Subversive machines:
designing architectural freedom through open systems.

Graduation document / MSc Architecture, Urbanism and Building Sciences.
Delft University of Technology, Faculty of Architecture, Urbanism and the Built Environment.

Explore Lab 28

2019 - 2020

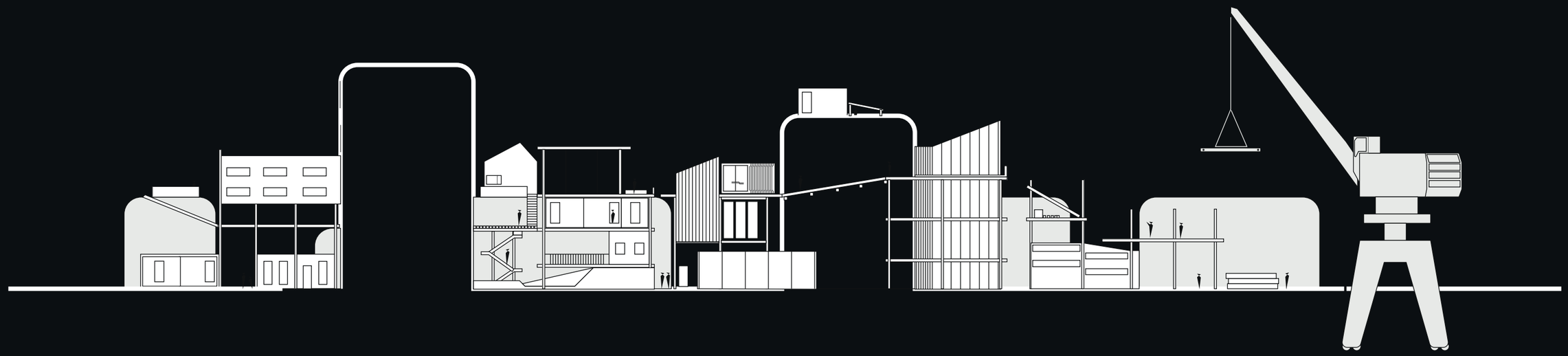
Preface

This document was produced to complete the MSc graduation process at the Delft University of Technology, Faculty of Architecture and the Built Environment., within the Explore Lab 28 graduation studio. The research paper and project that are shown in the pages that follow are my attempt at developing a method for an architectural design process that questions a number of deeply seated assumptions of what architectural intervention means -- in particular, it forms a continuation of a personal research programme that started during the programme's first thesis, in which I explored what it means for the architectural practice to truly open itself up to contemporary technological reality. The work in this document and the previous thesis are both underwritten by the desire to effect a change in the way that architecture interfaces with the places, peoples, objects and conditions that it encounters, one that might hopefully lead one day to a practice that is more sensitive to possibilities that lie outside the current status-quo.

Acknowledgements

I would like to express my sincere gratitude and appreciation to my graduation project advisors ir. Roel van de Pas, dr. ir. Stavros Kousoulas and ir. Hubert van der Meel for their insights, advice, recommendations and feedback. I would also like to thank Daan Vitner for facilitating the formal aspects of the graduation process and for his feedback during presentations.

Contents



Critical Technics in Architecture: a cybernetic approach.

Abstract *In this paper, I posit that for the field of architecture to come to a distinctly architectural application of computational technologies it requires the elaboration of a concept of critical technics. This is premised on a systems-view of technical development, which highlights the importance of time and situatedness for any consideration of change, genesis or becoming. In order to then construct an architectural technicity that can grapple with the external character of techncial development, I argue using the philosophy of technology of Gilbert Simondon and Stafford Beer's management cybernetics that what is needed for this is a radical opening-up of the architectural process in the form of a democratization, to augment architecture's capacity for producing alternate futurity.*

Keywords *Critical Technicity • Cybernetics • Technical Genesis • Concretisation • Systems-thinking • Architecture*

0.1 Non-market organization

In recent years, there has been a revisiting of the 20th century debate surrounding the viability of planned economies and the supposed necessity of market structures, in the face of a declining neoliberal world order and the emergence of new kinds of techniques for processing information that can be argued to provide an alternative to market structures[1]. However, this is an insight that has by now informed a number of different views on alternative techno-social principles of productive coordination that are not premised on utilizing price signals for resolving questions of organization, distribution, and agency (Srnicek, 2016, pp. 54, 55) – ranging broadly from seemingly progressive surveillance-technocracy capitalism to especially authoritarian forms of neoliberal capitalism that can both be said to have "broken free of the shackles of democracy" (Žižek, 2015) through the application of new computational technologies. There is therefore a sense in which research concerning data gathering and sensing techniques is arguably tied to a tendency toward different (yet presumably equally un-equal) forms of productive, distributive and social coordination (see Zuboff, 2015). With this development comes the emerging possibility for a moment of reconfiguration that relates to how these questions are dealt with. Arguably, the main issue with this observation is that the horizon of that reconfiguration is limited to a very narrow, ideologically defined window of change, dominated primarily by the notion of surveillance capitalism (Zuboff, 2015).

0.2 Formal Complexity

In keeping with this larger tendency, the field of architecture currently lies at the end of its first digital turn, in a nascent so-called second digital turn (Carpo, 2013, 2017). Digital technologies have taken on an increasingly important role both as themes within design problems and within the design process itself. There is a rich history of cutting-edge computational techniques and insights being applied in architectural design processes in recent history – starting from the first experiments at applying chaos theory and complexity theory by figures such as Eisenman and Jencks, through figures like Christopher Alexander and Cedric Price and their early forms of patterned and generative architecture, and leading eventually to the iconic parametricism of architects such as Zaha Hadid (Bachman, 2008; Rega & Settimi, 2010; Spencer, 2016).

This tradition, although certainly more varied than here presented, seems to have concerned itself primarily with the application of the notion of complexity to aesthetic questions – what we might call formal complexity (Rega & Settimi, 2010). Arguably, this process reflects what

1 This is evidenced by a number of recent popular and academic publications that highlight the restructuring of contemporary capitalism such as Mason, 2015; Mayer-Schönberger & Ramge, 2018; Phillips & Rozworski, 2019; Srnicek, 2016; Zuboff, 2019.

Stafford Beer referred to as using a computer to do quill-pen administration[2]:

"[...] we insist on retaining [...] those very limitations of hand, eye, and brain that the computer was invented precisely to transcend." (1974, pp. 32–33)

Rather than applying digital technology to solve problems in a similar but more expedited way compared to traditional methods, Beer argues that the logic of computation demands a reframing of how we think of problems. Instead of applying computation as an administrative tool, it allows for exploring reality in a different way – through modeling, computation opens up new approaches to problem-solving that allow one to interface with multi-causal, complex realities. In some sense, Beer argued for what has become known by now as a general ecological approach (see Hörl, 2017) to computation. In a previous thesis, I argued that the dominant paradigm of 'digital architecture' as it existed for the first digital turn had a number of inherent limitations that relate to the way in which digital tools structure our experience of reality (Mellas, 2018) – these limitations resemble Stafford Beer's remark quoted previously. For Beer, this was a prompt to come up with a different way of using computation – that attempt led him to conclude that what is paramount for any system to even be viable is that it is democratically regulated. Worker control was for Beer the key to avoid catastrophic failure for societal institutions in the face of a changing material environment – a radical cybernetic approach to organizational strategy as an adaptation to what seemed in his eyes an inevitable collapse of the institutions of mid-20th century state capitalism.

The discrepancy between this approach and the application of digital techniques in architecture as practiced today by itself might already be categorized as a general problematic, purely because it can be taken to mean that the field of architecture has not yet come to grips with contemporary technological reality and the opportunities it provides for rethinking how problems are constituted and, more crucially, what appropriate approaches to these problems entail. However, this general observation points towards a much deeper and consequential problematic for the field – namely that through a lack of understanding of these technologies, architecture loses its capacity to mediate how they are applied within the built environment, and in turn the possibility for architects to engage critically with these developments, from their own specific expertise and concerns. This highlights the relevance of coming up with a new framework for applying computation within architectural design. To keep up with technical development, and thus to stay relevant as architects, it is crucial that we elaborate on how architecture can critically incorporate digital technology into its activities as a field – rather

2 The creation of complex geometric patterns and forms has been a central pursuit for many architectural designers throughout the history of the practice, often with very successful and highly complex outcomes in terms of ornamental design. Contemporary design practice in this sense uses digital computational technology in a way that differs little from how it has used pen and paper throughout history – for drawing, geometrical construction and classical calculation.

than allowing the structural mechanisms[3] that underlie much of the development of these technologies to dictate what is and what is not relevant in today's built environment.

Furthermore, in recent years a number of projects have emerged that explicitly intend to subsume architectural and urban design to the creation of new markets through intensive data gathering, guided by the concept of the Smart City (Greenfield, 2013). At the basis for these developments is the underlying ideological assumption that the future built environment will be privately owned and operated, including its virtual and physical infrastructure (Greenfield, 2017; Poole & Shvartzberg, 2015), a move toward a form of surveillance capitalism in keeping with the previously described horizon for change in economic control.

The point of this paper is to demonstrate that the only way to harness the emancipatory and productive potentials of computational technology in architecture is through a general socialization of the architectural process. This would allow architects on the one hand to circumvent the commodification of architectural form and on the other to retain a distinctly architectural sphere of influence around the application of digital technologies within the built environment. More fundamentally, it could provide architects with a method to contribute to a futurity that defers from contemporary capitalist realism, through an architectural form that presents itself as a form of realist intervention which can re-organize itself toward particular desired futurities.

0.3 Critical computation

The tendencies described in the previous paragraph call for an examination of the way in which technology is used within architecture. To that effect, this paper proposes that we rethink the role of architecture in the application of technology and the role of technology in architecture. I will relate this to one set of digital technologies in particular, that can broadly be categorized as computational design. This might be rephrased as conceptualizing how computational design techniques can be used critically. The word critically is used here to refer to a capacity to generate alternatives – a critical use of technology, then, is the application of a technology in such a way as to engender alternative paths of development that are not necessarily limited to the logic of contemporary capitalism.

From this it becomes clear that it is necessary to dispense with the notion that technology is inherently geared towards particular value systems, what might be called a substantive theory of technology. Instead, using the work of Gilbert Simondon later authors that subscribe to the

3 This refers in particular to the external nature of many of these developments to architecture – while there are many architects attempting to 'design their way around' technologies that are in development, their original articulations and manifestations are presumably not elaborated by architects in most cases. This leaves any architectural application as an appropriation of existing invention, and thus risks both shoe-horning technologies into architectural practice, and unwarranted solutionism (see Murphy, 2012).

same position, I argue in the first section of this paper for a relational approach to technical development – one based on systems thinking and a particular strand of cybernetics. The reason for this is twofold: it is only through an open-ended conception of technical development that one can arrive at any meaningful formulation of an alternative kind of technicity, rationality, or future. Secondly, the previously described ideological premises for contemporary projects that deal with computational design, and the growing tendency to position architecture as a field for data-gathering within surveillance capitalism together present a certain urgency for architects to develop a grounded position from which to formulate an alternative application of these technologies. Envisioned as argued previously, this is something that a substantive theory would simply not allow for. Instead of resignation, we would do well to say that architectural value "is too valuable to be left to capital", echoing Brian Massumi (2018, p. 2). As such, I posit that through the literature on cybernetics – it can be argued that a further integration of sensor technology into the environment likely will not contribute to the overcoming of so-called technical alienation within the built environment. Moreover, in the second section I present the claim that what is vital for any architecture that is premised on generating an emancipatory futurity through computational technology is a reorientation of the technicity of the built environment towards the notion of embedded intelligence in a distinctly politicized and socialized form.

1.1 Technical development

Gilbert Simondon describes the development of technics[4] as the shaping of a technical object towards (internal) functional demands. This is referred to as a kind of self-sufficiency – the technical object "unifies itself internally" towards being a concrete technical object (Simondon, 2012, p. 26). This is an abstract process, where a technical object's constitutive components become more and more interoperable over the course of their development (Iliadis, 2013, pp. 15–16) through "concomitance and convergence" of multiple, different functions into singular multipurpose structures (Simondon, 2012, p. 28). Technical objects, for Simondon, behave as evolutionary beings that mutate toward their own inherent fitness curve – the key difference in this regard between natural (living beings) and technical objects (artificial beings) then, is that the former already exist as concrete objects (Simondon, 2012, p. 51). What is crucial in Simondon's terminology is that the term technical object does not refer to one specific object in space. Instead, it is a more abstract term that refers to a set, or branch, of technologies – such that one would say all attempts at building a combustion engine are part of one unitary, abstract combustion engine.

4 *Technics* refers here specifically to technique, as opposed to the more general English term technology which may refer to technique, technical objects and the study of technical objects (see Iliadis, 2013).

It is concretization, for Simondon, that informs the primary path of formation that technologies take, in turn even spawning new branches for other technologies over the course of their development. Simondon's philosophy of technology allows us to think of technicity as an open-ended, but structured process, bound to its own internal logic of coherence.

1.2 Technical control and culture

But what does Simondon have to say about the external factors that constitute this process, the associated milieu of the development of a technology? Within fields of research that study the development of technology, there are a number of theories that seek to explain how technologies are construed – the clearest division here lies between what might be categorized as a constructivist theory of technical development, and an instrumentalist theory. It is relevant to combine a reading of Simondon with the work of Andrew Feenberg, particularly his concept of the technical code. For Feenberg, a technology is a scene of struggle between the workers or operators of a technology, and those who manage it – both have their own connotations with a technology and its development, and thus their own requirements and demands of that technology. Feenberg, in this sense, follows Bruno Latour's formulation of a "parliament of things" (in Feenberg, 2002, p. 30). Contrary to Latour, however, Feenberg identifies that there is no leveled-off network of actors without power or hierarchy – instead, political struggle is inscribed in the way a technology manifests over its lifetime. What is stressed here is the ambivalence of technology -- as a process, not a thing. Feenberg describes technology as a structure that develops over time and is influenced from myriad directions – a relational account that resembles Simondon's notion of modulation. This leads Feenberg to the conclusion that what is needed is to democratize technical development through "a shift in the locus of technical control" (Feenberg, 2002, p. 32).

The development of technology is underwritten by the way in which it encodes a particular cultural configuration – Feenberg argues that it is in fact here that technology can serve as a way to cement or lock-in emancipatory views in society. After this, it becomes part of the way things nominally are – as a new kind of norm. This constitutes an affective dimension to technical development where it is the imaginaries and visions that a technology brings into the world that create meaningful contributions on a cultural level. Feenberg stresses that it is through this locking-in of imaginaries that the coherence of societal alternatives might be demonstrated and in turn made business as usual (Feenberg, 2002). This could be rephrased in Simondonian terms as saying that what matters for Feenberg is the associated milieu that is created through technics. Invention is the process wherein the information contained in this milieu is transduced into a new technical schema – it is passed on as a form of transindividual knowledge (Simondon, 2012, p. 252). Feenberg then, offers us through Simondon a way of conceptualizing technics in a critical way: through modulation of an environment one might influence the constitution of future technics.

This is a useful way of formulating a notion of criticality in light of technology as a field of political struggle – what is needed then is a way of orienting this modulation towards particular alternatives. What Feenberg points to is the asymmetry of the political arena within which this modulation takes place, centering the notion of a technical class struggle in line with traditional Marxian analysis. However, with his concepts Feenberg is at first glance concerned primarily with resolving the apparent contradictions between reified notions of culture and technology through his notion of a technical culture -- his concept of the technical code is ostensibly cultural, a code between participants in society. However, beyond the cultural level, there are internal dynamics and logics that govern how processes unfold within the world. While there likely exist a number of these logics that do have some cultural expression or even take place on the cultural level in their totality, it seems lacking to restrict one's analysis only to this. This means that rather than modulating the operations and structures that constitute technical objects, it is needed to examine how one might go about modulating the logics that govern their genesis – the formulation of a metalogic.

1.3 Systems-view and futurity

Both Feenberg and Simondon describe the genesis of technology as a system in all but name – consisting of codes, rules and logics that govern the specifics of a technology's coming-into-being. One way of making this further explicit is by generalizing the common conception of technical development, as a linear process from point A to point B, into a multidimensional field – where it is the logics that govern the topology of the space of possible outcomes that a particular technology might follow. As Marx and Engels posit in Capital, the conditions of a movement beyond capitalism "result from the premises now in existence" (in Thoburn, 2003, p. 3). When discussing these conditions in relation to technology from a Marxian standpoint, the process in which these technologies are produced and the way in which they are integrated into processes of social (re)production take on central importance.

We might interpret this in a way that lends itself to Simondonian terminology: it is only when present organizational, and technical conditions reach a metastable state, one of oversaturated potentiality, that transduction into new forms of organization can take place. A key component of the notion of transduction is that it is a transmission of information through material – this is the central thesis of Simondon's work on individuation against hylomorphism and the place where his concept of modulation comes in. As such, one might more precisely state that this transduction relies on specifically material encodings of organizational forms. Bernard Stiegler (2017), following Simondon's work on technics and mechanology, argues that this takes place through the genesis of technical systems. Through internal evolutionary tendencies, technical systems induce internal changes, which necessitate socio-technical changes on other levels of societal becoming. Stiegler notes that "these adjustments constitute a suspension and a re-elaboration of the socio-ethnic programs or socio-political programs that form the unity of the

social body" (Stiegler, 2017, p. 130). This view, which Stiegler terms organology, underscores the fundamental connections that exist between technical and social systems. As such, Stiegler's work serves to emphasize a point that is central to this paper: that there exists a reciprocal relation between technical systems and social systems – both systems forming part of one another's associated milieu.

Augmenting the previously described conception of a critical technology through Feenberg with Stiegler's organology points clearly towards a logic that takes place on a separate level from the cultural. In a sense, Feenberg's notion of a critical technology is a form of socially mediated but unidirectional technical genesis: effecting changes in an environment with the aim of changing future technicity. Stiegler argues that these changes in technicity have the potential to be foundational beyond the ways that Feenberg describes – implementing not

just imaginaries of alternatives, but in fact generating a localized reconfiguration of the social-political domain. Beyond this, it can be argued that it is technicity itself that enables the concept of futurity – it is through inscription[5] that a reference point can be retained, without which one would be limited to experiencing a present (Colebrook, 2016).

To Stiegler, this relies on the premise that ways of thinking are informed by technical conditions: as such, technical objects can be said to create their own particular subjectivity in those that are subject to their use. A psycho-social individuation takes place through technical objects, which then contributes to collective ways of thinking, thus constituting a circuit of transindividuation (Stiegler, 2017, p. 137). Following Simondon, Stiegler argues this proceeds through the spatialization of temporal forms of reason, which today can be said to take the shape of data-gathering through sensing technologies. However, this is primarily a one-way process as well: surveillance technologies impose a particular subjectivity, but the private ownership of these systems and their black box nature stemming from that private character do not allow for any reciprocal influence on the logics that govern these technical objects (Zuboff, 2015). Where they do, this influence is mediated through an internal tendency toward technocratic barriers – a sufficient level of understanding of and engagement with ambient sensor technology is often required to even have an overview of its capacities and features, and thus, to conceptualize how it might be applied, changed, hacked or adopted. Arguably, this amounts to a cutting off of so-called smart systems from paths of individuation that take place through struggle, transindividuation or democratic control.

This line of thought is compatible with contemporary Marxian views on processes of subjectification that take place under capitalism[6] – in particular, they resonate with the notion

5 Stiegler's retentions. Similarly, for Feenberg, within technological historicity, this happens through the technical code.

6 I am here refering specifically to what is commonly known as Value-Form theory (see González, 2019).

that different technical (and thus (re)productive) conditions generate different emancipatory goals, subjects and processes, beyond traditional class-based understandings. In contrast to Feenberg, this is a decentering of class struggle as the main engine of technical genesis. Instead, this view relies on the notion that what has changed fundamentally since Marx's time is that there is no more concept of a universal, trans-historical emancipatory subjectivity to speak of – as such, one arrives at a theoretical vantage point where different, distinctly historical subjectivities carry their own potential for an idiosyncratic emancipatory futurity. This is an argument that opens up a critical capacity, as defined earlier in this paper. Fundamentally, this position comes with a number of consequences attached. Primarily, this implies an opening up of futurity – not merely beyond transhistorical notions, but in addition beyond what might be referred to as a "residual linearity and humanism" (Colebrook, 2016, p. 13).

In summary, this section has described so far how technical development possesses potentials – it can occur across a multitude of paths. As such, it produces what one might term outcomes, which are contingent on material conditions within an environment which determines the limits of technical potential. In cybernetic terms, this amounts to the description of a system.

2.1 Complexity and variety

Considering technical development as a system opens up a number of avenues of investigation, primarily by allowing one to specify further how that system might be influenced and to ask from which loci and through which logics this might proceed to shape technical genesis toward desired outcomes. This would result in a critical system of technics that takes on the form of a regulator, in traditional cybernetic terms (see: Conant & Ashby, 1970). To characterize this critical system I refer to Stafford Beer, one of the cyberneticians part of the second generation of British cybernetics (Pickering, 2010). Beer's work differed from many of his more commonly referenced peers in that he placed emphasis on the relation between what amounts to an organizational system's relative democratization[7], and its ability to function in the face of complexity (Swann, 2018). As such, Beer represents what Stiegler describes as "the new bases" of cybernetics (Stiegler, 2017), as opposed to the popular conception of cybernetics as a military, controlling technicity that is more commonly associated with Norbert Wiener.

Beer offers us a compelling line of reasoning to reject the data-driven paradigm of digital computation that drives on a logic of representation: digital machines "are pre-occupied with access" (in Pickering, 2010, p. 235). This is in reference to the fact that control-systems, the predecessors to contemporary digital systems, were built to generate intermittent output, in

7 This is not Beer's term as used in his technical writing – instead, Beer refers to a kind of autonomy at different levels within an organisation, so that decisions can be decentralized, this was the basis for his management theories and models, where too much hierarchy is seen as inhibitive to the self-organising capacity of a system (see: Pickering, 2010; Swann, 2018).

the form of print-outs, during a process of computation. The result of this is a paradigm of computation that is charged with getting representable answers to questions, whereas the most important result of a computational system in the cybernetic view is performative. As Pickering emphasizes, it is the navigation of a field without a representative mapping of it, as with a steersman (Kubernetes) navigating toward a distant light on the shore through incremental adjustments. Pickering characterizes the demand for overview in terms of representative models aptly as "an enormous detour [...] into and through a world of symbols" (2010, p. 235).

Instead, Beer's position towards hylozoism and the agency of matter seem more in line with Simondon's concept of modulation – both presuppose that material itself can facilitate an operation without a subjection of matter to form, and without the imposition of an ideal, or blueprint that precedes this emergent process of in-formation. For Simondon, this is primarily observed within the development of technics according to its own logic, for Beer, it is organizations of people that self-organize. By looking for appropriate types of matter already in existence, one can engage in the world as it is offered (Pickering, 2010), and thus engage it in a relational way. Furthermore, it is for Simondon precisely this attitude of considering an object within its milieu, that opens up the space of what is possible – its field of potential. Simondon develops a convincing argument for technicity that is thoroughly embedded in its associated milieu by way of concretization – he demonstrates that it is through a synergy between a technical object and its environment that new potentialities can be rendered accessible, as with the example of the Guimbal Turbine (Simondon, 2012, p. 57).

Ultimately, a seemingly similar line of thought leads for Beer to an ambition to formulate a paradigm of biological computation (Pickering, 2010, p. 231), as something radically distinct from what is conventionally seen as computation, even today – as a form of computation that relies on ecological systems that are found as they are in the world. He arrives at this through his concept of exceedingly complex systems, arguing that while our representational logic cannot meet the variety[8] in these systems with adequate reciprocal variety (Pickering, 2004), another naturally complex system such as the complex system of a pond found in nature might. Here it is important to note that the notion of regulatory variety matching system variety that Beer inherits from Ashby[9] resembles a process of adaptation within a system to its milieu much like the genesis of technology is to Simondon. Simondon has been described as a proto-cybernetician – as such there is a number of similarities between his work on technicity and that of the later cyberneticians such as Ashby, Pask and Beer (Feenberg, 2019; Pickering, 2002, 2010).

8 Variety, in cybernetic terminology, is defined as the amount of possible states or outcomes that a system has (S. Beer, 1983)

9 See: Ashby, 1991; Conant & Ashby, 1970

How does this concept of variety fit in with contemporary paradigms of computational technology within the field of architecture? For some, by taking contemporary technics in the direction of 'animate knowledge' where one might argue that we have today the technical means to animate our inanimate surroundings through ambient sensor technology (by now mostly garnered under the concept of Big Data), meaning we might overcome the alienation of technology (Leach, 2018). This amounts to a strategy of matching what one might call natural variety with technical variety – it is implied that this technical variety would somehow amount to the level of variety that occurs in living systems by the choice of words. Notably, this is a move that follows the principles described so far – in animating an environment through ambient technology, a designer intervenes in the milieu of a system, changing the terms on which interaction between systems take place. Through Ashby and Beer's line of reasoning it could however be argued that it is precisely this impulse to seek greater and more complex technics that affects technical alienation through the inadequacy of technical variety in matching living system variety (which, to Beer, stands apart as exceedingly complex) and the corresponding need for reduction and normativity to keep the technical system viable in light of this discrepancy. This amounts to an asymptotic complexification: a greater and greater animation of technical systems, that might eventually approach exceeding complexity, but for the foreseeable future remains distinctly lacking in variety.

2.2 Metastability

If one follows Beer's categorization, the discrepancy between complexity and exceeding complexity outlined in the previous paragraph points toward a certain limit with regards to how well technical systems might interface with their environment. While this paper has so far highlighted a number of similarities between Simondon and Beer's work, there are also key differences that become evident particularly with this limit in mind – one of which is particularly relevant for this paper: as Mills (2015) argues, Beer and others working within the tradition of his Viable Systems Model[10] do not describe a mechanism that accounts for novelty in complex systems – by basing their model on homeostasis and ultrastability, there is little room left for a concept of invention. Accordingly, this view of social organization works only when one assumes that all interactions are probabilistic – a "removal of the indeterminism and novelty from the domain of the social" (Mills, 2015, p. 6). He further argues that this amounts to a disregarding of politics and in turn a favoring of technocratic logic, as politics is precisely the mechanism that resolves indeterminism in the social domain.

What this highlights most of all with regards to the main question of this paper is the importance of invention – systems that evolve through metastability rather than the more commonly

10 Mills refers specifically to Alex Pentland.

described concept of equilibrium stability. This means going, as Simondon describes, beyond being "enslaved by the finality of the whole" (Simondon, 2012, p. 119) through unremitting re-organization. Another way of describing this is as self-production (autopoiesis[11]), rather than solely self-reproduction (or self-regulation). Autopoietic systems can be categorized as systems that can re-inform their internal configurations: through metastability, these systems have the capacity to generate new states, and as such are continuously in a state of becoming, rather than being. This brings us back to Feenberg: his conception of a critical technology relies on a capacity for reorganization which lies within the political. Bearing the notion of autopoiesis in mind, this can be rephrased as centering the decision-making (and thus informing) capacity of social processes. This points in the direction of a distinctly politicized cybernetic approach to technicity.

2.3 Radical cybernetics

In the previous sections I have argued that one can characterize technical development as a system, as such, it is a contingent process that is embedded within an environment – most concretely in terms of the limits to potential, in terms of what is considered possible, and in terms of what is viable. Moreover, as I have previously argued, considering technics as a system means accepting that it is fundamentally political in nature – for social systems, their capacity for informing is related to the degree to which a system can resolve indetermination. This is in turn tied to the level of complexity that a system holds. In order to interface with the exceedingly complex, autopoietic nature of the built environment then, there is a sense in which current models of architectural practice fall short.

This becomes particularly clear when one considers the notion of failure, and its relation to invention and reorganization. Stafford Beer's original work on cybernetics hinged primarily on the notion of viable systems – autopoietic systems that can retain their functioning in light of any environmental change, and as such, in failure have the capacity to overcome that failure through a reorganizing capacity. For this, a system has to sacrifice its direct functionality in the following way: a system that is narrowly functional is limited to a very specific given set of rules, when these rules no longer manage to adequately enable the system to interface with its environment, it fails (Bates & Bassiri, 2016, pp. 205–206). Crucially, the specificity of a system's rules constrains the complexity of the system – meaning that it cannot meet the variety of its environment. As such, the point can be made that for a system to be viable, it has to have a level of plasticity – it has to be able to reorganize its governing logic in light of environmental change (Bates, 2016). This is a point that Simondon elaborates on more fully: it is not just that functionality negatively impacts a system's plasticity, but more generally, that it is through a

11 See: Varela, Maturana, & Uribe, 1974

greater level of abstraction away from functional demands that a technical object is made open to multifunctionality – and thus further concretized (Kousoulas, 2018).

The conclusion of this argument is that for any meaningful concept of change, and thus futurity, to arise[12] a system has to be specifically porous in its governing logic – especially with regards to its environment. In the context of organizational systems, where this interaction fundamentally relies on humans, this can be taken to indicate that what is crucial for any sort of autopoietic property to arise is a direct relation between the subjects of these processes and the system that is being designed if one intends to engender a critical form of technics.

This reasoning can be extended by looking at contemporary literature on the research into artificial intelligence – currently, there is a growing acknowledgement of what might be referred to as embedded cognition or situatedness, and its importance in nurturing any intelligence toward greater levels of complexity that is indebted to Stafford Beer's work, influencing both the dominant paradigm in AI and coincidentally contemporary cognitive science (R. D. Beer, 2014; Froese, 2007). What is relevant to this paper is that there is a sense in which current paradigms of cognition and (artificial) intelligence recognize the importance of material (re) organization in shaping systems' behaviors through the concept of autopoiesis (R. D. Beer, 2014). Moreover, it has been argued that any venture into the creation and maintenance of general intelligence systems seemingly has to rely on a distribution, and thus exteriorization, of intelligence (Pasquinelli, 2015). This "offloading of our cognitive processing into the environment" (R. D. Beer, 2014, p. 131) is what allows an understanding of intelligence as a distributed phenomenon – a process that takes place through a network of technical and biological individuals in the Simondonian sense.

Architecture and a critical technicity

Returning to the central question of this paper then, it might be argued that one way of modulating the outcomes of technical development lies with this environmental porosity and its relation to cognition as a network of technical and biological individuals. Within an architectural context it is important to emphasize that this environment consists in more than purely the physical boundaries and objects that surround an intervention – instead, the broader positioning of an object within its physical, ideological, technical and social context defines an overarching system-environment that a buildings occupants interface with during their stay in, or use of, a building.

Crucially however, the component that takes this architectural environment beyond the traditional notion of an architectural context, as is commonly used to refer to these aspects, is

12 Beyond the aforementioned necessity for retentions in Stiegler's terms.

the importance of its change over time. By foregoing the nature of architecture as a process that unfolds over time, I would claim that architectural practice is relieved of discussing and perhaps even conceptualizing this part of an intervention. As such, one might argue that architecture is lacking a form of retention which would enable the formation of a critical technicity in the built environment.

This is especially evident if one considers that the transmission of architectural design intentions relies first and foremost on static images – snapshots of an intervention's lifetime, often limited to the image of a newly built structure. One might thus posit that architecture as a discipline in its current form has no memory-for-time that enables designers to grapple with these questions and to participate in the shaping of futurity when it comes to the lifetime of the building in any conscious manner.

Furthermore, as I have outlined in the previous chapters, the notion of development in se is premised on change over time – for any meaningful conception of a technical development within the built environment itself, and not external to it, a centering of this understanding of architecture as a system is required, and thus, an understanding of the architectural intervention as a continuous moment – a proceeding intervention. Crucially, one can then consider the aforementioned processes of invention and individuation of technics within the architectural process. Within that process this position allows one to appreciate the necessity of a relational approach if one's aim is to in-form a particular emancipatory futurity due to the role that plasticity and situatedness play in enabling this potential.

It is here that I refer back to the notion of criticality that I deploy in this paper. If one's intention is indeed to maximize the multiplicity of emancipatory outcomes that a system can generate, then due to the nature of the process of in-formation being premised on the resolution of indeterminism, a key role in this system lies with its integration with one particular aspect of its environment – namely the biological entities that occupy them. A critical technicity within architecture then, is one that is premised on a politicized architectural process – providing the capacity for the emergence of new rules and logics that follow from reconfigurations of the unity that defines the total relation between building, environment and user. A dance of continuous reinvention on the part of both architectural intervention and occupant – a "technicity that determines the potentials of a shared becoming" between technical and physical individuals (Kousoulas, 2018, p. 6).

References

Ashby, W. R. (1991). Requisite Variety and Its Implications for the Control of Complex Systems. In Facets of Systems Science (pp. 405–417). Boston, MA: Springer US.

Bachman, L. R. (2008). Architecture and the four encounters with complexity. Architectural Engineering and Design Management, 4(1), 15–30.

Bates, D. (2016). Automaticity, plasticity, and the deviant origins of artificial intelligence. Plasticity and Pathology: On the Formation of the Neural Subject, 086, 194–218.

Bates, D., & Bassiri, N. (2016). Plasticity and Pathology: On the Formation of the Neural Subject. Fordham University Press.

Beer, R. D. (2014). Dynamical systems and embedded cognition. In K. Frankish & W. M. Ramsey (Eds.), The Cambridge Handbook of Artificial Intelligence (pp. 128–151). Cambridge University Press.

Beer, S. (1974). Designing Freedom. John Wiley & Sons, Ltd.

Beer, S. (1983). The Will of the People. The Journal of the Operational Research Society, 34(8), 797.

Carpo, M. (2013). The digital turn in architecture 1992-2012. Wiley.

Carpo, M. (2017). The second digital turn: design beyond intelligence. MIT Press.

Colebrook, C. (2016). Futures. In B. Clarke & M. Rossini (Eds.), The Cambridge Companion to Literature and the Posthuman (pp. 196–208). Cambridge: Cambridge University Press.

Conant, R. C., & Ashby, W. R. (1970). Every good regulator of a system must be a model of that system. International Journal of Systems Science, 1(2), 89–97.

Feenberg, A. (2002). Transforming Technology: A Critical Theory Revisited. Oxford University Press.

Feenberg, A. (2019). The Internet as network, world, co-construction, and mode of governance. Information Society, 35(4), 229–243.

Froese, T. (2007). On the role of AI in the ongoing paradigm shift within the cognitive sciences. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 4850 LNAI, pp. 63–75).

González, E. (2019). From Revolution to Democracy: The Loss of the Emancipatory Perspective. In E. González, A. C. Dinerstein, A. G. Vela, & J. Holloway (Eds.), Open Marxism 4 (pp. 155–167). Pluto Press.

Greenfield, A. (2013). Against the Smart City. Verso.

Greenfield, A. (2017). Radical Technologies: The Design of Everyday Life. Verso.

Hörl, E. (2017). Introduction to general ecology: The ecologization of thinking. In E. Hörl & J. Burton (Eds.), General Ecology: The New Ecological Paradigm (pp. 1–74). Bloomsbury Academic.

Iliadis, A. (2013). Informational Ontology: The Meaning of Gilbert Simondon's Concept of Individuation. Communication +1 , 2(September), 1–18.

Kousoulas, S. (2018). Shattering the black box: Technicities of architectural manipulation. International Journal of Architectural Computing, 16(4), 295–305.

Leach, N. (2018). Adaptation. In A. Graafland & D. Perera (Eds.), Architecture and the Machinic: Experimental Encounters of Man with Architecture, Computation and Robotics (pp. 46–59). DIA Architecture School.

Mason, P. (2015). Postcapitalism: A Guide to Our Future. Allen Lane.

Massumi, B. (2018). 99 theses on the revaluation of value: a postcapitalist manifesto. University of Minnesota Press.

Mayer-Schönberger, V., & Ramge, T. (2018). Reinventing capitalism in the age of big data. London: John Murray Publishers.

Mellas, Z. (2018). Critical Digitalism: Instrumental Reason and its Limitations. (Unpublished master's thesis). Delft University of Technology, Delft, Netherlands.

Mills, S. (2015). Simondon and Big Data. Platform: Journal of Media and Communication, 6: pp. 59-72

Murphy, D. (2012). The architecture of failure. Winchester: Zero.

Pasquinelli, M. (2015). Alleys of your mind: augmented intelligence and its traumas. Lüneberg: Meson Press.

Phillips, L., & Rozworski, M. (2019). The People's Republic of Walmart. Verso.

Pickering, A. (2002). Cybernetics and the Mangle: Ashby, Beer and Pask. Social Studies of Science, 32(3), 413–437.

Pickering, A. (2004). The science of the unknowable: Stafford Beer's cybernetic informatics. Kybernetes, 33(3/4), 499–521.

Pickering, A. (2010). The cybernetic brain: sketches of another future. University of Chicago Press.

Poole, M., & Shvartzberg, M. (2015). The politics of parametricism: digital technologies in architecture. Bloomsbury.

Rega, G., & Settimi, V. (2010). Nonlinearity in architecture versus science: Borrowing the lexicon of complexity or exploiting its powerfulness? In P. J. S. Cruz (Ed.), Structures and Architecture (pp. 167–174). London: Taylor & Francis Group.

Simondon, G. (2012). On the mode of existence of technical objects. (C. Malaspina & J. Rogove, Eds.). Minneapolis: Univocal Publishing.

Spencer, D. (2016). The architecture of neoliberalism: how contemporary architecture became an instrument of control and compliance. Bloomsbury.

Srnicek, N. (2016). Platform capitalism. Wiley.

Stiegler, B. (2017). General Ecology, Economy, and Organology. In E. Hörl & J. Burton (Eds.), General Ecology: The New Ecological Paradigm (pp. 129–150). London, New York: Bloomsbury Academic.

Swann, T. (2018). Towards an anarchist cybernetics: Stafford Beer, self-organisation and radical social movements. Ephemera, 18(3), 427–456.

Thoburn, N. (2003). Deleuze, Marx and politics. Routledge.

Varela, F. G., Maturana, H. R., & Uribe, R. (1974). Autopoiesis: The organization of living systems, its characterization and a model. BioSystems, 5(4), 187–196.

Žižek, S. (2015, February 1). Capitalism has broken free of the shackles of democracy. Financial Times.

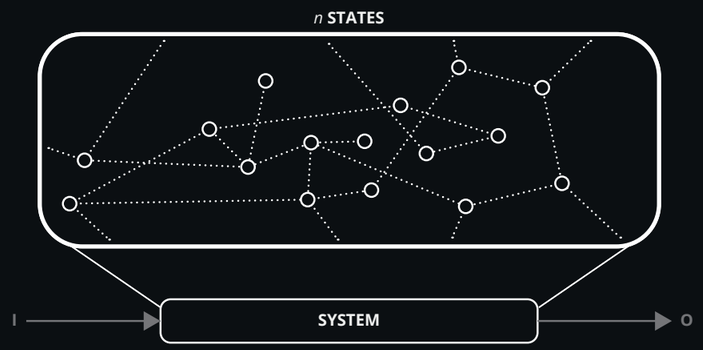
Zuboff, S. (2015). Big other: Surveillance capitalism and the prospects of an information civilization. Journal of Information Technology, 30(1), 75–89.

Zuboff, S. (2019). The age of surveillance capitalism: the fight for a human future at the new frontier of power.

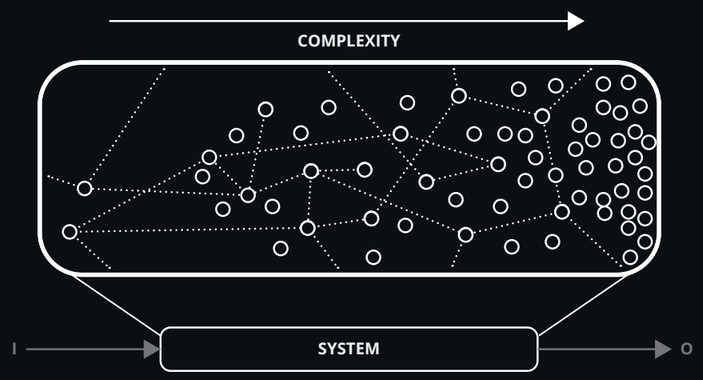
Architectural complexity: systems & environments.

Variety, Complexity

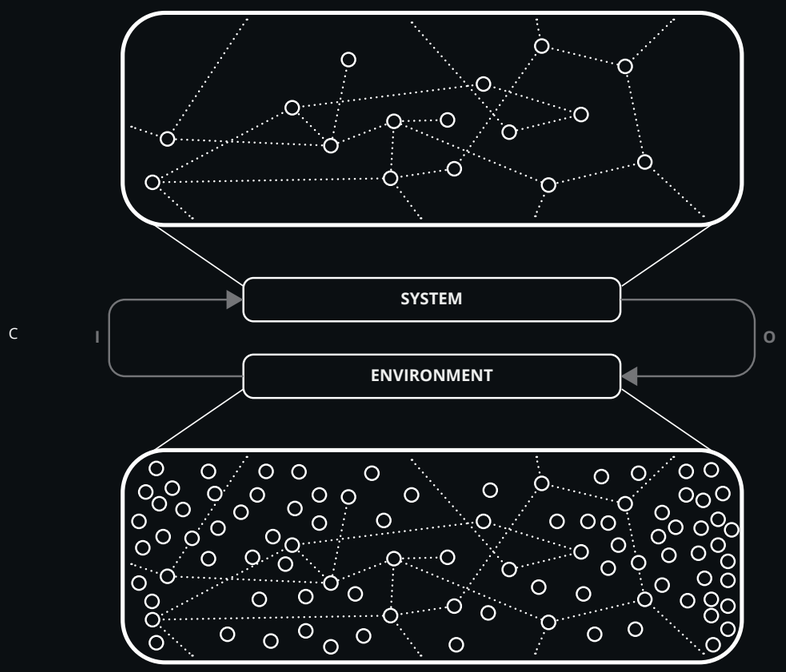
A



B



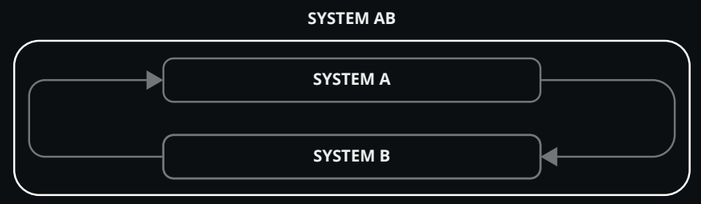
C



A system consists of a number of states, the amount of states in a system can be described as its variety (A), which forms a measure of a system's complexity (B). Systems are in every case embedded within an environment, which can be characterized in the same way: it has a number of states -- and often a very high complexity (C).

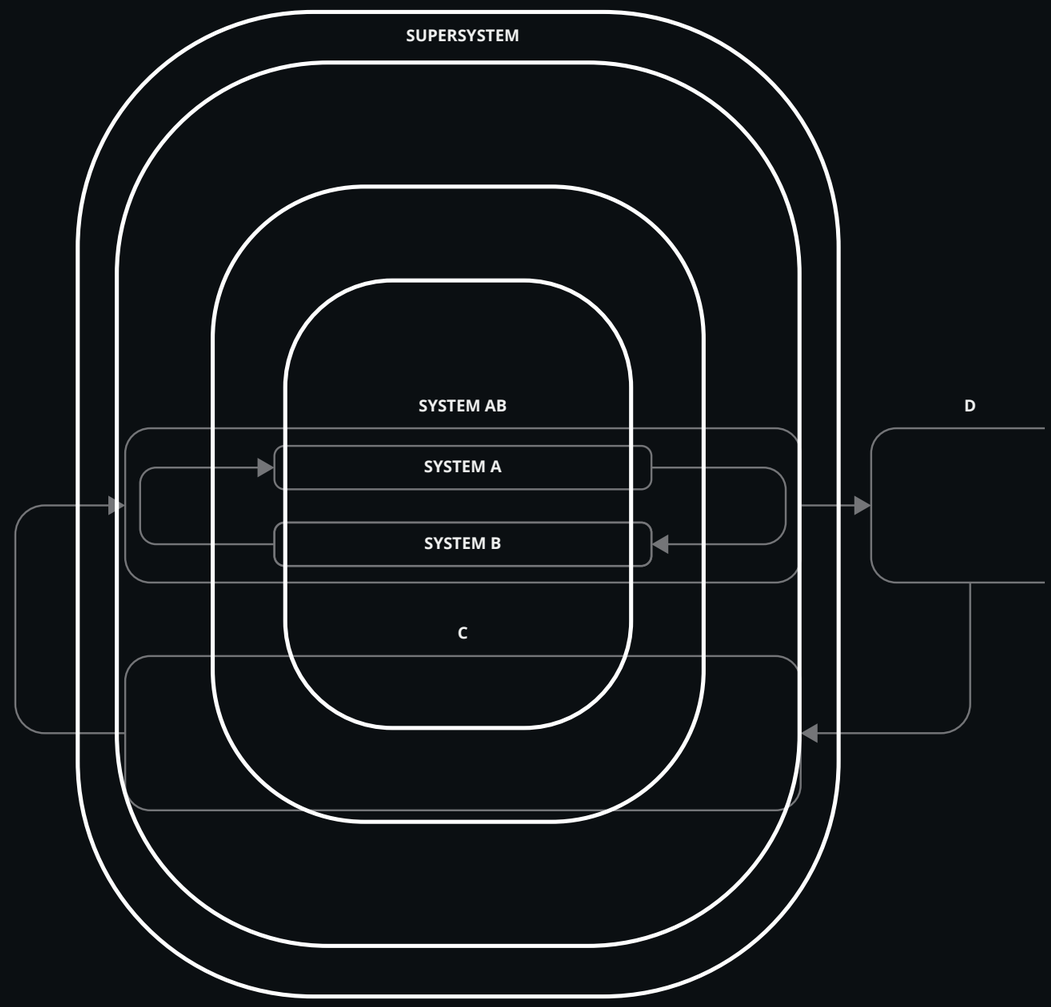
Systems can consist of subsystems - as is the case with environments. (D)

D

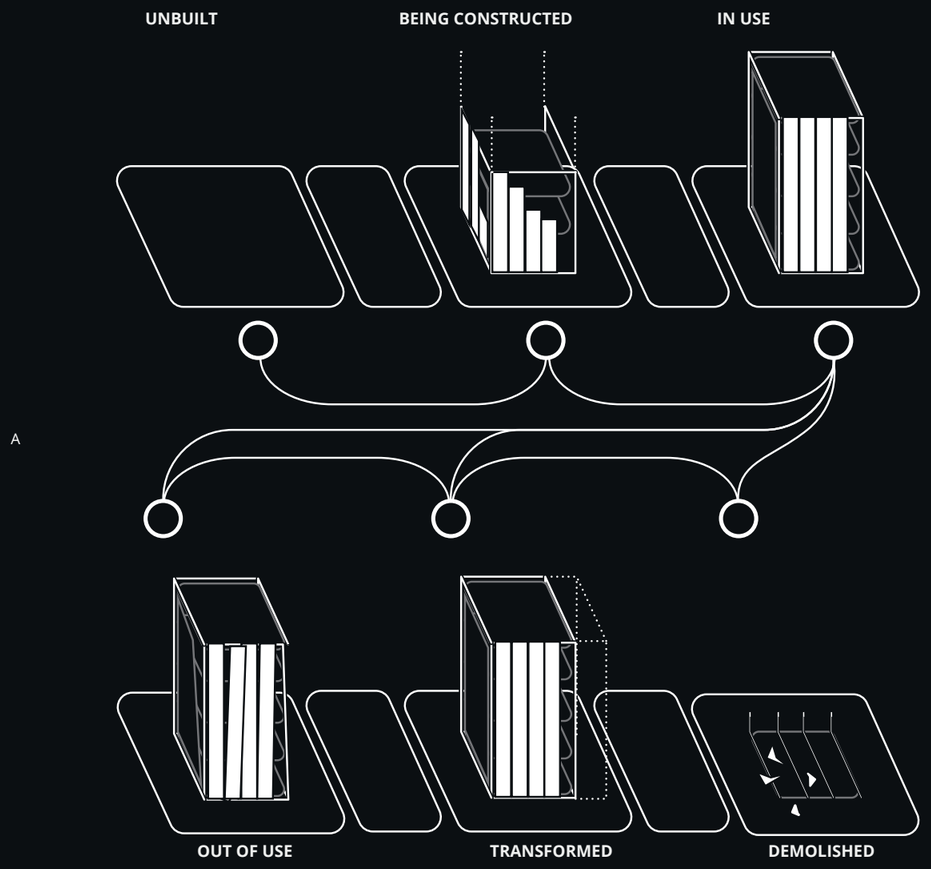


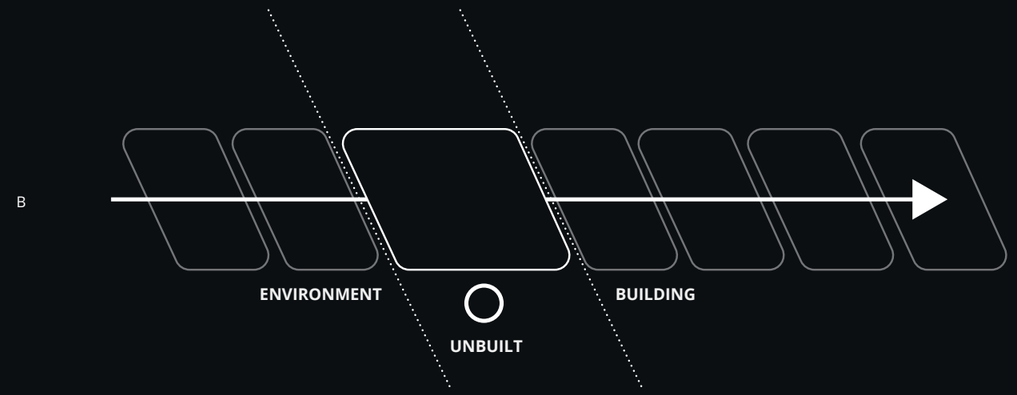
Recursivity

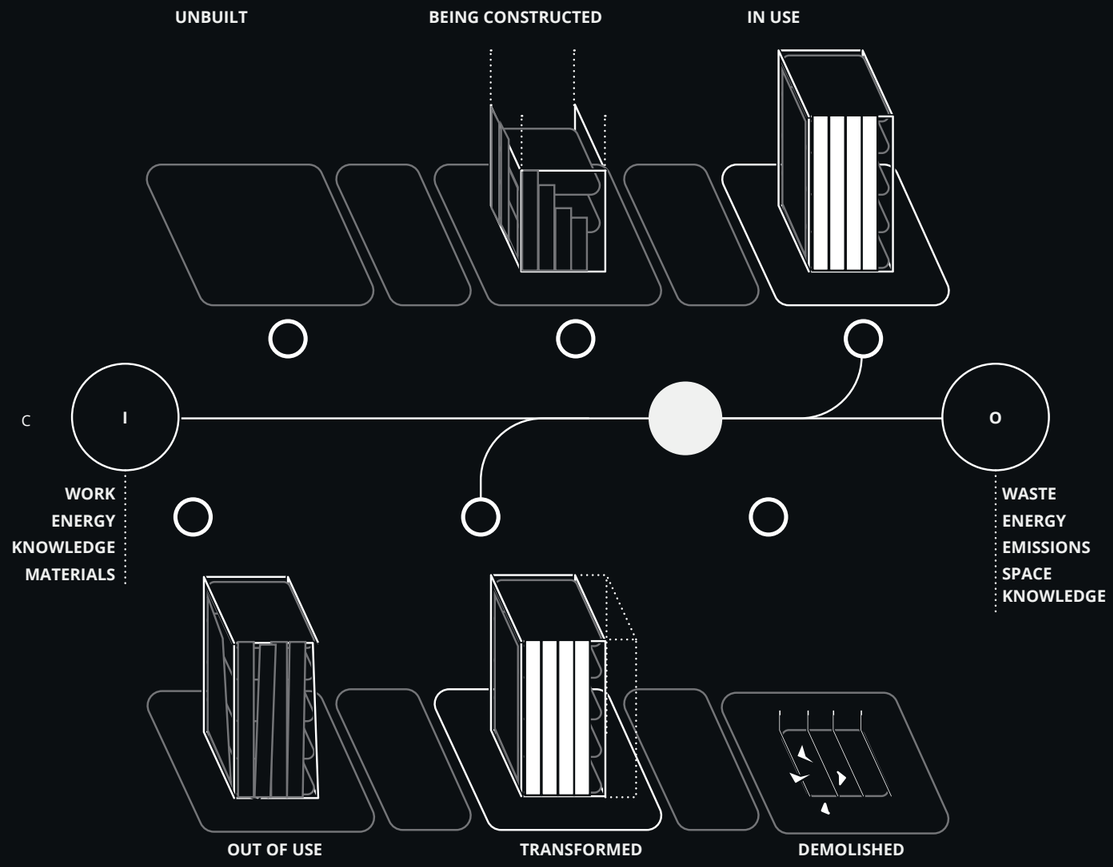
Systems made up of other systems can be characterized as being recursive: parts have the same properties as wholes -- each with their own interrelated states, and thus behaviours owing to those relations.



Building states





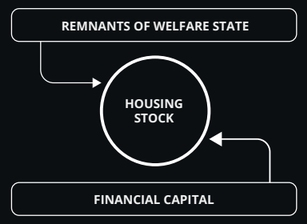


Similarly, buildings can take on states (A) -- those states are not as discrete as here shown, but instead, form continuous transitions between them (B). As with any system, there are inputs and certain behaviour that produce outputs (C).

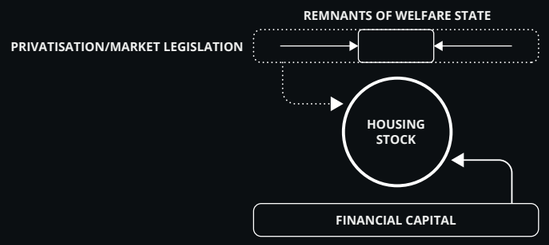
Design conditions: program, site, principles.

Design problem: housing in amsterdam.

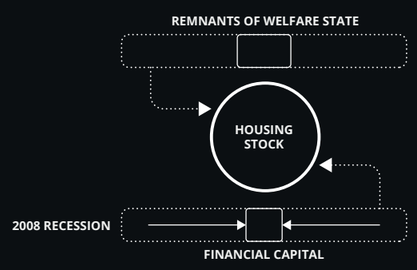
A



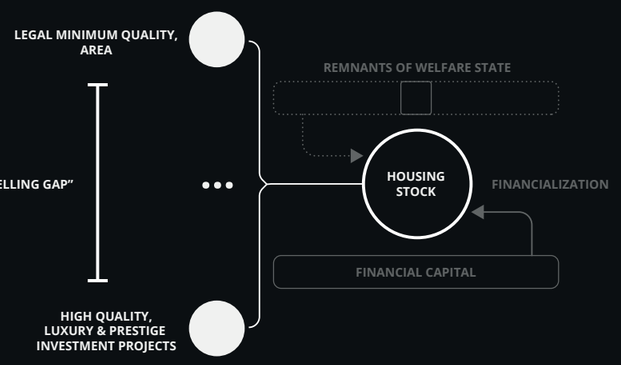
B



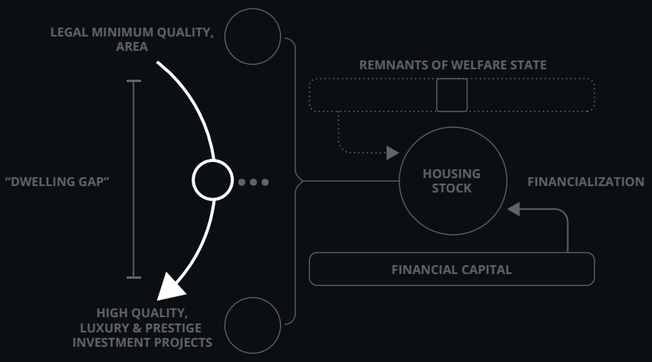
C



D

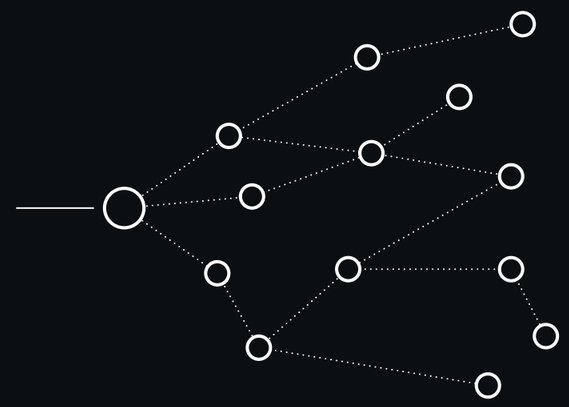


D



The city of Amsterdam has traditionally profited from a very strong welfare state. A significant portion of the city's housing stock originates with it, the other portion can be ascribed to private actors (A). Recent drives towards privatisation and legislation centered around enforcing market mechanisms in housing have decreased the effect of the remnants of the welfare state on housing construction (B). While financial capital recovered quickly after the 2008 recession (C), its effects when seen together with the reduction in influence for public actors have caused a so-called "dwelling gap" to arise (D). Moreover, what remains of public housing is quickly being sold off into private hands (D). This dynamic provides the backdrop for this project: an attempt at utilizing architectural strategies to circumvent the rapid sell off of public housing by introducing indeterminacy (E).

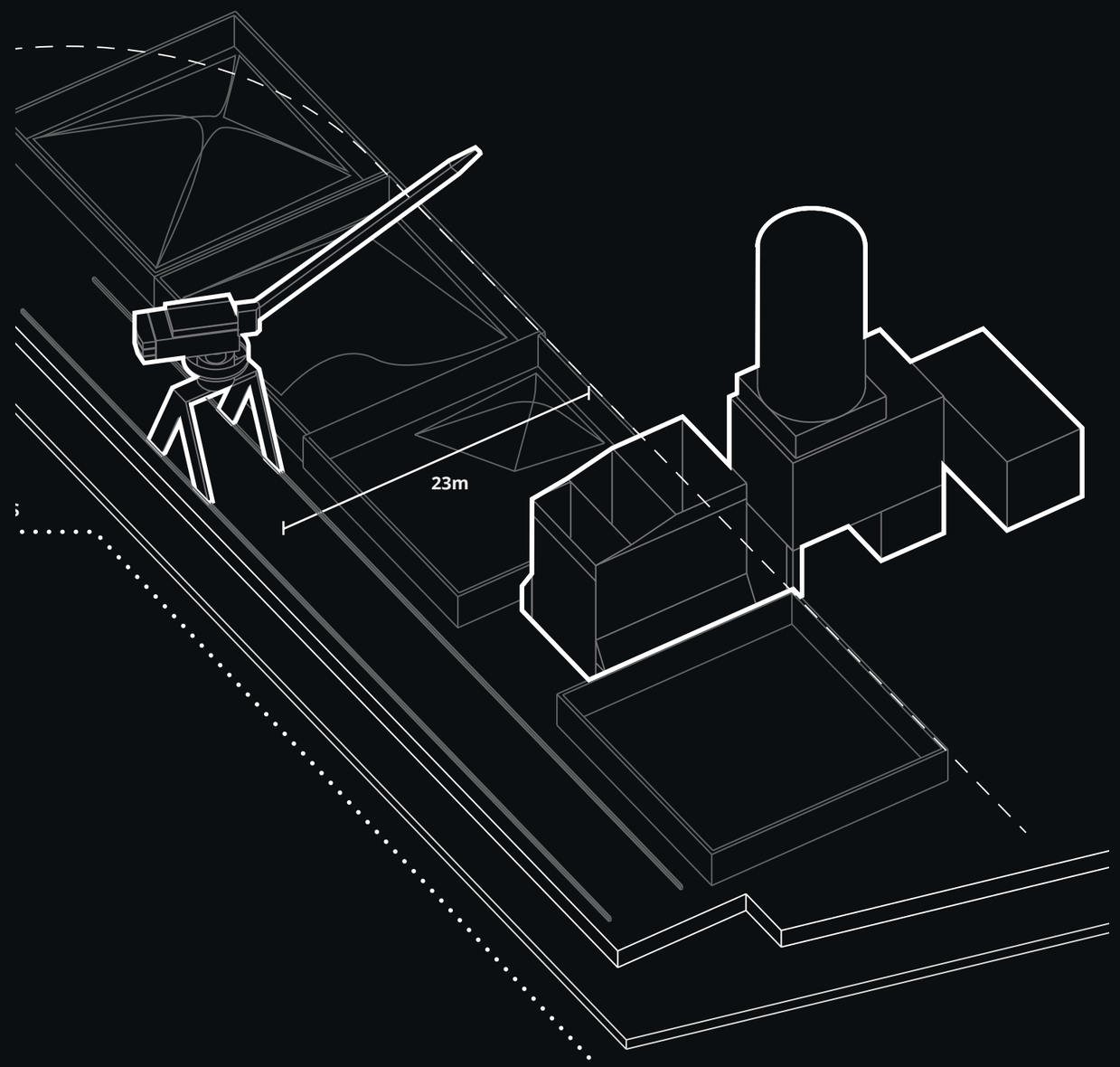
E



The current Mebin concrete plant, on the northwestern tip of the city of Amsterdam, provides a convenient location for exploring dynamic architectures due to a number of factors. Due to encroaching residential development, the area that the plant occupies might soon come to lie in a zone that does not allow for industrial levels of (noise) pollution.

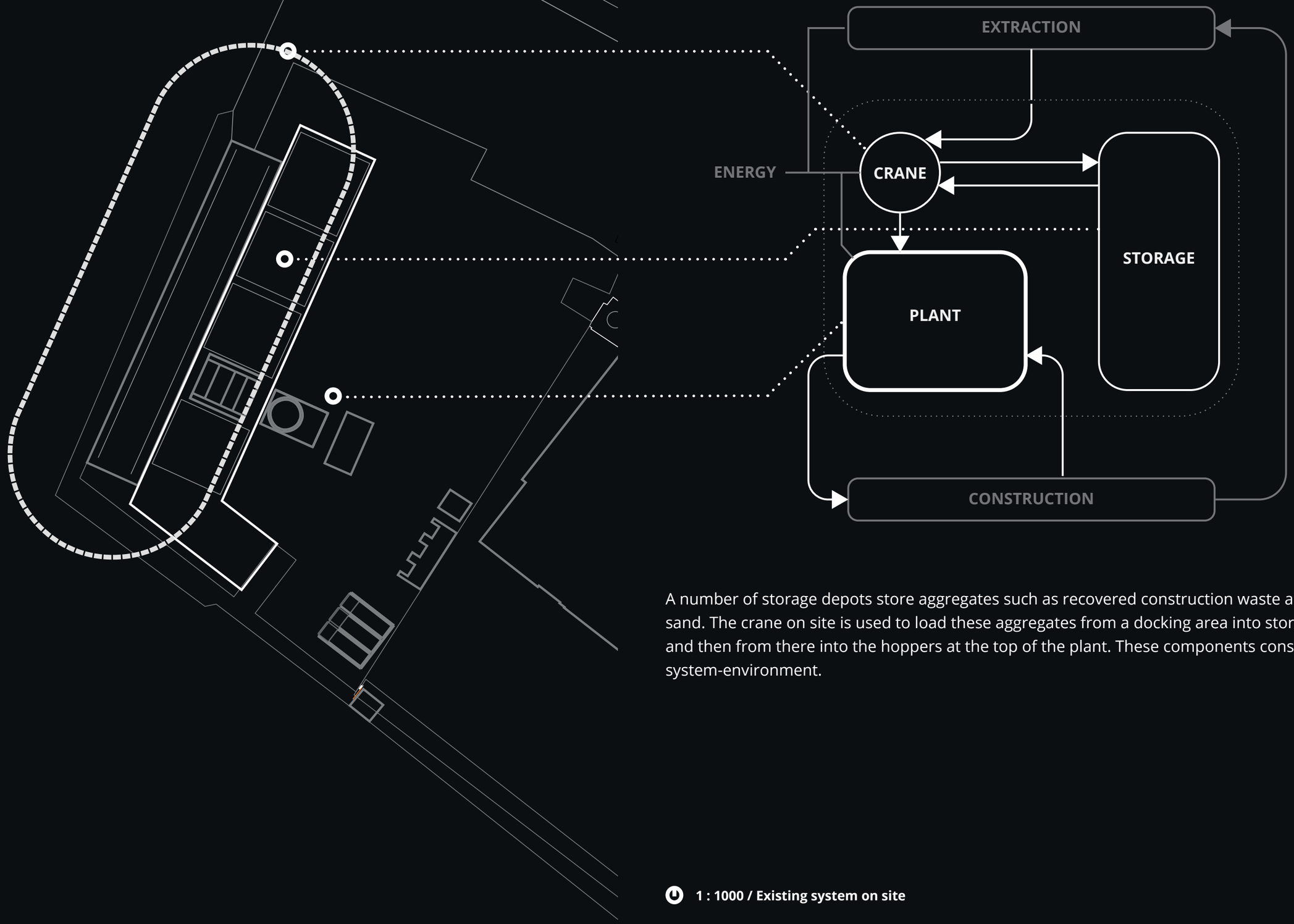
1:25000

Site



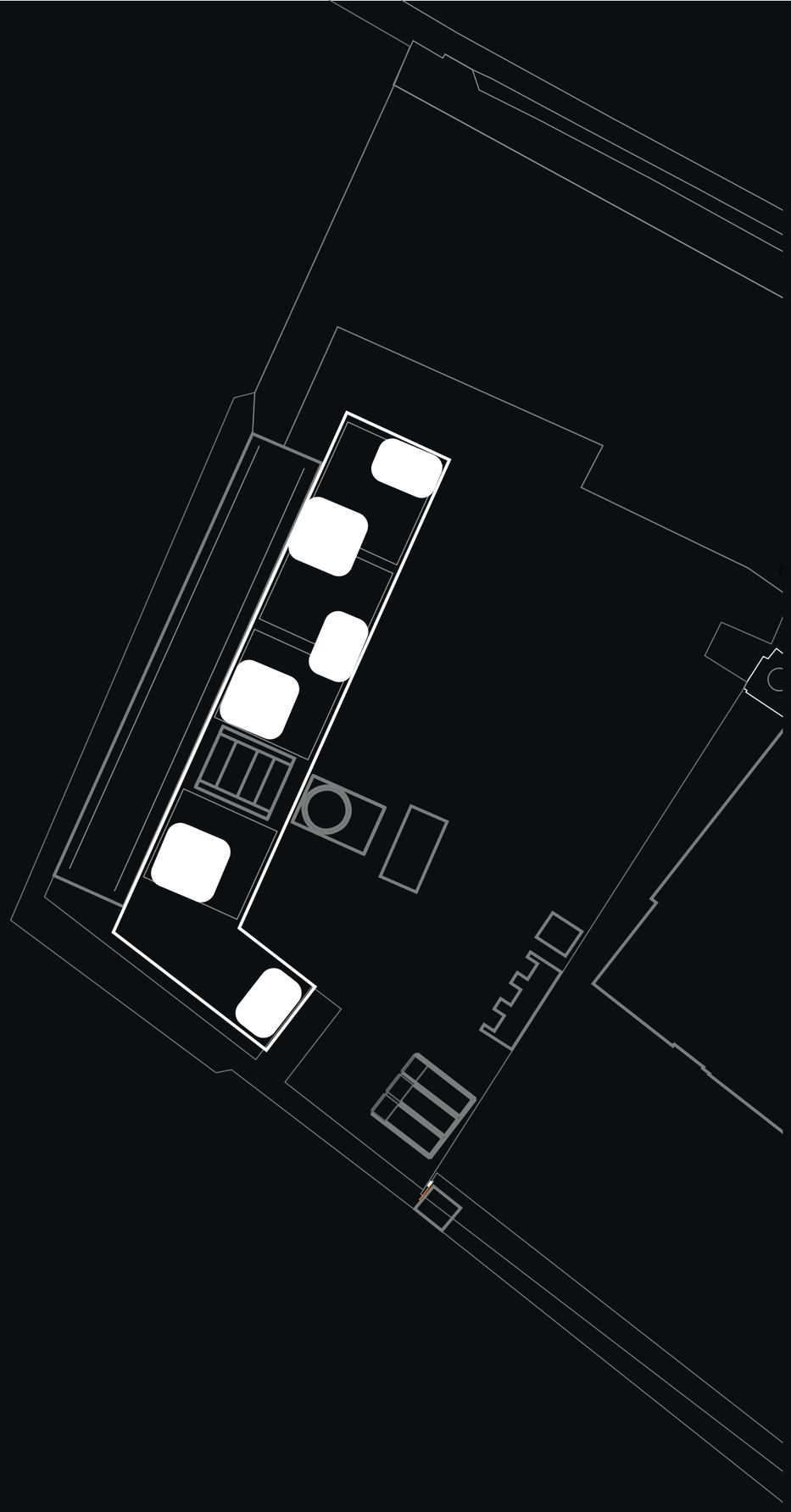
The site consists of a number of components that together make up the machine that is the conrete plant.

System-environment

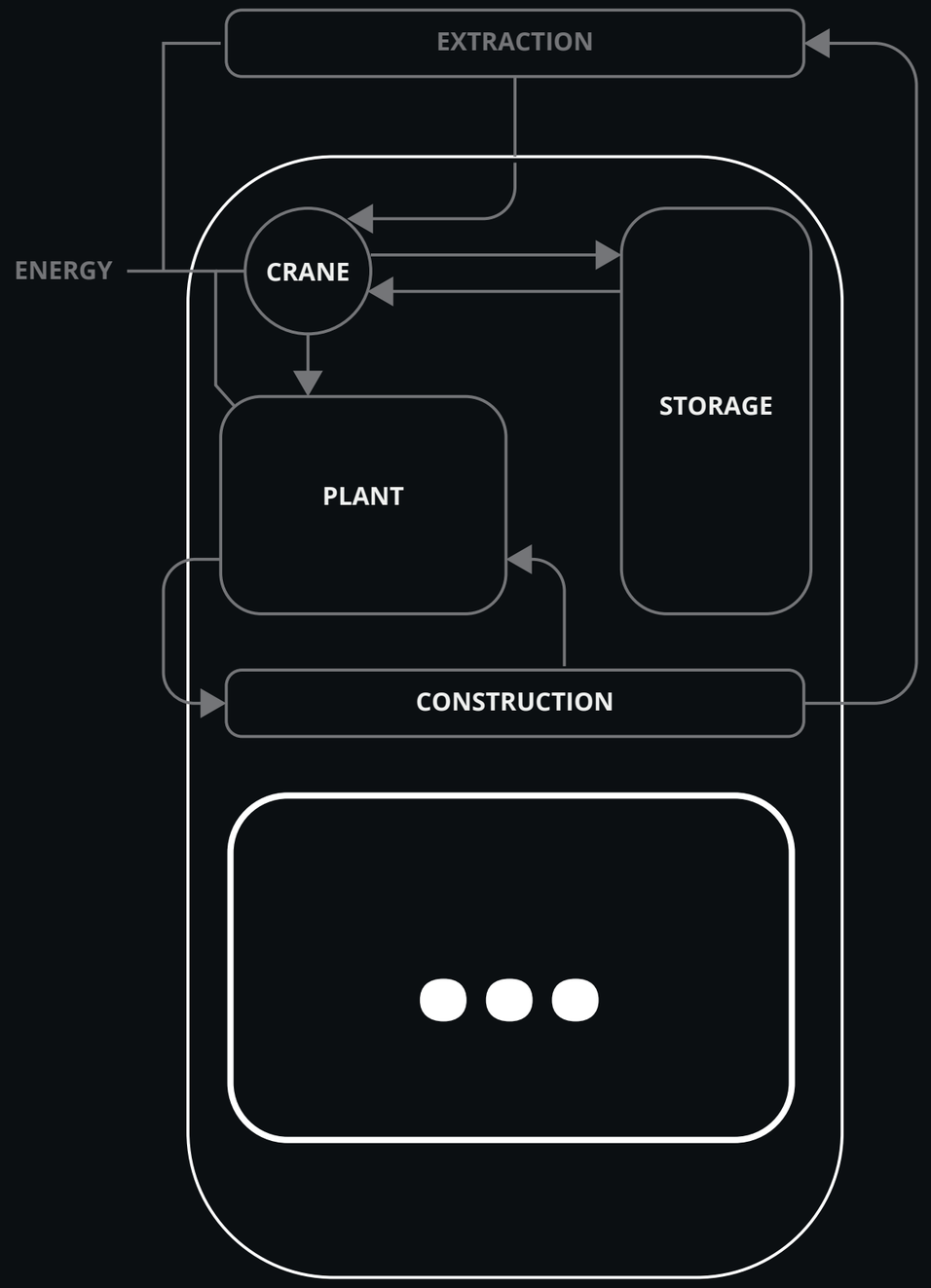


A number of storage depots store aggregates such as recovered construction waste and sand. The crane on site is used to load these aggregates from a docking area into storage, and then from there into the hoppers at the top of the plant. These components constitute a system-environment.

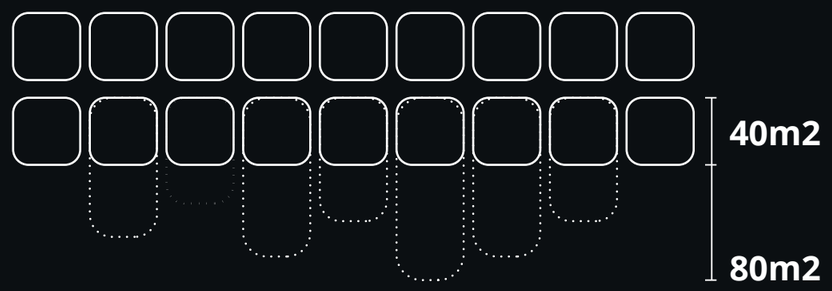
1 : 1000 / Existing system on site

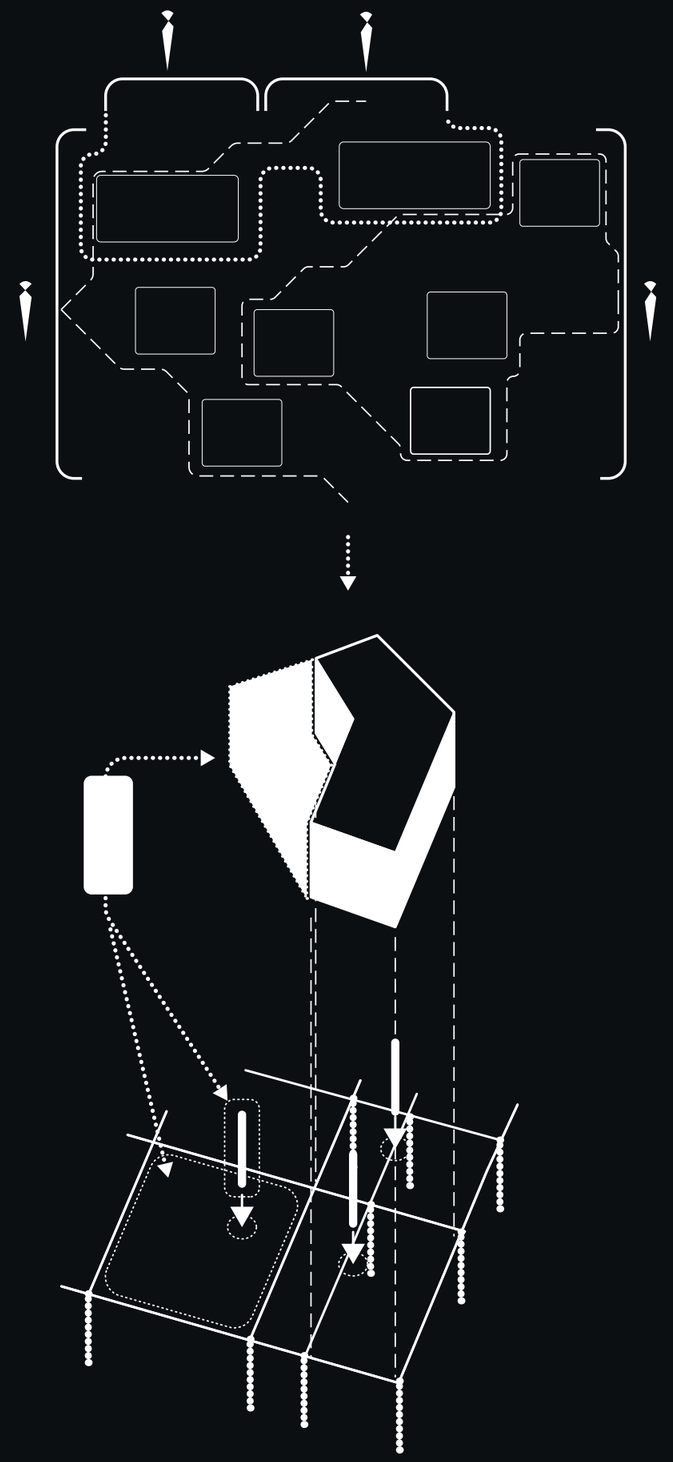


Plug-in approach



1 : 1000 / Early Sketch of Approach

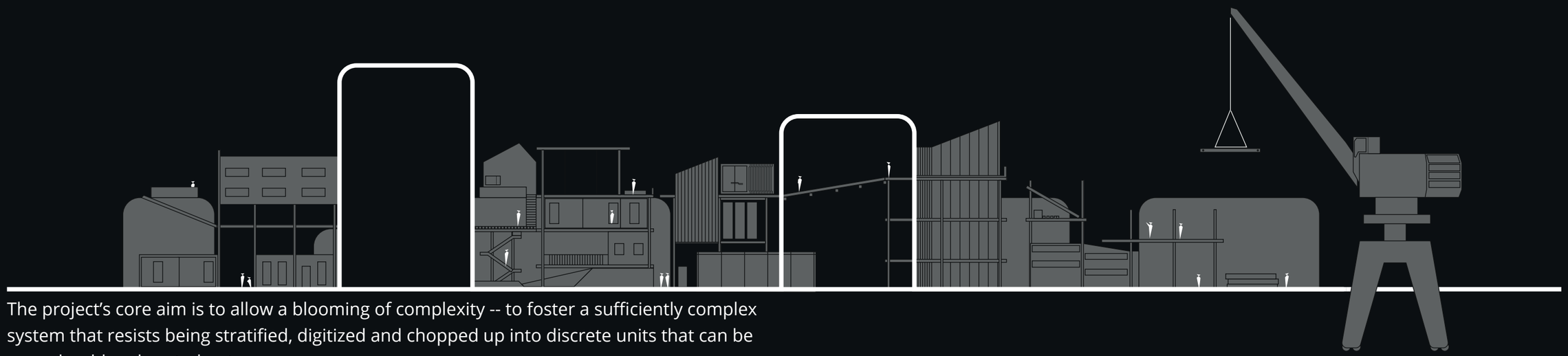




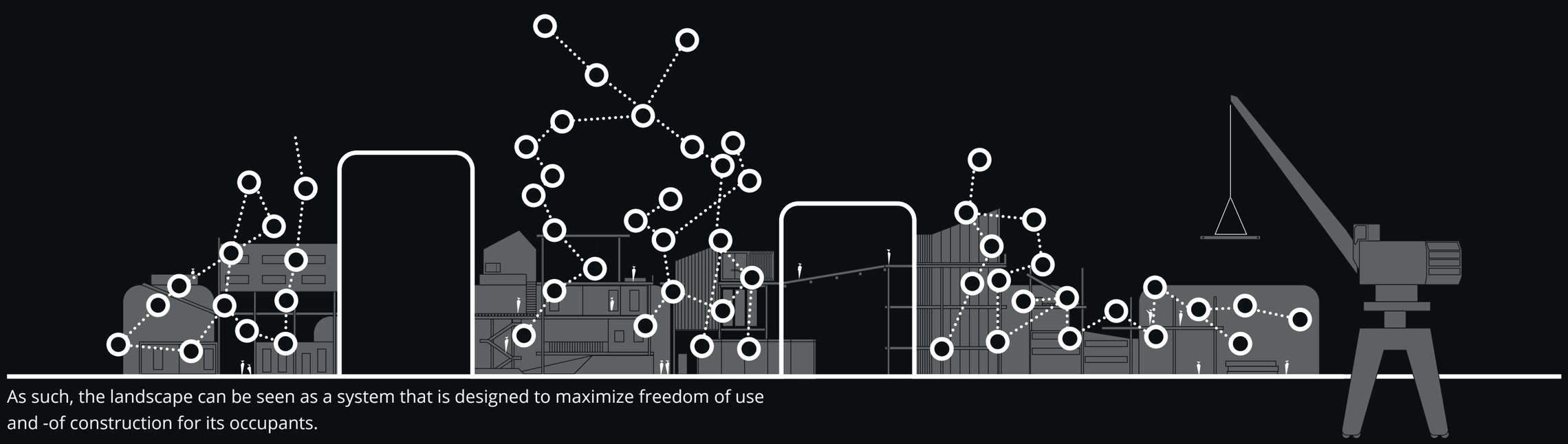
The project's program can be formulated as follows:

A landscape that can provide a material substrate for 60 occupants, with 40-80m2 of space for each of these. It provides flows of energy, construction materials, and a suitable support structure to enable convenient construction and reconstruction for its occupants.

Material Landscape

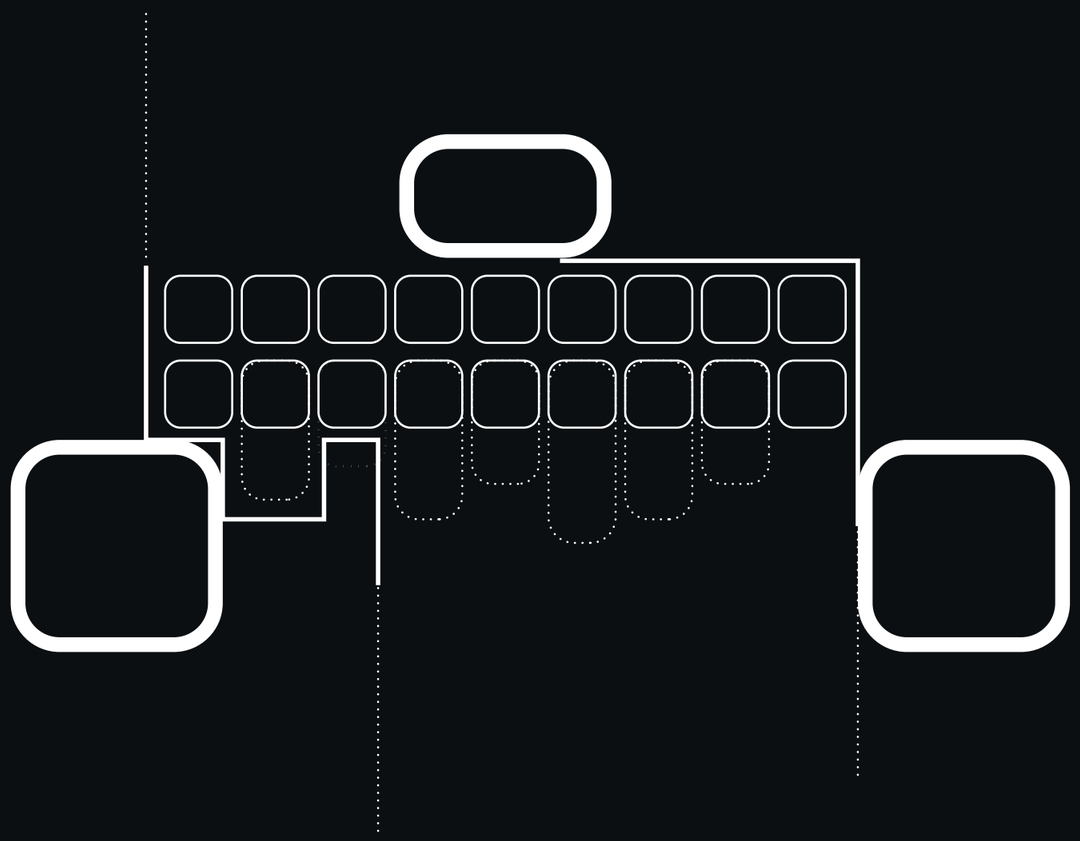


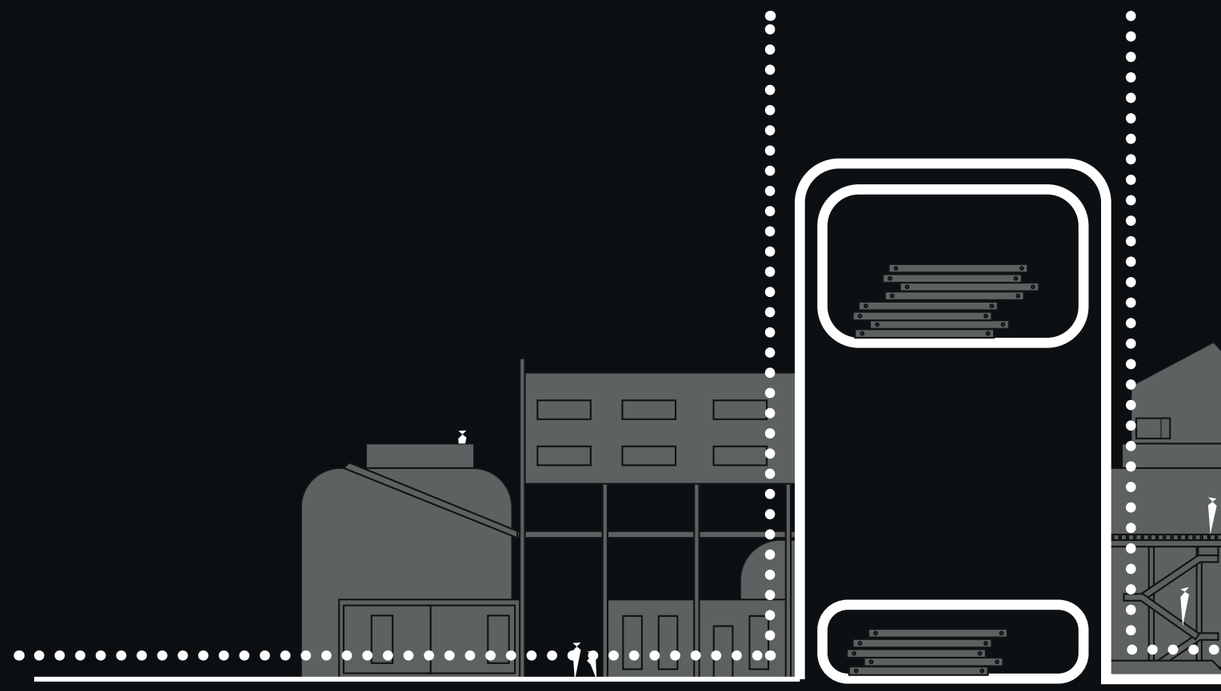
The project's core aim is to allow a blooming of complexity -- to foster a sufficiently complex system that resists being stratified, digitized and chopped up into discrete units that can be owned, sold and rented out.



As such, the landscape can be seen as a system that is designed to maximize freedom of use and -of construction for its occupants.

Infill / cores

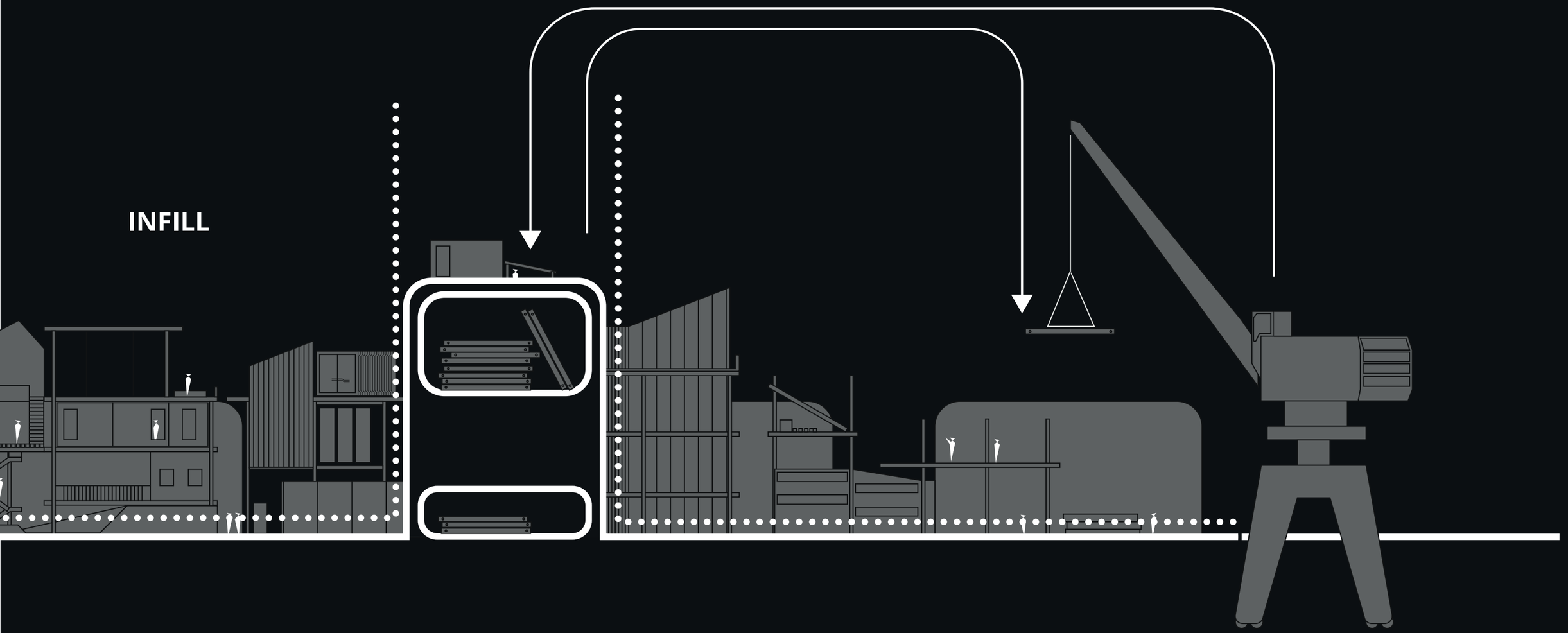




Storage of materials

The landscape consists of a set of cores that function as enabling constraints for occupants while simultaneously providing as many necessities as possible for architectural freedom in the infill inbetween.

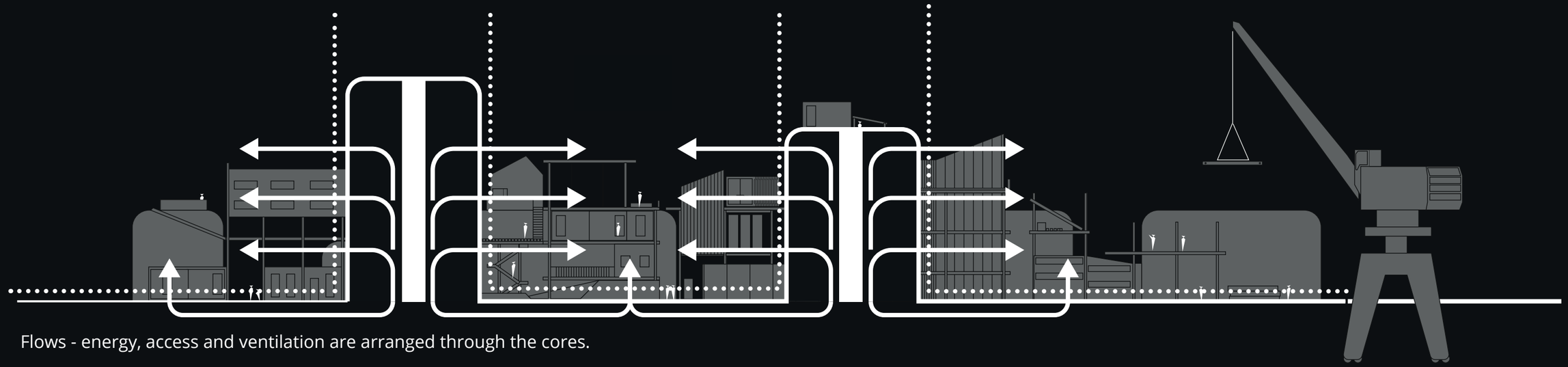
INFILL



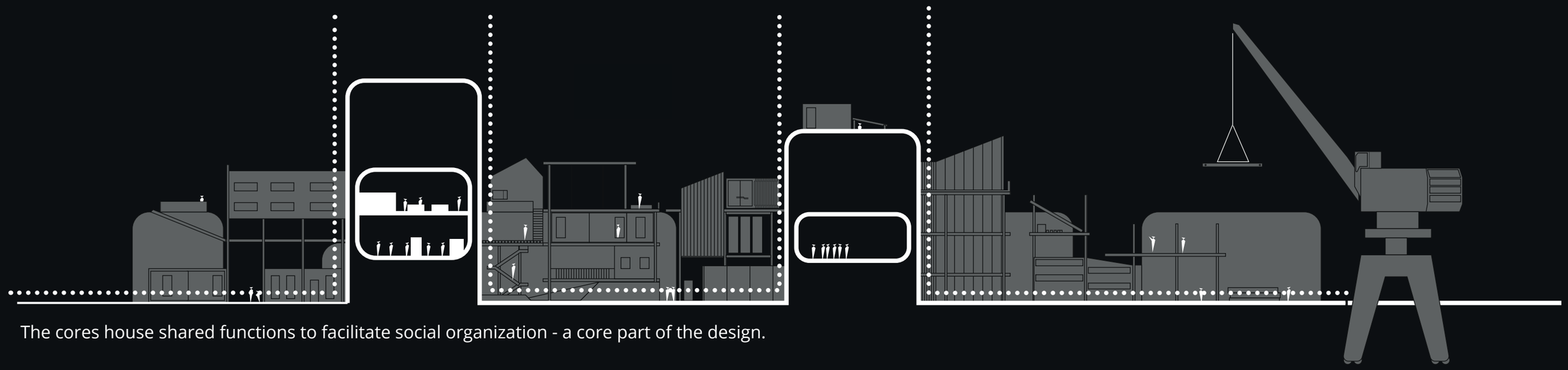
LANDSCAPE

CRANE

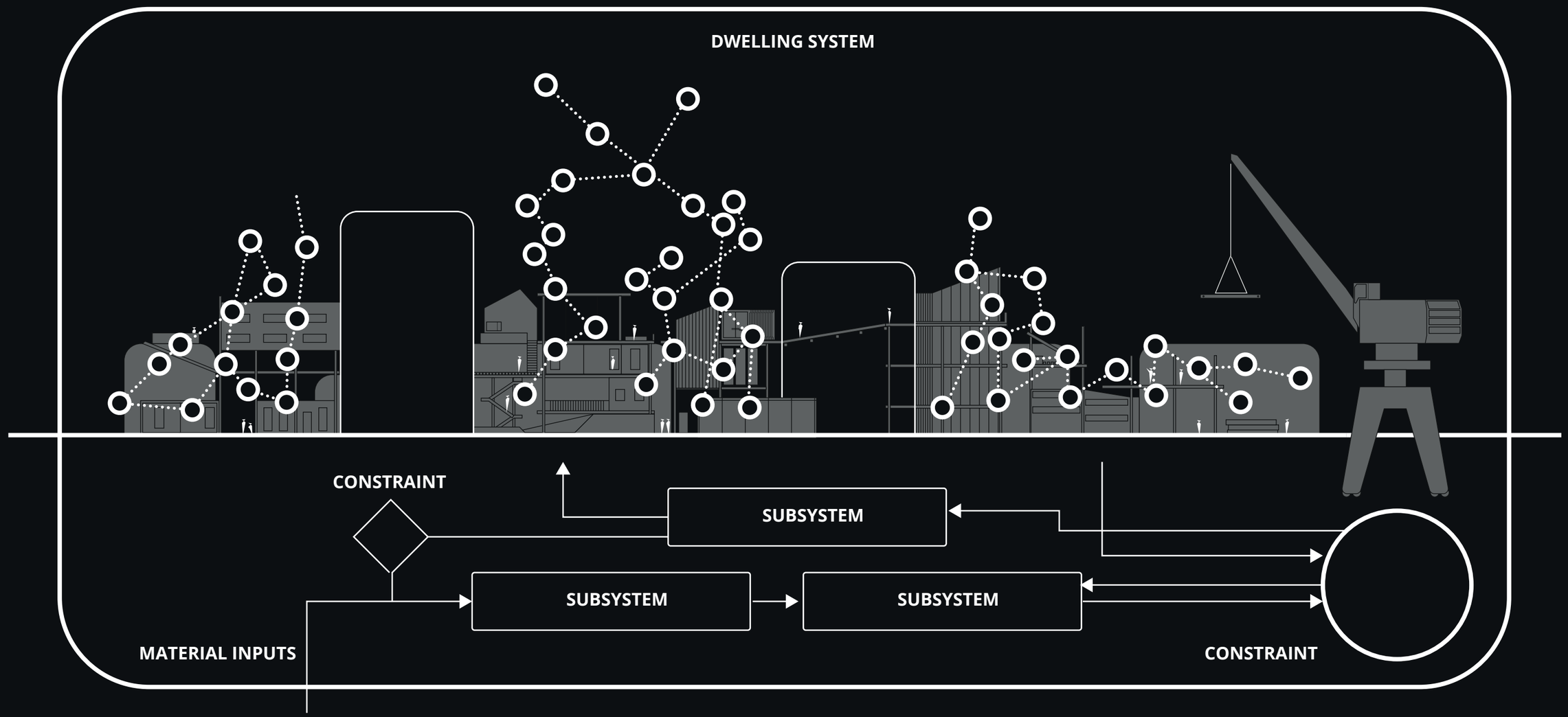
Infill / cores



Flows - energy, access and ventilation are arranged through the cores.

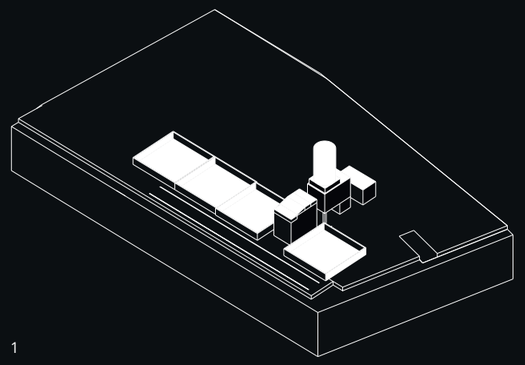


The cores house shared functions to facilitate social organization - a core part of the design.

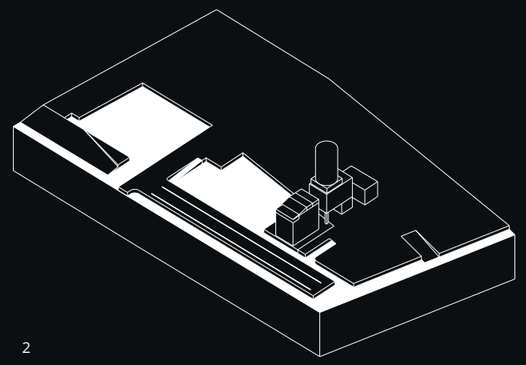

DWELLING SYSTEM
CONSTRAINT
SUBSYSTEM
SUBSYSTEM
SUBSYSTEM
MATERIAL INPUTS
CONSTRAINT

An open system: designing constraints.

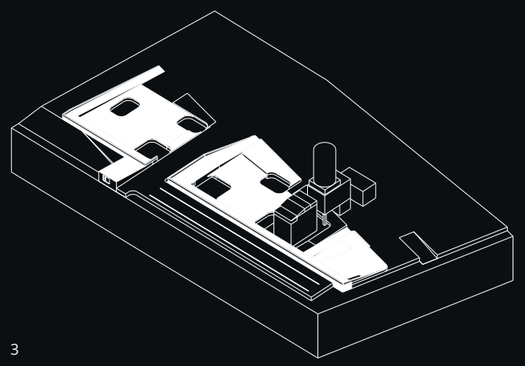
Construction process



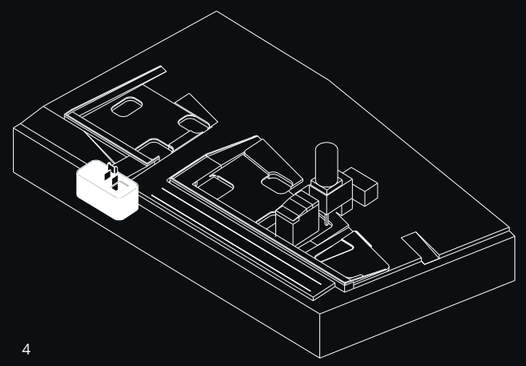
1



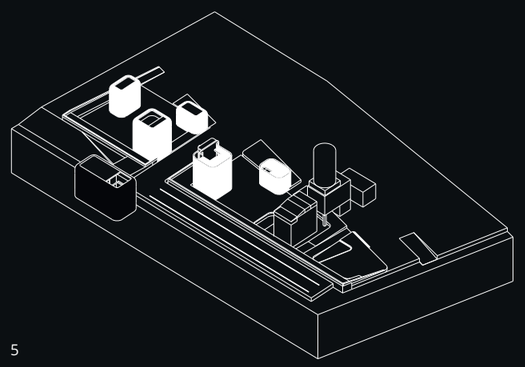
2



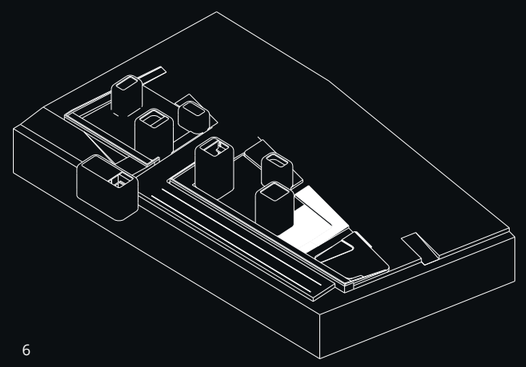
3



4



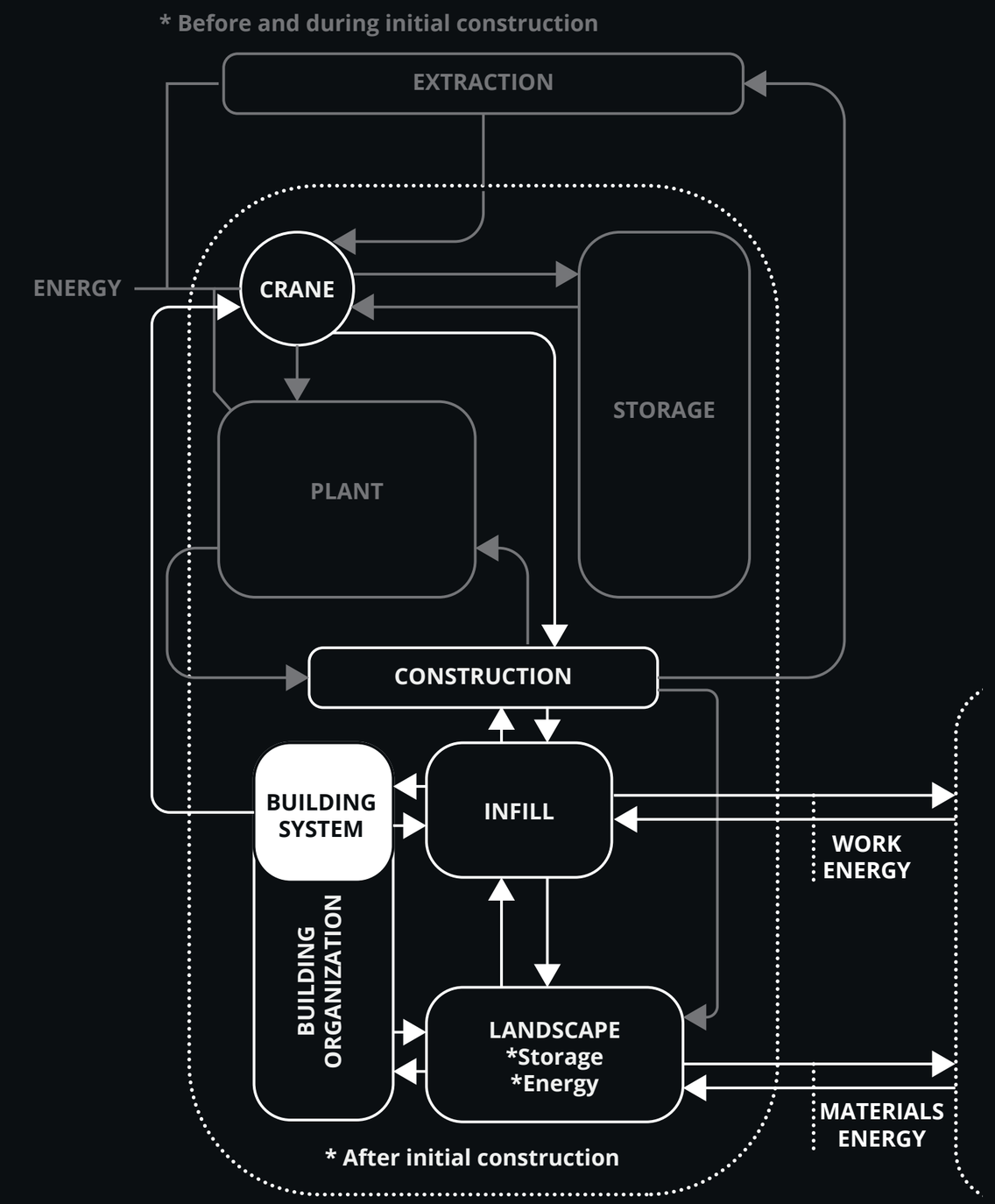
5



6

The construction process (1-6) of the landscape is an integral part of the approach to this site: the building functions as a plug-in to the existing system-environment of the site, that gradually spreads over the site and eventually takes the place of the concrete mill.

Plug-in approach



Organization

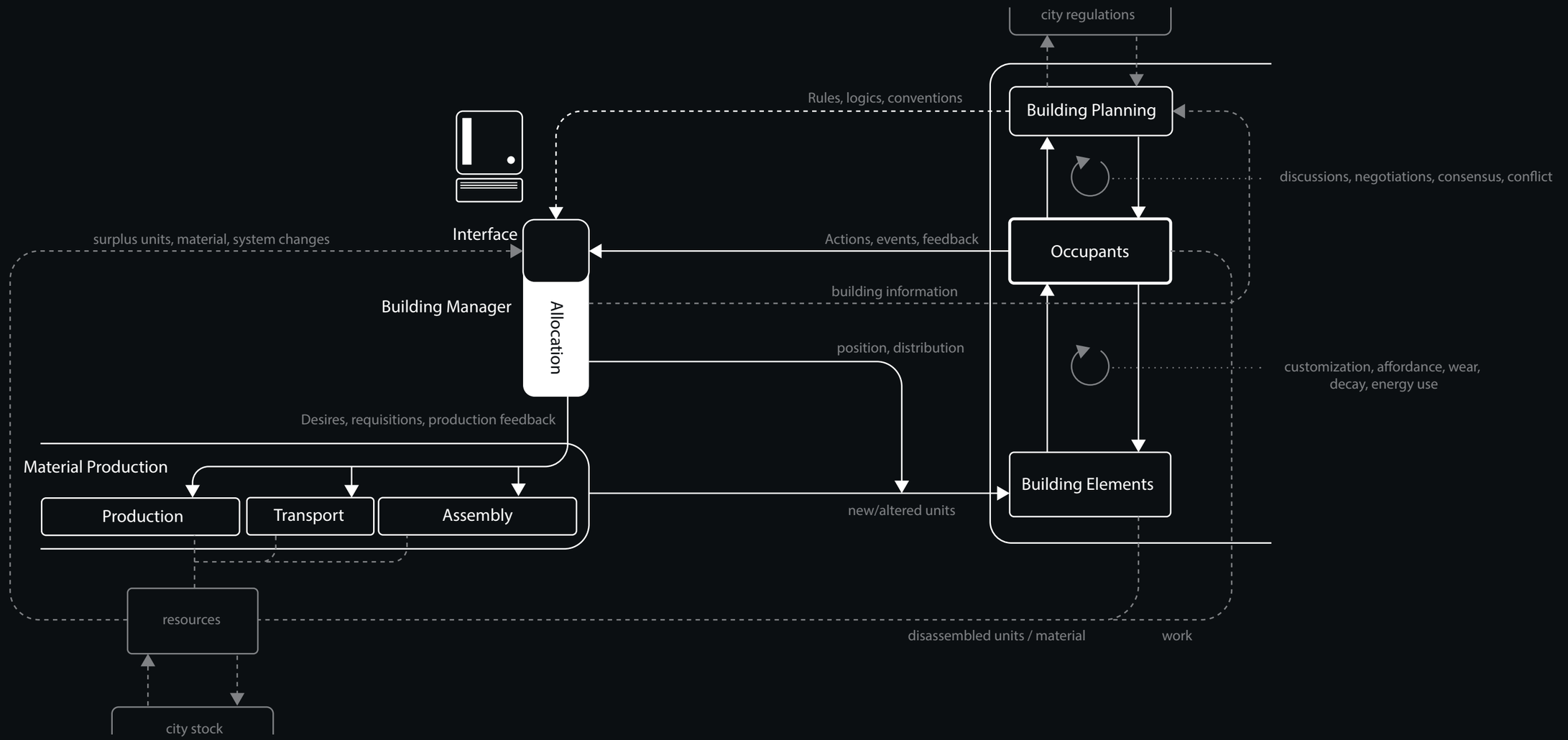
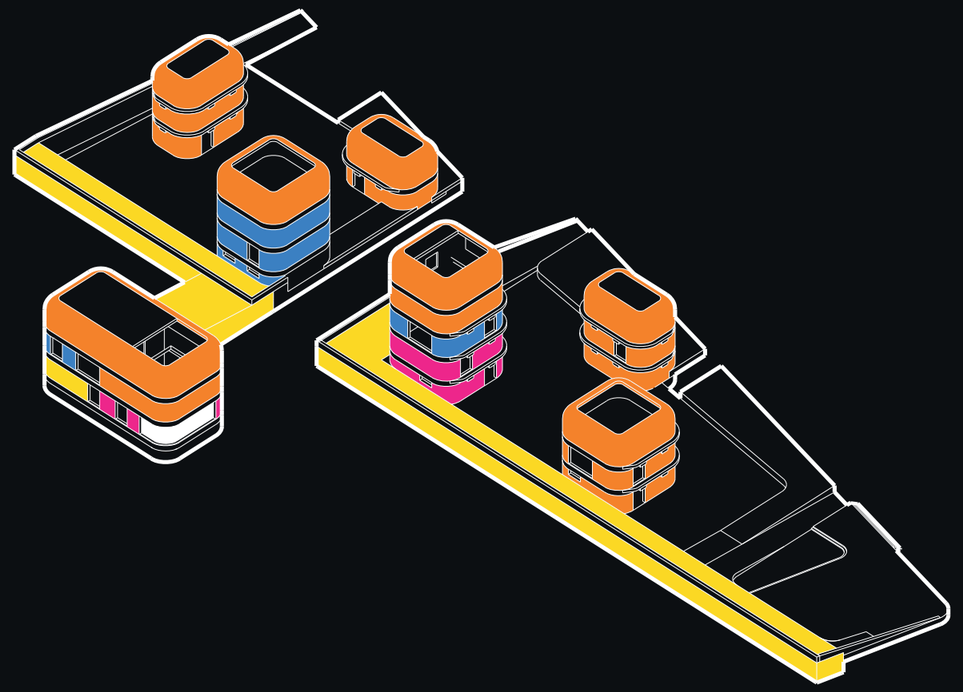


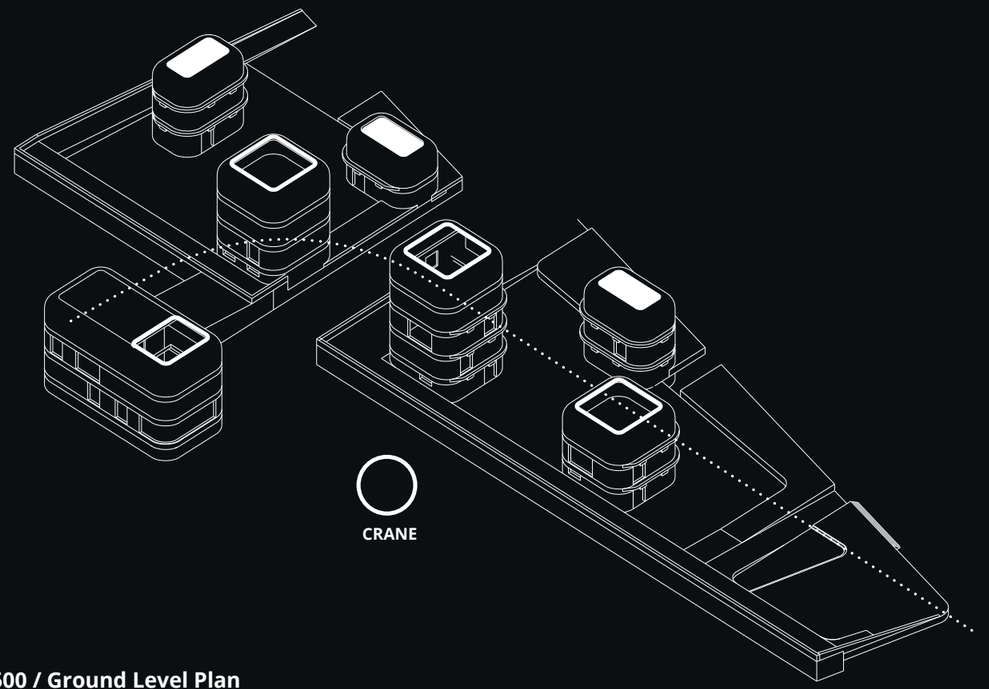
Diagram of the organizational structure for the cooperative on site.

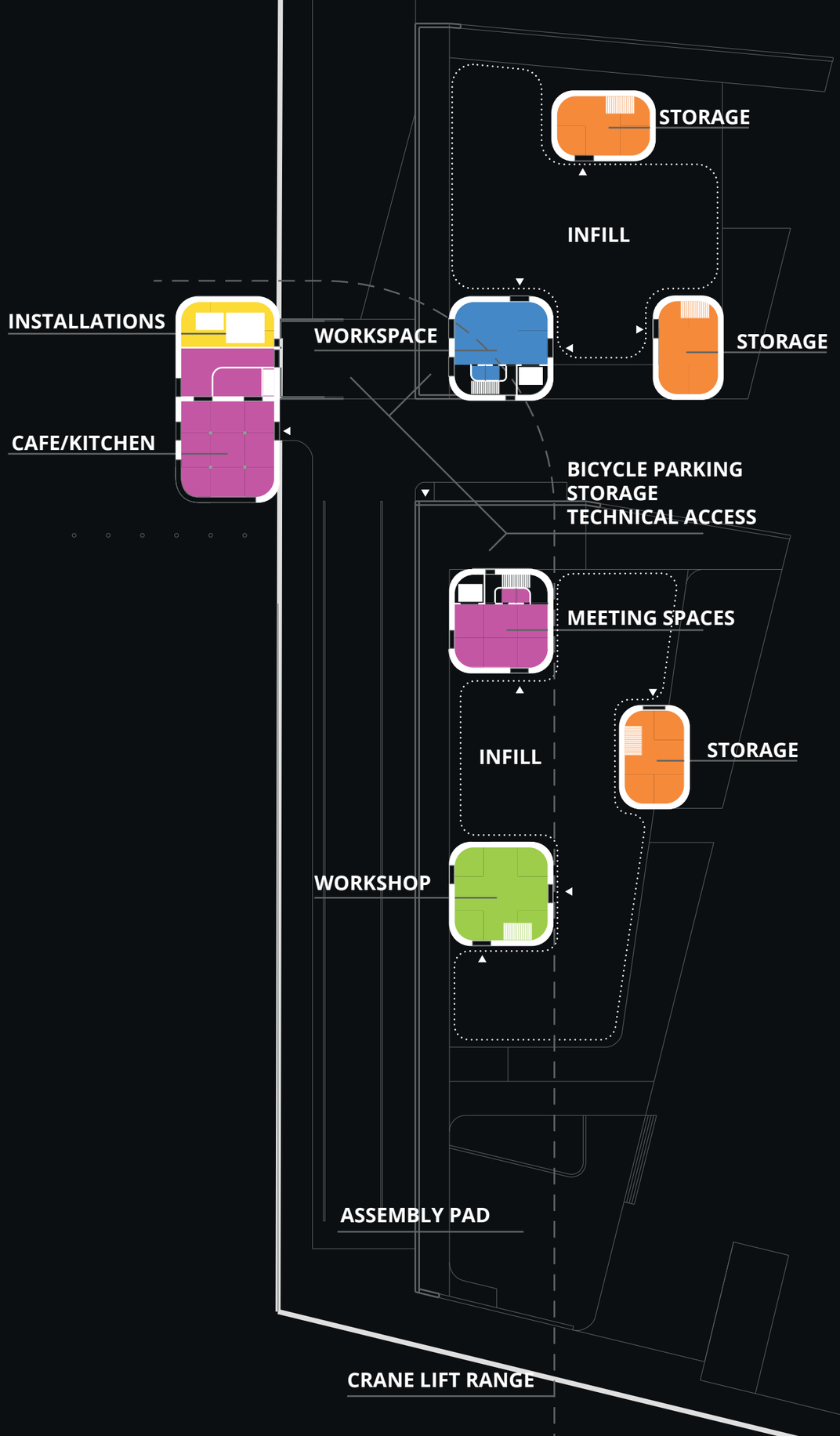
Use

Functional layers.



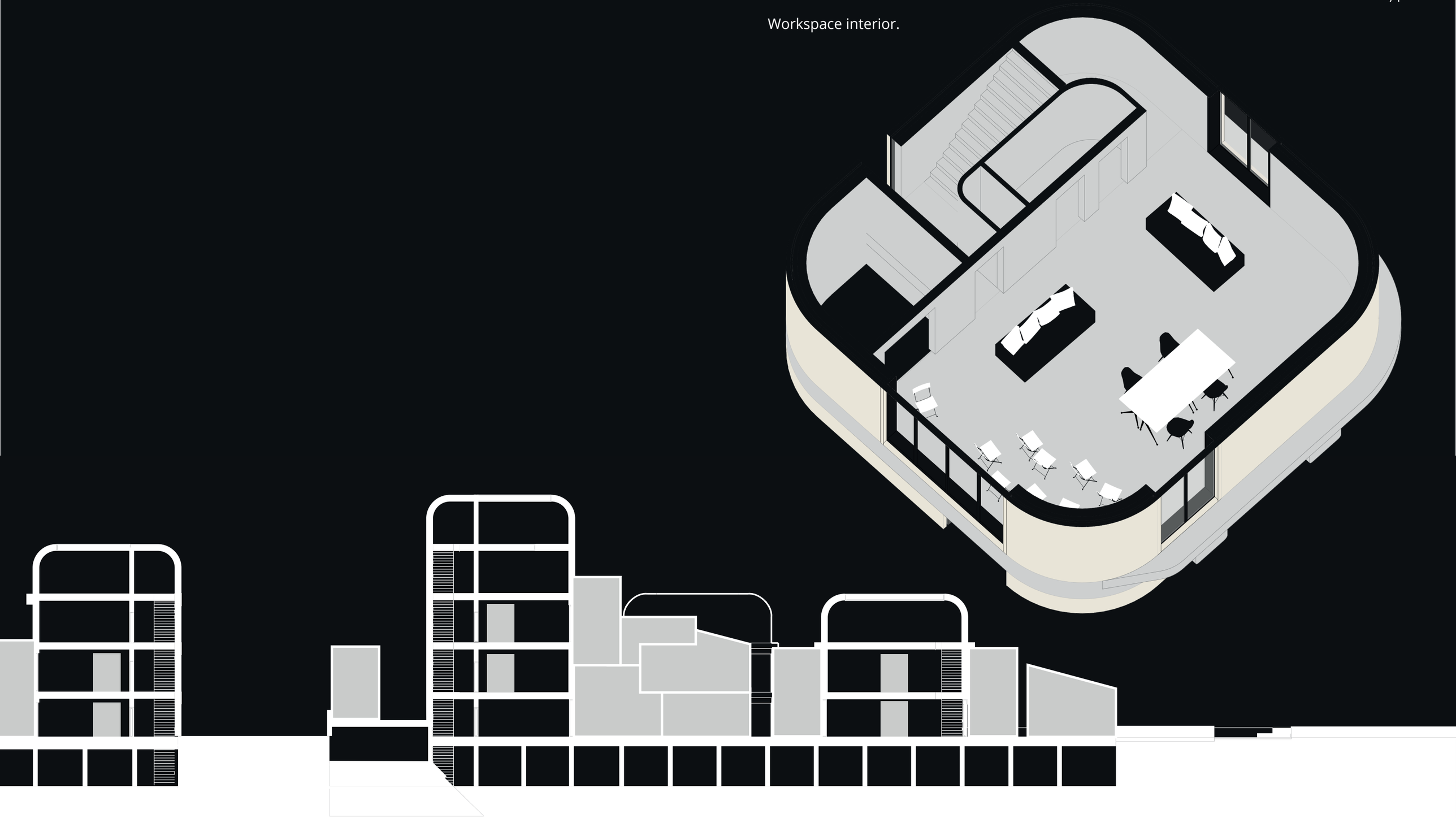
Crane access opening diagram.





1 : 500 / Ground Level Plan

Workspace interior.



crane access corridor

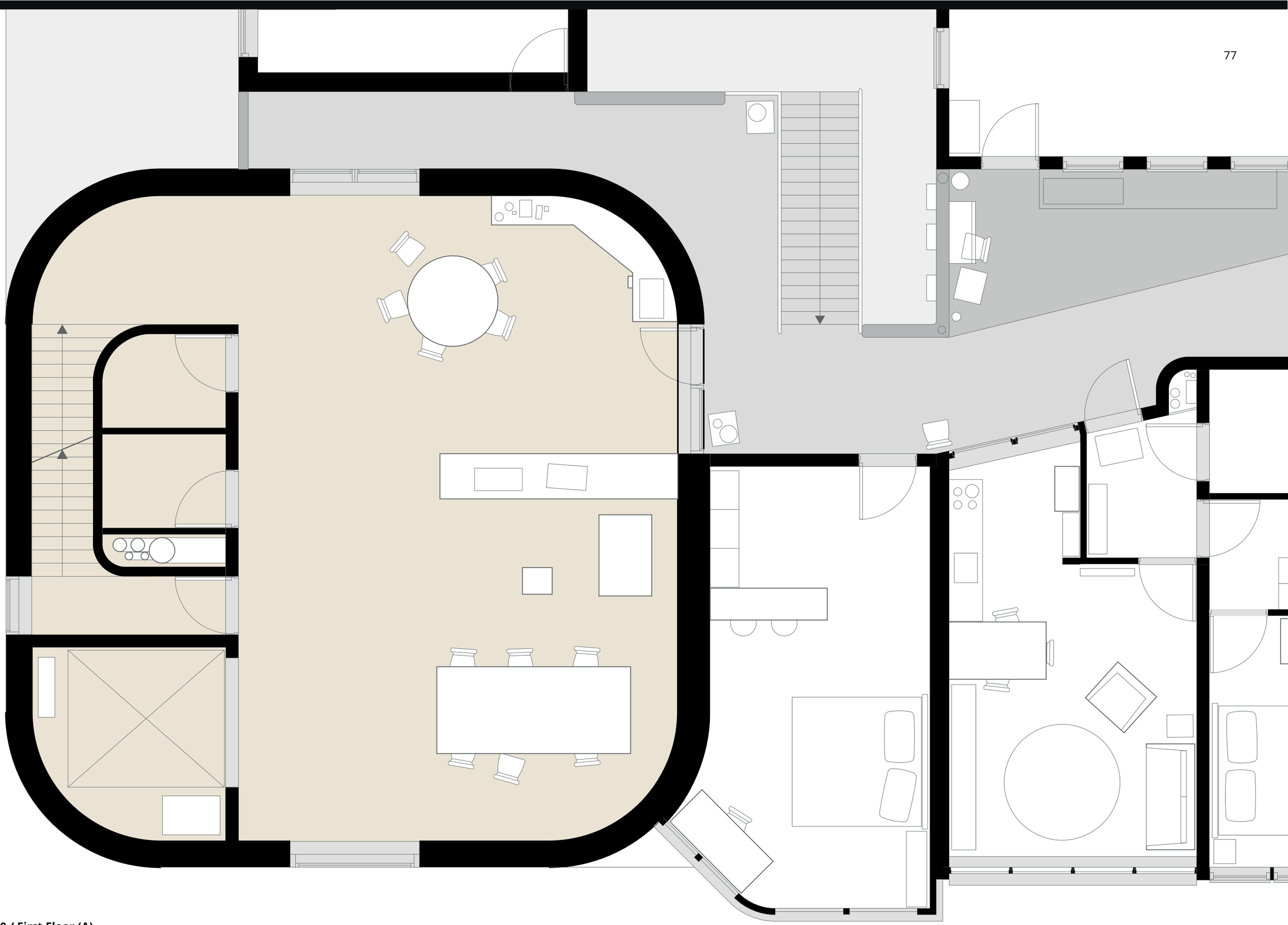
service area access

bike storage

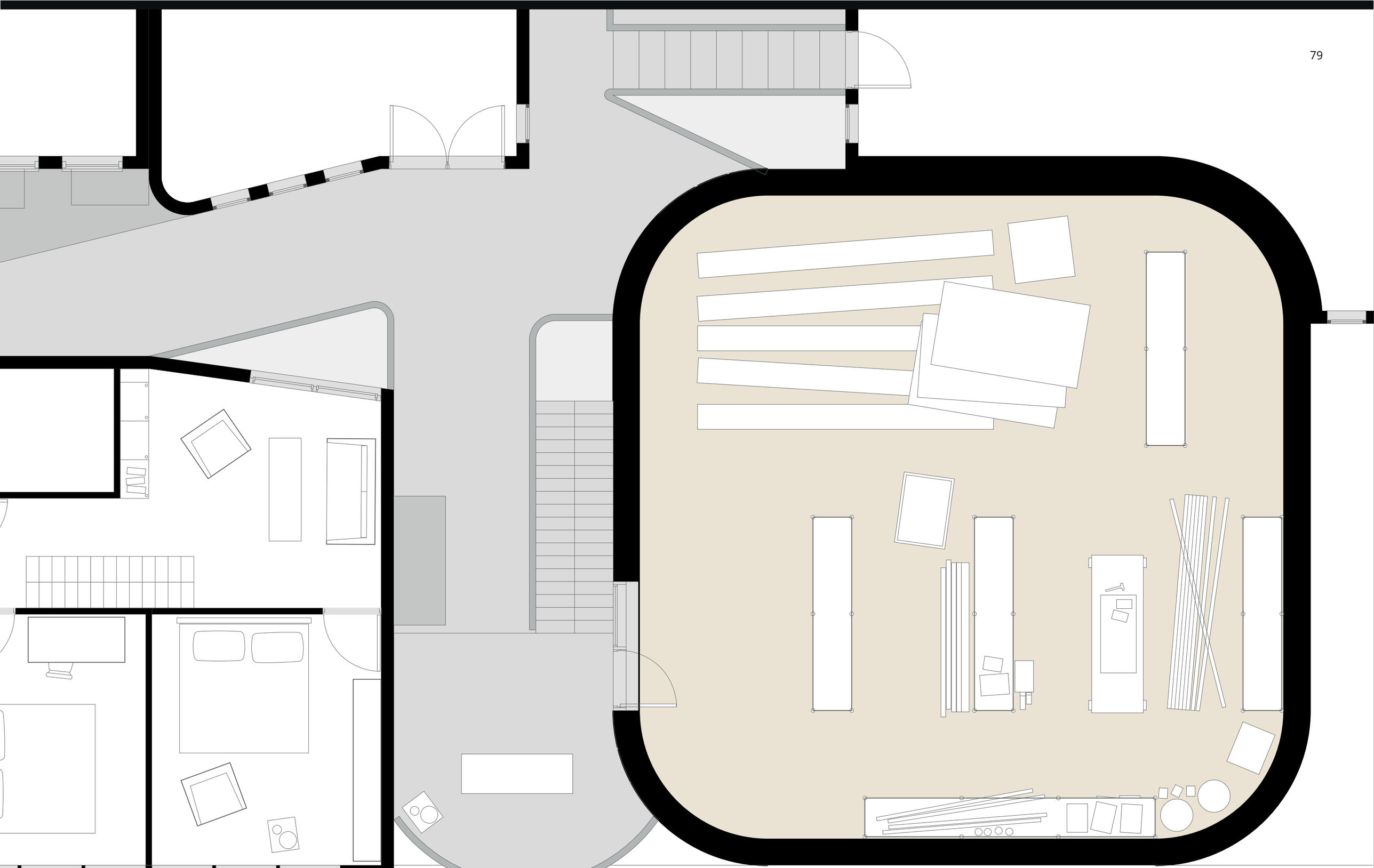
service area

assembly pad

1 : 100 / Section diagram



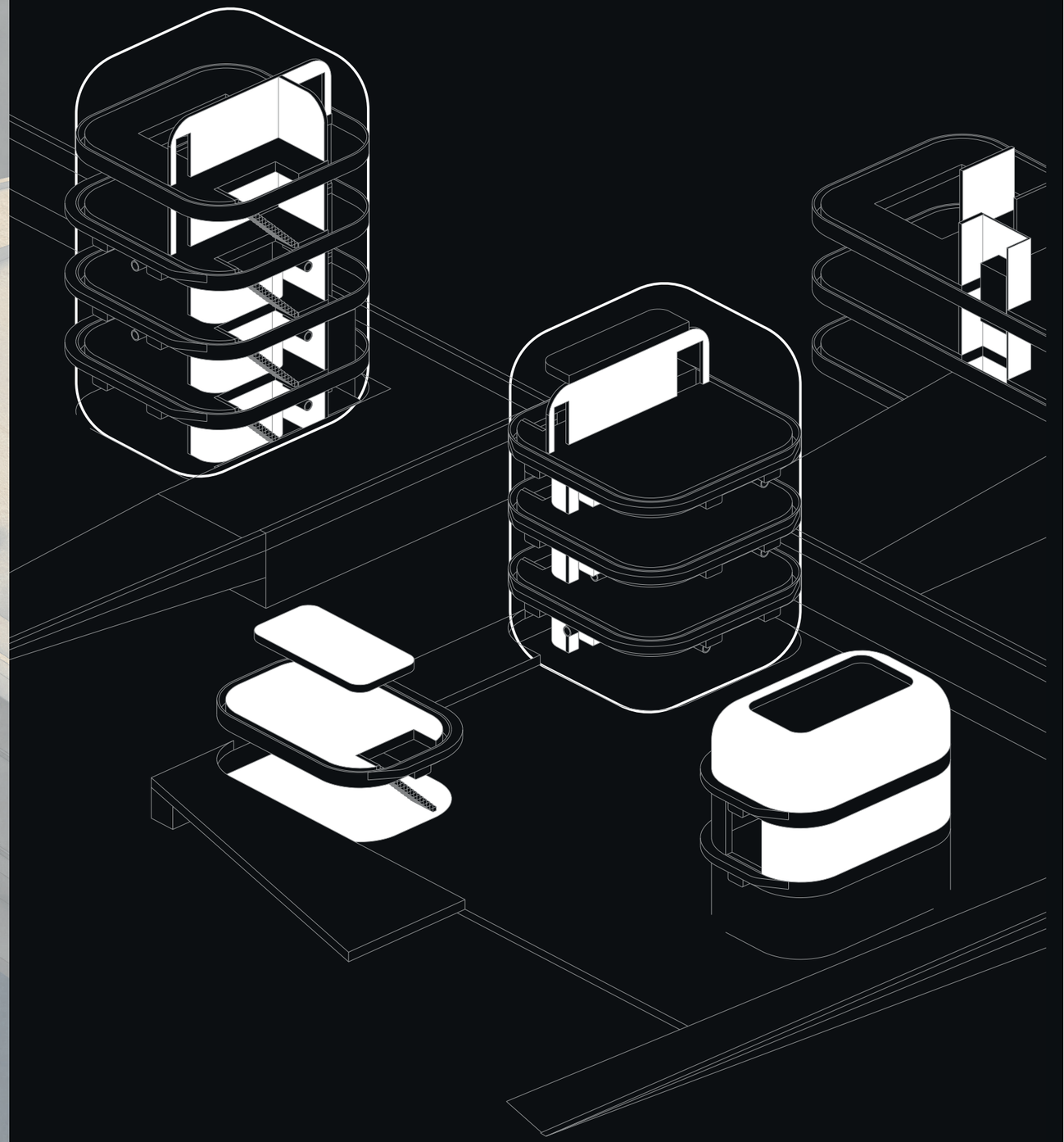
1 : 50 / First Floor (A)



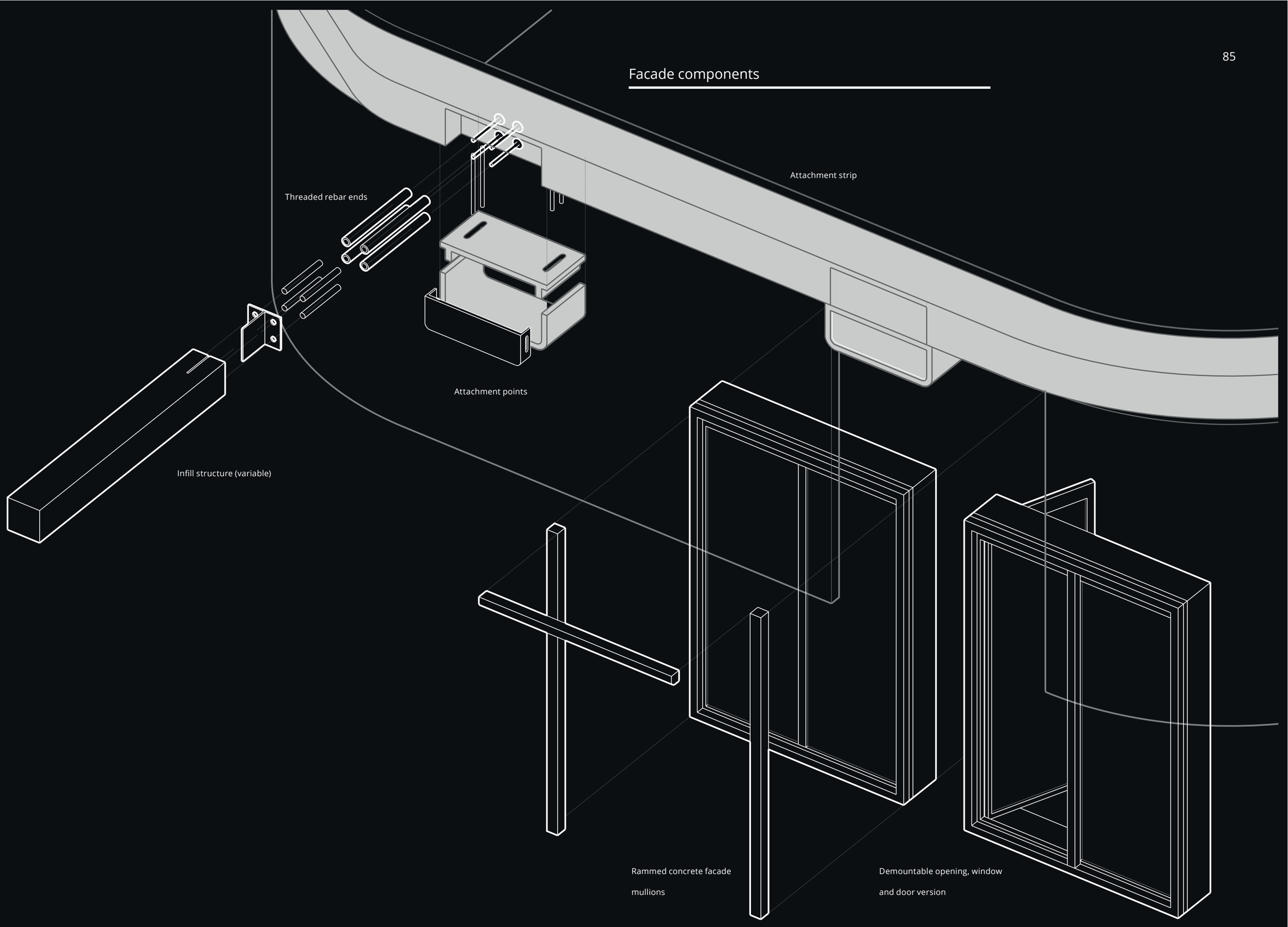
1 : 50 / First Floor (B)

Technical implementation

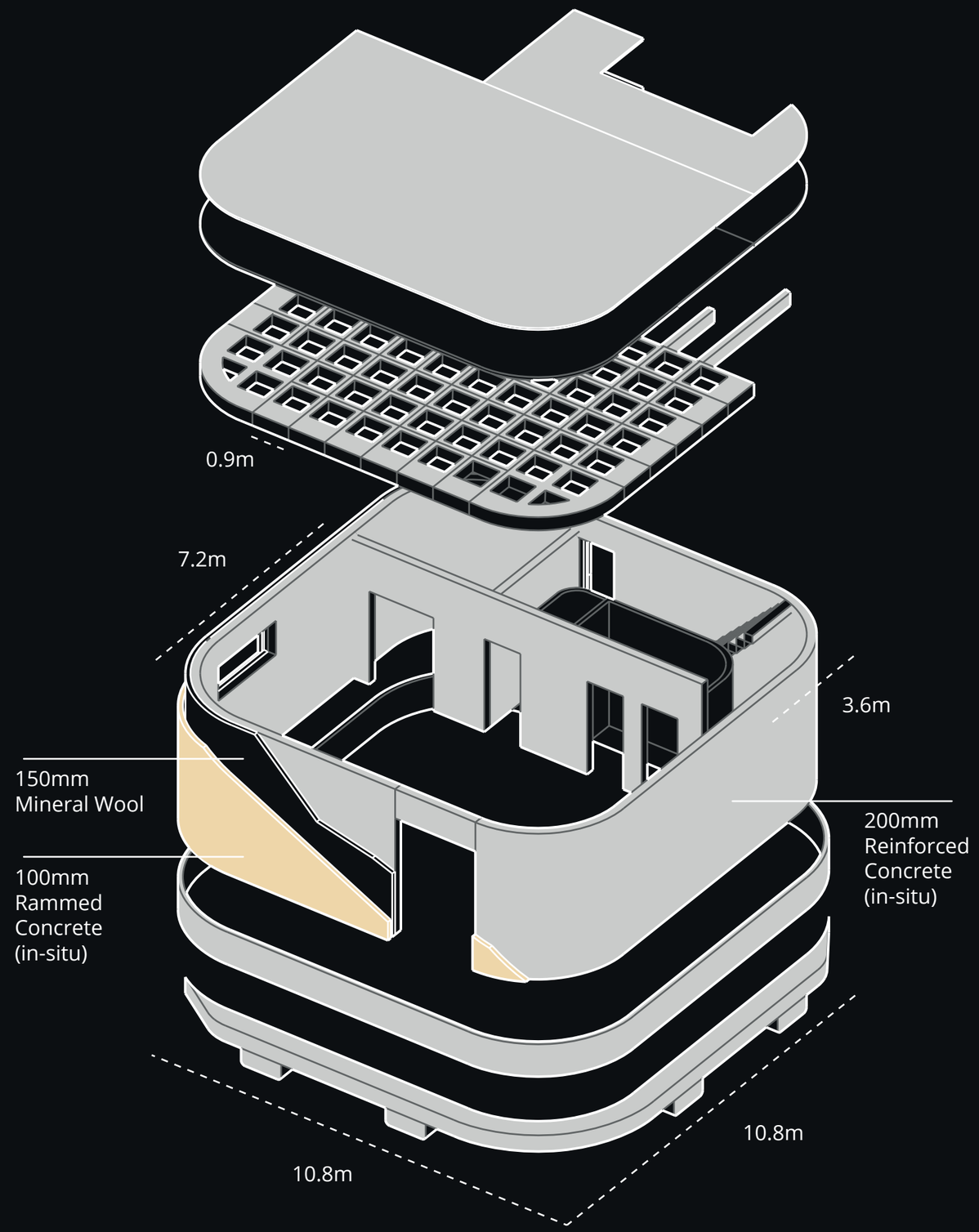
Structural elements



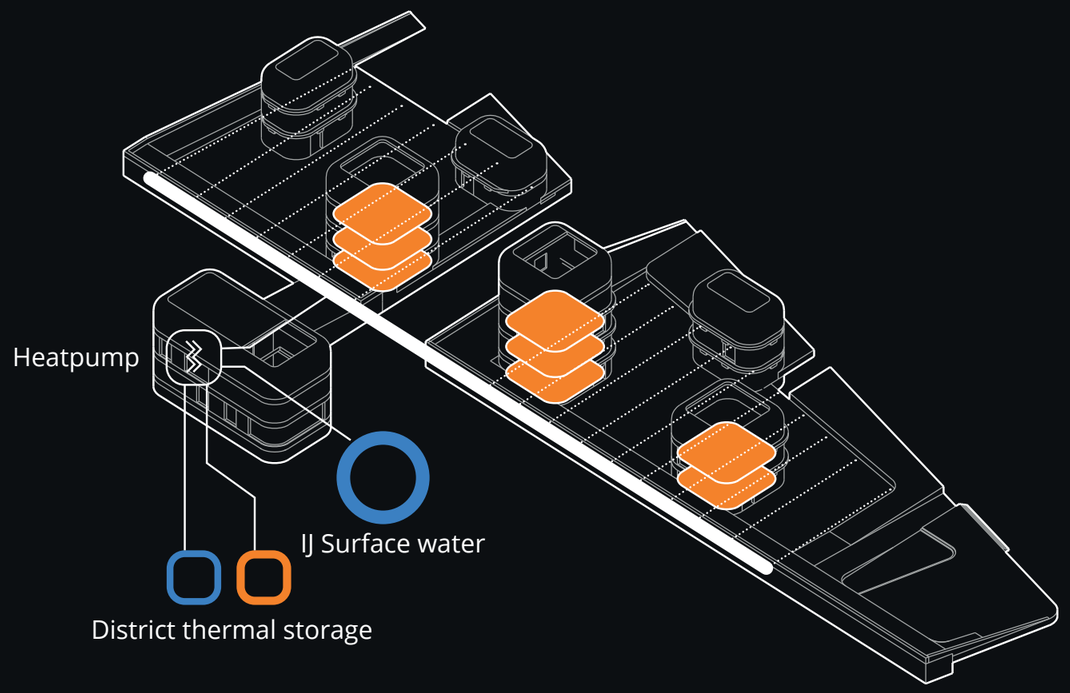
Facade components



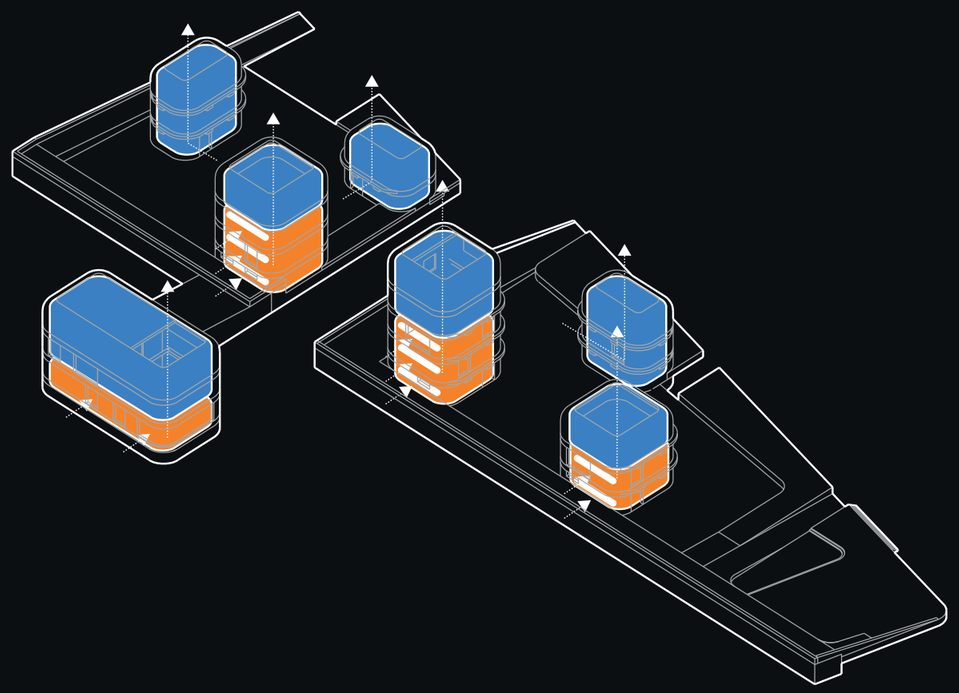
Core build-up

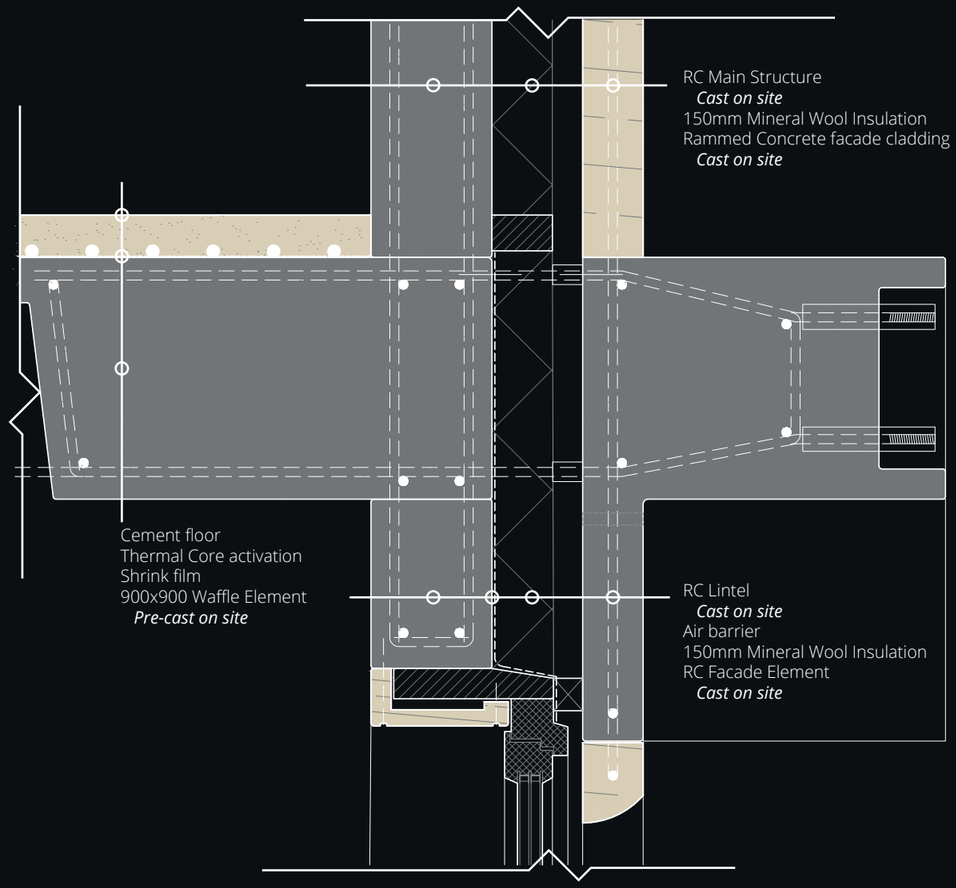


Thermal system

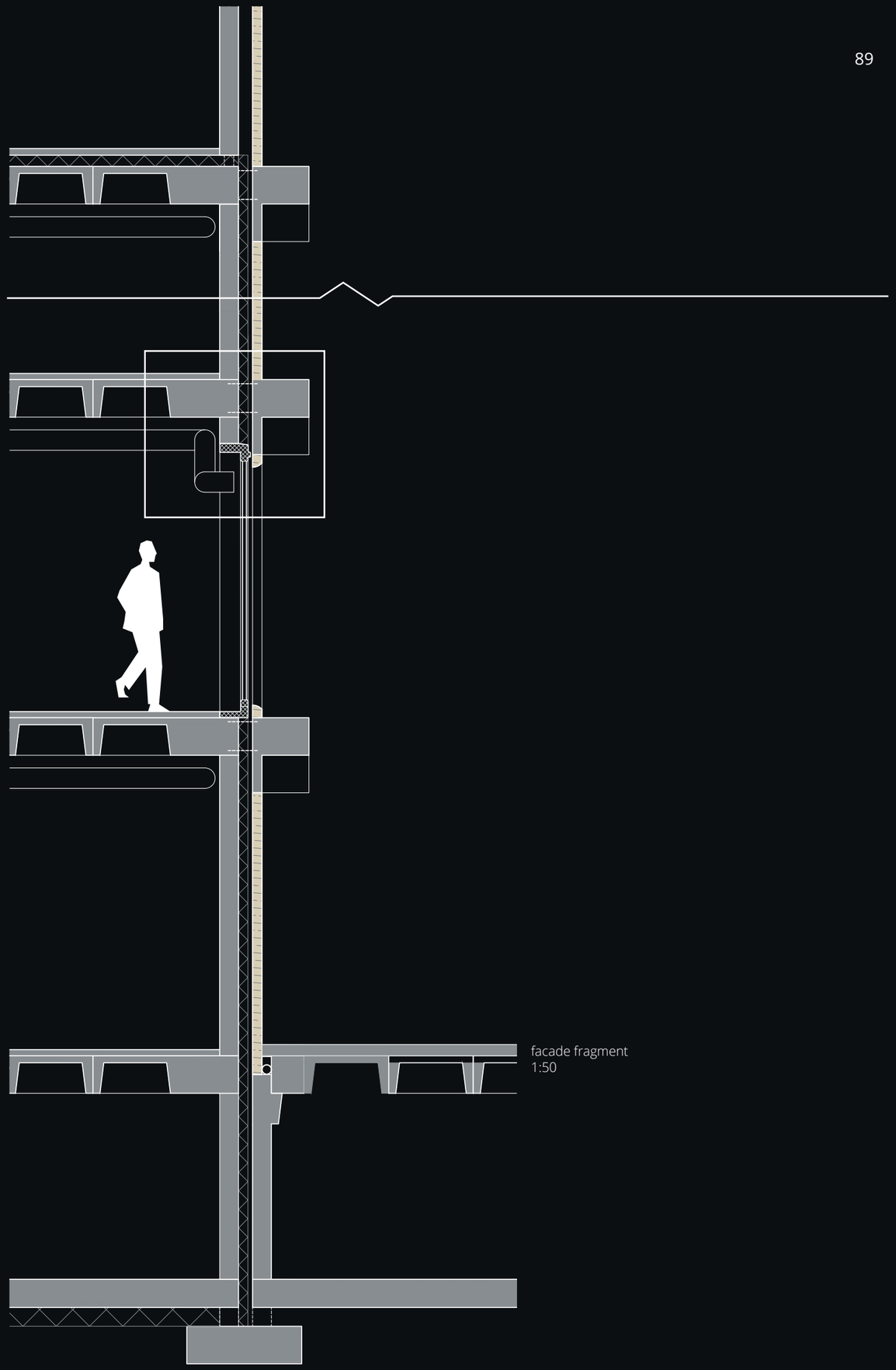


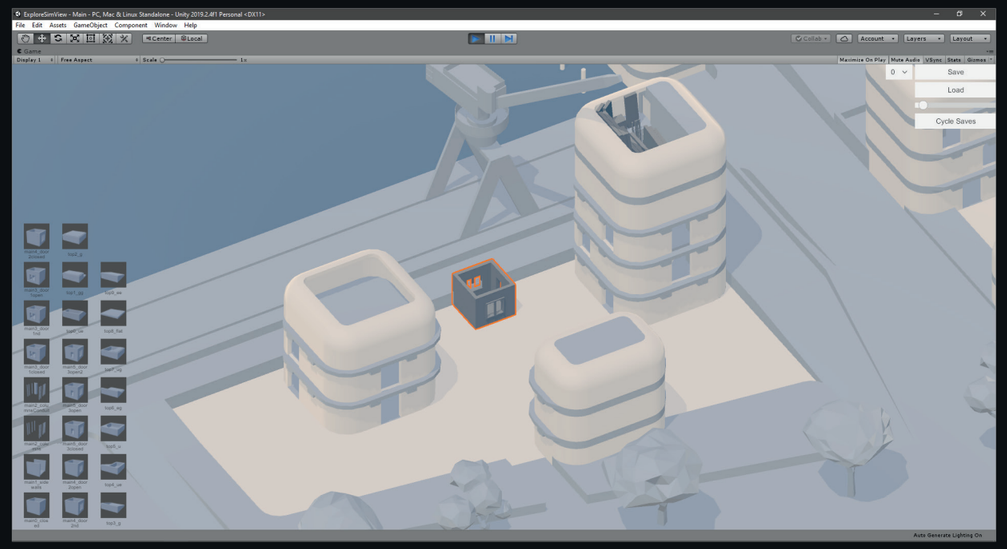
Ventilation & insulated segments

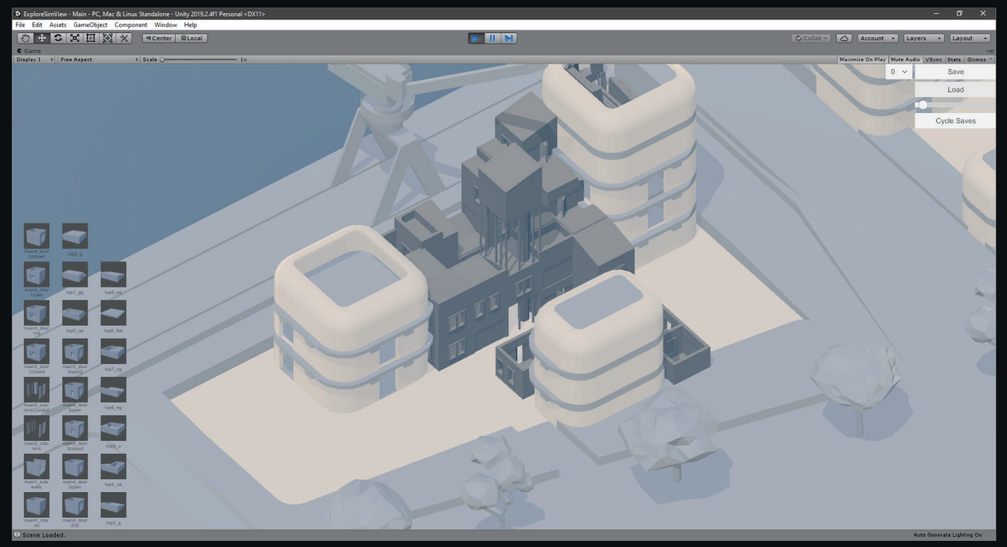


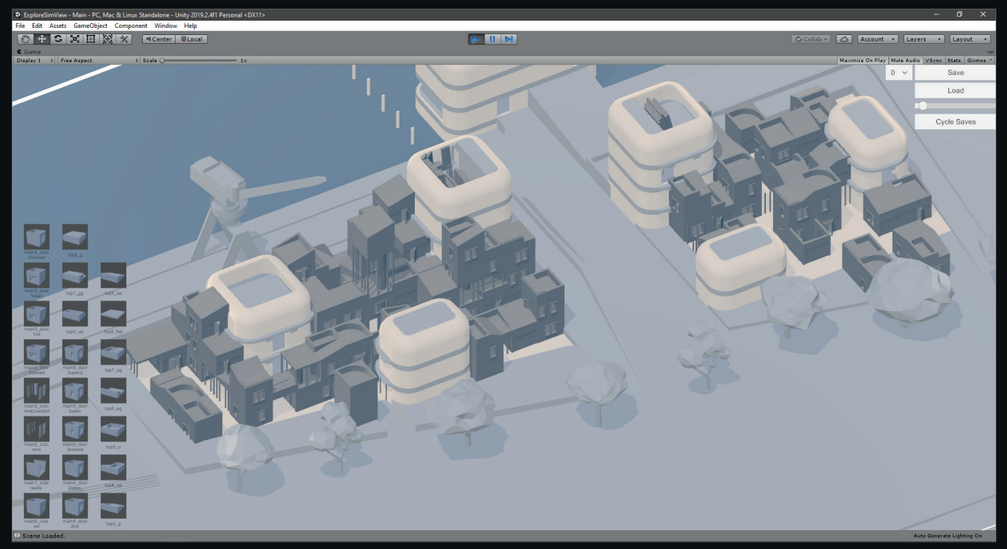


Detail Attachment Point
1:5









▶

Appendix:

Architectural system visualization.

Visualization code excerpt

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class BuildingPlacement : MonoBehaviour
{
    private static BuildingPlacement instance;
    public static BuildingPlacement Instance
    {
        get
        {
            if (instance == null)
            {
                instance = GameObject.FindObjectOfType<BuildingPlacement>();
            }
            return instance;
        }
    }

    private Transform currentModule;

    private SnapCollider snapCollider;
    private CraneAnim craneAnim;

    public bool inBuildingMode;
    private bool isSnapping;
    private bool UISnapToggle;

    [HideInInspector] public Vector3 mousePosWorld;
    [HideInInspector] public bool placed;
    [HideInInspector] public int numModules = 0;
    public const float _yStepSize_ = 3.3f;
    public const float _moduleSnapSpeed_ = .9f;

    void Awake()
    {
        craneAnim = FindObjectOfType<CraneAnim>();
        inBuildingMode = true;
    }

    // Update is called once per frame
    void Update()
    {
        numModules = ModuleManager.Instance.ModuleCount;
        UISnapToggle = GameObject.Find("/UI/Canvas/SnappingToggle").GetComponent<Toggle>().isOn;


        if (inBuildingMode)
        {

            if (currentModule == null) // currently not placing a module
            {
                if (Input.GetMouseButtonDown(1)) // delete module with right click
                {
                    Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
                    RaycastHit hit;
                    LayerMask layerMask = LayerMask.GetMask("Modules");
                    if (Physics.Raycast(ray, out hit, Mathf.Infinity, layerMask))
                    {
                        currentModule = hit.transform;
                        Debug.Log("Deleteing " + hit.transform.ToString());
                        ManagerCancelBuilding();
                    }
                }
            }

            if (currentModule != null && !placed) // placing a module
            {

                // Enable outline while placing
                currentModule.GetComponentInChildren<Outline>().enabled = true;


                // Cancel on right click
                if (Input.GetMouseButtonDown(1))
                {
                    ManagerCancelBuilding();
                }

                // Get world mousepositon
                mousePosWorld = GetMouseWorldPos();
                if (InSnapDistance() && UISnapToggle)
                {
                    GameObject snapModule = snapCollider.nearbyObjects[0].transform.parent.gameObject;
                    ModuleSnapToTarget(currentModule.gameObject, snapModule);
                }
                else                {
                    currentModule.position = Vector3.Lerp(currentModule.position, ModuleSnapToMouse(currentModule),
                                                                                    _moduleSnapSpeed_);
                }

                ModuleMoveY(currentModule);
                ModuleRotate(currentModule);

                // Place module on lmb
                if (Input.GetMouseButtonDown(0))
                {
                    if (IsPlaceable() || isSnapping)
                    {
                        ManagerPlace();
                        Debug.Log("Placing");
                    }
                    else
                    {
                        Debug.Log("Cannot place here");
                    }
                }

            }
        }

    }

    // GENERIC METHODS

    bool IsPlaceable()
    {
        if (currentModule.GetComponent<PlaceableBuilding>().colliders.Count > 0)
        {
            return false;
        }
        return true;
    }

    bool InSnapDistance()
    {
        if (snapCollider.nearbyObjects.Count > 0)
        {
            return true;
        }
        return false;
    }

    public Vector3 GetMouseWorldPos()
    {
        Vector3 mousePos = Input.mousePosition;


        LayerMask layerMask = LayerMask.GetMask("Groundplane");
        Ray castPoint = Camera.main.ScreenPointToRay(mousePos);
        RaycastHit hit;
        if (Physics.Raycast(castPoint, out hit, Mathf.Infinity, layerMask))
        {
            return hit.point;
        }
        else
        {
            return currentModule.transform.position;
        }

    }

    // MANAGER METHODS

    public void ManagerSetItem(GameObject c_moduleType, int c_typeIndex)
```

```
{
    placed = false;

    // Instantiate a new module
    currentModule = (Instantiate(c_moduleType)).transform;
    currentModule.name = “Module” + numModules + “-” + c_moduleType.name;

    // Set module to modules object, find its placeableBuilding component and assign its type.
    currentModule.parent = GameObject.Find(“Modules”).transform;
    currentModule.GetComponent<PlaceableBuilding>().ModuleType = c_typeIndex.ToString();
    currentModule.tag = “Module”;
    currentModule.gameObject.layer = 9;

    snapCollider = currentModule.GetComponentInChildren<SnapCollider>();
}

public void ManagerCancelBuilding()
{
    currentModule.GetComponent<SaveableObject>().DestroySaveable();
    Debug.Log(“Cancelling building”);
}

public void ManagerPlace()
{
    placed = true;

    // Pass on crane position data
    Vector3 placementpos = currentModule.position;
    craneAnim.SetCraneDestination(placementpos, 1f);

    // Disable outline when placed
    currentModule.GetComponentInChildren<Outline>().enabled = false;

    // Reset manager
    currentModule = null;
}

//MODULE METHODS
public void ModuleMoveY(Transform c_ModuleTransform)
{
    float yAxis = Input.GetAxis(“Mouse ScrollWheel”);
    if (yAxis != 0)
    {
        float yStep = yAxis * 10 * _yStepSize_;
        c_ModuleTransform.Translate(new Vector3(0, yStep, 0));
    }

    if (c_ModuleTransform.position.y > 0)
    {
        HeightLineManager.Instance.DrawHeightLines(c_ModuleTransform.gameObject);
    }
}

public void ModuleRotate(Transform c_ModuleTransform)
{
    if ((Input.GetMouseButtonDown(2) || Input.GetKeyDown(KeyCode.R)) && !isSnapping)
    {
        c_ModuleTransform.Rotate(new Vector3(0, 90, 0));
    }
}

public Vector3 ModuleSnapToMouse(Transform c_ModuleTransform)
{
    isSnapping = false;
    Vector3 currentModuleTargetPos;

    float gridSize = 3.6f;

    currentModuleTargetPos = new Vector3(mousePosWorld.x, currentModule.position.y, mousePosWorld.z);
    currentModuleTargetPos = new Vector3(Mathf.Floor(currentModuleTargetPos.x / gridSize) * gridSize + gridSize / 2,
                                         Mathf.Floor(currentModuleTargetPos.y / _yStepSize_) * _yStepSize_,
                                        Mathf.Floor(currentModuleTargetPos.z / gridSize) * gridSize + gridSize / 2);

    return currentModuleTargetPos;
}

public void ModuleSnapToTarget(GameObject c_currentModule, GameObject c_targetModule)
{
    isSnapping = true;

    // Get nodes of snapModule object
    List<Transform> currentModuleNodes = c_currentModule.GetComponentInChildren<SnapNodes>().nodesOpen;
    List<Transform> snapModuleNodes = c_targetModule.GetComponentInChildren<SnapNodes>().nodesOpen;

    //Find the nearest node to the mouse on target
    float nearestDist = Mathf.Infinity;
    Transform snapModuleTargetNode = null;

    int loopIndex = 0;
    int nearestIndex = 0;

    foreach (Transform node in snapModuleNodes)
    {
        float dist = Vector3.Distance(mousePosWorld, node.position);
        if (dist < nearestDist)
        {
            nearestDist = dist;
            nearestIndex = loopIndex;
            snapModuleTargetNode = node;
        }
        loopIndex += 1;
    }

    int targetIndex = nearestIndex;

    if (Input.GetButtonDown(“NodeCycle”))
    {
        targetIndex = (targetIndex + 1) % snapModuleNodes.Count;
        snapModuleTargetNode = snapModuleNodes[targetIndex];
    }

    // Find nearest node on current Module to target node
    nearestDist = Mathf.Infinity;
    Transform currentModuleTargetNode = null;
    foreach (Transform node in currentModuleNodes)
    {
        float dist = Vector3.Distance(snapModuleTargetNode.position, node.position);
        if (dist < nearestDist)
        {
            nearestDist = dist;
            currentModuleTargetNode = node;
        }
    }

    // Translate using vector from current node to target node.
    Vector3 snapVector = snapModuleTargetNode.position - currentModuleTargetNode.position;
    snapVector.y = 0;
    currentModule.transform.Translate(snapVector);

    //snapModuleTargetNode.GetComponentInParent<SnapNodes>().RefreshNodes(snapModuleTargetNode);

    }
}
```