) models which transform what is typically a costly, simulation-based evaluation of various performance metrics into a direct evaluation that happens in real-time and requires, in most cases, only a simple visual representation of the generated geometry. It should be noted that performance, in the context of the Prophet module, strictly relates to design performance and not algorithmic performance.

Prophet is crucial to the proper functioning of the proposed generative system. The feasibility of the whole system relies upon the Prophet module, indeed the main reason that such a system has not been developed yet within the AEC domain is the inescapable cost of evaluations related with design performance. These simulations vary in run time and computational demands, ranging from a few seconds to a few days or even weeks of compute time, depending on the underlying complexity of the design and the metric evaluated. Since Prophet allows us to conduct these evaluations efficiently and in real-time the system can focus instead on generation and developing a truly expressive generative capacity. Prophet is also what transforms large-scale generative design from a primarily offline process to a potentially fully online generation.

The initial version of the system will support the following design performance evaluations:

- **Human Comfort** (HC), relating to thermal, wind, and visual aspects of comfort throughout the space.
- **Daylight Performance** (DP), relating to the distribution and availability of daylight throughout interior space and based on two yearly metrics, Daylight Autonomy (DA) and Useful Daylight Illuminance (UDI) and solar availability in exterior space based on two yearly metrics, Sunlight Hours (SH) and Solar Radiation (SR). This evaluation is climate-specific therefore location-specific surrogate models will be trained.
- **Natural Ventilation Performance** (NVP), relating to the efficiency of wind flow and ventilation throughout the space, measured in terms of flow (m/s) or volume (m3/s).

Each evaluation has its own specific scope and use and while the user should be able to select the metrics she wants for her generative experiment, the system will take care of suggesting appropriate evaluations and performance ranges according to the type of design task at hand.

Surrogate modeling for design performance evaluation is a potentially disruptive technology given the cost of evaluations in the AEC domain. However, there has been very little adoption in practice. The only tool utilizing pretrained models for real-time prediction currently developed with practice in mind is the Intelligent Framework for Resilient Design [2] (InFraRed). InFraRed is able to evaluate urban designs on a variety of important, and expensive, performance metrics, including wind flow and wind comfort. This research builds on the work done with InFraRed, trying to take the next step in the evolution of large-scale generative design which involves building a process that can actually take advantage of real-time performance evaluation.

While there is little work being done in the industry, there is a lot of research work done in surrogate modeling. Here it should be noted that InFraRed, the idea on which the Prophet module is built on, is not a traditional surrogate model since it is not being trained in real time using evaluations that are run through simulations (the standard process of surrogate modeling). That said it does serve the same function, hence it is closely related. Wortmann et al. [WCNS15] use radial-basis function surrogate modeling optimization as a method to promote understanding rather than optimization. The method trains a model by interpolating simulation data and explores the solution space in a way that prediction error reduces, within a set budget of evaluations (simulations). Westermann and Evins [WE19] conduct a comprehensive review on the application of surrogate modeling in the sustainable design domain. They find 57 studies that have been conducted and develop a comprehensive matrix of the methodologies used along with the problem the method it was applied. One interesting fact is that almost all of the surrogate modeling studies were conducted on energy simulation perhaps indicating one of the disadvantages of surrogate modeling, the requirement to run online simulations which for other metrics can take a considerable amount of time, and that requirement remains the same across different studies (no transfer learning).

The use of surrogate modeling has been very popular in QD, understandable given the large number of evaluations that QD methods require (typically in the order of 100,000). Gaier et al. [GAM18] use surrogate modeling and MAP-Elites as alternative to the classic optimization algorithms on a 2-dimensional airfoil optimization problem. They develop a new illumination algorithm, Surrogated Assisted Illumination (SAIL) that leverages surrogate modeling to create the map of the design space explored. Gaie et al [GAM17] show that SAIL provides a better performing solution than MAP-Elites and they also compare the QD surrogate modeling approach to a traditional CMA-ES algorithm. It turns out that in the same amount of evaluations required required for CMA-ES to optimize one cell of the design space (that has specific feature values) SAIL finds a near-optimal solution for every cell. In the Gaming AI domain, Karavolos et al. [KaravolosLiapisYannakakis18] use surrogate modeling for automated level design where they are able to investigate efficiently which level structures would result in a balanced match opening up new possibilities for mixed-initiative design.

The Prophet module is perhaps the most sensitive part of the system as it relies on external functionality (surrogate or pretrained models that conduct evaluations) for its performance. The main challenge here, especially in the AEC domain where all design tasks are real-life problems with real implications, is related to validation of prediction outputs and the ability to communicate error estimates to the user. It is crucial for the human designer to be able to trust the results of Prophet which makes the visualization of uncertainty (of prediction) a crucial part of Prophet's functionality. This uncertainty is a requirement for a mixed-initiative system of this kind to work. Another smaller, technical challenge, is to efficiently set up an inferencing pipeline that can manage both large design spaces and predict performance in real-time with a speed that can match generation and direct evaluation.

Despite all that, Prophet opens up a completely new field for generative design, where evaluation of metrics that actually matter becomes a reality. Prophet allows for essentially parametric design at generative scales and diversity, something that completely changes the way we conduct computational design. Another important aspect is that of lifelong-learning that can happen through the module. Each session the user takes part in generates a new batch of data that can be used as training inputs for model performance improvement. This continuous training can happen at different intervals, depending on user preferences, and can also be fully automated by the system. Prophet will also provide the opportunity to human designers to supply their own simulation data in order to produce new or better pretrained models. Finally, an exciting opportunity is the ability to extract latent representations of the design space not just in the input space but also in the performance space. For this, Prophet's results will be passed on to the Learner module where models will be trained to learn latent representations of the performance space.

References:

[**GAM17**]
Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Feature space modeling through surrogate illumination. *ArXiv*, 2017.

[**GAM18**]
Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient design exploration through surrogate-assisted illumination. 2018. arXiv:1806.05865.

[**WE19**]

Paul Westermann and Ralph Evins. Surrogate modelling for sustainable building design – a review. *Energy and Buildings*, 198:170 – 186, 2019. URL: http://www.sciencedirect.com/science/article/pii/S0378778819302877, doi:https://doi.org/10.1016/j.enbuild.2019.05.057.

[**WCNS15**]
Thomas Wortmann, Alberto Costa, Giacomo Nannicini, and Thomas Schroepfer. Advantages of surrogate models for architectural design optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 29(4):471–481, 2015. doi:10.1017/S0890060415000451.

[**KaravolosLiapisYannakakis18**]
D. Karavolos, A. Liapis, and G. N. Yannakakis. Using a surrogate model of gameplay for automated level design. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, volume, 1–8. 2018.

[2] www.infrared.city