Design Science

Mapping the types of modularity in open-source hardware

Kosmas Gavras[1] and Vasilis Kostakis[2,3]

[1] *Department of Architectural Technology, School of Architecture, National Technical University of Athens, Stournari and Patision 42, 106 82 Athens, Greece*
[2] *Ragnar Nurkse Department of Innovation and Governance, TalTech, Akadeemia tee 3, 12618, Tallinn, Estonia*
[3] *Berkman Klein Center for Internet & Society, Harvard University, 23 Everett Street, Cambridge, MA 02138, USA*

Abstract

The importance of intangible code modularity in open-source software, as well as of tangible product modularity in proprietary hardware, is widely acknowledged. Nevertheless, modularity in open-source hardware (OSH) remains under-researched. This article first describes qualitatively different types of modularity based on two OSH case studies and then quantifies each type of modularity, following a unified network-based approach. The results are discussed and compared within each case to test the 'mirroring hypothesis', and between cases to evaluate the impact of physical against intangible modularity types. The ultimate goal is to prompt a discussion into a wide but under-explored subset in OSH.

Key words: open-source hardware, modularity, design modularity, fabrication modularity, mirroring hypothesis

Received 29 June 2020
Revised 08 May 2021
Accepted 12 May 2021

Corresponding author
K. Gavras
kgavras@mail.ntua.gr

Des. Sci., vol. 7, e13
journals.cambridge.org/dsj
DOI: 10.1017/dsj.2021.11

1. Introduction

The open-source hardware (OSH) phenomenon is described and distinguished from proprietary hardware by two main constituents. First is the application of an alternative mode of knowledge production and management – best exemplified by open-source software (OSS) – in the material realm (Shirky 2005; Balka, Raasch, & Herstatt 2009; Raasch, Herstatt, & Balka 2009; Li *et al.* 2017; Boujut 2019). And second – partly fuelled by the first – is the still-emerging distributed manufacturing paradigm that has resulted from the maturation and affordability of digital fabrication technologies (Kostakis *et al.* 2015; Pearce 2015).

The OSH phenomenon can be defined and its degree of openness evaluated following a number of published approaches (Bonvoisin & Mies 2018; Gavras 2019; Bonvoisin *et al.* 2020; Open Source Hardware Association 2020). However, OSH is distinct from other schemas that, by definition, feature limited openness. For instance, in open-architecture products, certain parts of the platform and all basic functions are closed and proprietary, while certain less crucial interfaces are open for add-ons from third parties (Koren *et al.* 2013).

While the application of modularity in the OSH practice is common (see Bonarini *et al.* 2014; Kostakis & Papachristou 2014; WikiHouse 2018a; Open-Structures 2019), theoretical approaches are recent and limited (see Gavras 2019;

Kostakis 2019; Bonvoisin *et al.* 2020). Conversely, the OSS literature concerning modularity dates back to the 1970s (Parnas 1972). Also, the corresponding literature on proprietary hardware is vast (see the literature review by Bonvoisin *et al.* 2016) and well-established (Starr 1965).

For instance, modularity has been the object of extended research in the automotive industry (Johnson & Broms 2000; Fellini *et al.* 2004; Pandremenos *et al.* 2008; Cabigiosu, Zirpoli, & Camuffo 2013; Albers *et al.* 2019), as well as the aircraft (Frigant & Talbot 2005), computer hardware (Baldwin & Clark 2000) and prefabrication industries (Halman, Voordijk, & Reymen 2008). Modularity bears the cross-industry benefit of cost reduction due to economies of scale (Fixson 2007; Hackl *et al.* 2019). Ulrich (1994) highlights the ease of reverse-engineering as a negative effect of modularity. Sanchez & Mahoney (1996) argue that, following a modular product design approach, authority is exercised during the design of product architecture before component design development occurs. However, the approach and context of modularity in proprietary hardware often contrasts with basic constituents of the open-source model. In the open-source model, organisation and authority is bottom-up (Ball & Lewis 2018) as most of the tasks are self-selected by contributors (Boisseau, Omhover, & Bouchard 2018), and 'copying' technical knowledge is a common practice associated with overall positive impact (Raymond 2000). Moreover, economies of scale are incompatible with the idea of localised manufacturing (Kostakis *et al.* 2015).

The distinction between the open-source and proprietary realms has been evaluated in the OSS literature; MacCormack, Baldwin, & Rusnak (2012) paired open-source and proprietary software of similar size and function, showing that the code of OSS is more modular by a factor of six. MacCormack, Baldwin, & Rusnak's (2012) finding echoes the 'mirroring hypothesis', according to which product architecture and organisation architecture are dual in structure. Baldwin & Clark (2002): 5, Baldwin & Clark 2006) articulate the importance of modularity as a structural constituent of the open-source model, in terms of both product and organisational architecture:

(i) 'Modularity makes complexity manageable' through the decomposition of a complex task into simpler tasks, and the selection of optimal solutions from a pool of interchangeable options;
(ii) 'Modularity enables parallel work' through the division of a monolithic system into numerous independent modules and
(iii) 'Modularity is tolerant of uncertainty'. Certain elements of code structure can be swiftly altered at any time and in potentially unforeseen ways.

This article argues that modularity is applicable to various elements and aspects of OSH production; from (co)designing and (co)manufacturing to using hardware. However, our focus is not on how modularity is created, but rather on how modularity within the OSH realm may manifest as a design practice, an organisational principle/property or, sometimes, as both. We thus aim to map, assess, and classify the different types of modularity in OSH.

This article is organised as follows: Section 2 describes the research approach and method. Section 3 discusses qualitative and quantitative definitions of modularity. Section 4 outlines five main types of modularity and Section 5 quantitatively assesses those types, drawing from the selected cases. Section 6 discusses the results of the research and highlights its limitations. Section 7 presents the gist of the current paper and proposes future directions for research and practice.

2. Research design

2.1. Research scope

We discuss modularity in the OSH framework by:

(i) Articulating a generic, qualitative and quantitative definition of modularity from the literature. The aim is to describe modularity irrespective of the subject (e.g., hardware,[1] community), size or industry (aircraft, automotive or building).
(ii) Tracing a comprehensive modularity typology based on cases and literature, with an emphasis on the less-explored, intangible side of OSH.
(iii) Quantifying the modularity levels of the selected cases to provide preliminary evidence regarding, first, the relation of respective or alternative modularity types between cases, and, second, the relation among modularity types concerning the 'mirroring hypothesis' within cases.

2.2. Research significance

The present research is primarily useful in addressing a gap in the OSH literature regarding conceptualisations of modularity. Some of the existing definitions of modularity are specific to a certain subject or industry and are thus unhelpful in terms of a wider understanding of the phenomenon. For example, Emanuel, Wardoyo, & Istiyanto (2011) present a definition that is exclusively software-specific; Salvador (2007) conceptualises product family modularity in a way that is only partially extendable to the single product; Braha & Bar-Yam (2007) conflate modularity with other properties. Additionally, most quantitative modularity studies adopt comparative methods concerning software or proprietary hardware. Comparative methods are useful, but arguably inadequate for evaluating OSH thoroughly. Consider a structure that is more modular by a factor of six compared to another: both might be almost nonmodular on an absolute scale. Therefore, absolute measures of modularity are also indispensable in practice and research.

However, any quantification method is meaningless without a guide on the actual subjects and types of modularity. To the authors' knowledge, existing OSH literature does not cover the classification and assessment of modularity types. Consequently, our ambition is to set common ground for future research required in order to understand an unexplored dimension of OSH.

2.3. Research methods

To address the research goals, two methods were used. Regarding the qualitative aspects of our goals, hybrid instrumental-collective case study research was employed (Stake 1995). Case selection was based on the authors' strategy and background, and excluded mechanical and electronic devices. In-depth analysis of

[1]Henceforth, for the purpose of clarity, the terms 'product' and 'hardware' will point to material artifacts in the proprietary and open-source realms, respectively.

a less represented OSH subset may offer advances of cross-disciplinary importance (Papalambros 2015). The first case is OpenStructures (2020), a community of industrial designers making furniture, as well as other hardware; and the second is WikiHouse (2020), a community for open-source housing. In terms of the age and volume of designed hardware, both projects are two of the most mature cases in the described OSH subset. The first case represents conventional design through standard computer aided design (CAD) systems, and the second exemplifies a growing trend of design by coding. Data were collected from project websites and secondary sources and databases. Direct interpretation (Stake 1995) was used to process data and to extract new meanings concerning the cases.

For the quantitative aspect of the research, tools and methods from network science were employed to approach a quantification method. The algorithm for modularity analysis proposed by Blondel *et al.* (2008) was used in the Gephi[2] software. Data collection was performed manually from the project's website in the case of OpenStructures, and through a custom script made by the authors in the case of WikiHouse. In the interpretation of the quantitative results, an axiomatic principle was held that a more modular hardware, *ceteris paribus*, is a more open hardware, in analogy with the relevant literature on OSS. However, modularity and openness in corporate environments are also associated with product imitations and negative effect on financial turnover (Ulrich 1994; Fixson & Park 2008).

3. Modularity in the OSH

This section focusses on the criticisms and the emphases of different viewpoints on modularity. The Section 3.1 articulates generic qualitative definitions of modularity. The second Section 3.2 explores generic quantitative definitions.

3.1. Qualitative definition

According to Persson & Åhlström (2006), the term 'degree of modularity' describes the degree of coupling among different modules in a product. Consequently, in the case that every module can couple with any other module in the system, the product is highly modular. But the highly coupled product is also highly monolithic, no matter how many parts comprise the product (Figure 1, left). If the concept of modularity is limited to the simple division of the whole into parts, it could be inferred that hardware, broadly speaking, has always been modular (Brusoni & Prencipe 2001). Yet, on the contrary, modularity in OSS does not simply refer to the division but also to the quality of the division (Raymond 2000). However, even this definition is imprecise and incomplete, as more than one of the existing network metrics (e.g., modularity index, clustering coefficient, density, centrality and degree distribution) refers to qualitative aspects of the division or coupling of a structure, leading to ambiguity in terminology.

For example, the common use of the qualitative terms 'loosely coupled' and 'tightly coupled' to describe modular and nonmodular networks, either organisations or products, is imprecise. Technically, the misconception is expressed by the association of modularity with the clustering coefficient (see Braha & Bar-Yam

[2]https://gephi.org/

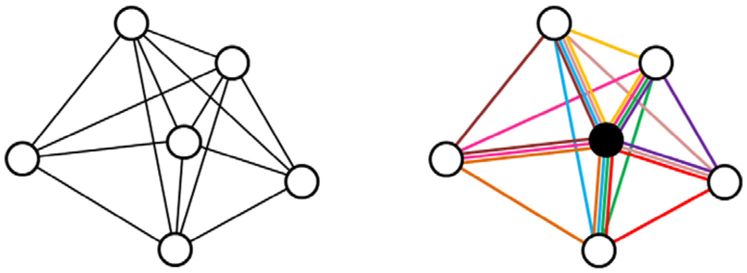


Figure 1. Left: High degree of coupling within a monolithic structure.[3] Right: High clustering coefficient within a monolithic structure. $C = \frac{\text{actual closed triangles}}{\text{possible closed triangles}}$ (Colors signify actual closed triangles per the black-filled node).[4]

2007). This is a metric concerning a specific qualitative dimension of coupling. 'Clustering coefficient [0–1]' expresses the incidence of closed triangles among the nodes of a network. A clustering coefficient value of zero is the equivalent of an unconnected network, whereas the value of 1 stands for a completely connected network (Figure 1, right). Obviously, neither value corresponds to a modular network, which may lie anywhere between the extreme values. Clustering is an important metric concerning network cohesiveness, but cohesiveness is not directly related to the property of modularity. Nevertheless, the clustering coefficient has been associated with certain participatory properties of OSH collaborative networks (Bonvoisin *et al.* 2018).

Product design literature (Ulrich 1995; Erens & Verhulst 1997) offers a more focused definition with the following: 100% modularity in design means that one function is allocated to one single module, while 0% means that all functions are allocated to different modules. However, this definition misses two important aspects: extendability beyond products and multi-level organisation. When mapping the modularity of a collaborative network, the function-based definition is obsolete, apart from the special case that the collaborative network is a perfect 'mirroring' of the product functions. However, even in product design, function discretisation is neither always applicable (Ulrich & Seering 1990), nor the sole known modularisation principle (Bonvoisin *et al.* 2016). Secondly, in real-world cases, modularisation is rarely a single-level organisational process. Modules can be nested in other modules, and so on: a function internally and recursively contains other functions, coupled in a modular way. The power of nested modularisation beyond OSS is exemplified in the parable of Hora and Tempus (Simon 1962).

Another example of an over-specialised modularity definition is Salvador's (2007) definition for product families, which is built upon the concepts of commonality (common parts), separability (separable parts) and combinability (reconfigurable parts). According to Salvador, the unit of reference is the product family, but despite that, some of these concepts are also fully or partially transferable to the single product. However, the concepts of commonality and combinability do not transfer clearly to the modularity definition of the collaborative network.

[3]image processing: authors, image source: authors.
[4]image processing: authors, image source: authors.

Modularity is thus a specific quality of coupling among the components of a system, whether for hardware or people. Outlining a definition at the intersection of several realms, modular structure – as simulated by nodes and edges – features a certain discretisation into modules, characterised by higher internal connectivity among module nodes, rather than external connectivity among modules (Baldwin & Clark 2000; Newman & Girvan 2004; Blondel *et al.* 2008; Jung & Simpson 2016). Connectivity is more than the mere sum of connections (edges), as outlined in the following quantitative approach.

3.2. Modularity metrics

Various methods to assess modularity exist (Guo & Gershenson 2003), but none are universally accepted. Most methods are based on the design structure matrix (DSM), which is an analytical tool used to simulate the structure of complex systems (Steward 1981; Eppinger *et al.* 1994; Sharman, Yassine, & Carlile 2002; Sosa, Eppinger, & Rowles 2004; Sosa, Eppinger, & Rowles 2007; MacCormack, Baldwin, & Rusnak 2012). Modularity metrics based on DSMs have been the subject of extensive research (Huang & Kusiak 1998; Newcomb, Bras, & Rosen 1998; Sosa, Eppinger, & Rowles 2000; Guo & Gershenson 2004; Hölttä-Otto & de Weck 2007; Yu, Yassine, & Goldberg 2007; Hölttä-Otto *et al.* 2012). According to Baldwin, MacCormack, & Rusnak (2014), modularity methods based on DSMs are superior to network-based methods. The main advantage of DSM-based methods is the analysis of directed connections among nodes.

Baldwin, MacCormack, & Rusnak's (2014) assumption is that technical systems are always directed. However, in the case of a spatial attachment interface between hardware modules, the cumulative dependency between modules is bidirectional and, by extension, undirected. Asikoglu (2012) analysed a set of electro-mechanical household products and concluded that 85% of them were composed of directly contacting interfaces. Moreover, our cases feature exclusively spatial attachments, which are not common within either electronic or mechanical OSH subsets. Therefore, the advantage of DSMs over network-based methods is less important in hardware, and specifically in the examined OSH subset.

Collaborative networks, as for social networks in general, are undirected structures. Designer 'A' cannot collaborate with designer 'B' without the reverse being true. To extend the argument, even in software, when function 'B' is dependent on function 'A', if the dependent function is radically modified, the independent one will also have to be modified. Conversely, any modification in any function, whether dependent or independent, does not automatically transfer to any other function if the interface remains intact. Hence, the directionality of DSM-based analysis methods is meaningless in the described context.

Besides directionality, Milev, Muegge, & Weiss (2009) and Jung & Simpson (2016) have recognised different but important limitations in various DSM-based metrics. Notable cumulative limitations of the DSM-based metrics (Table 1) include the following:

(i) Modularity levels are dependent on the size of the analysed object, narrowing comparative evaluation to structures of equal size and precluding analysis of a single project in chronological order.

Table 1. Limitations of DSM-based modularity metrics

		Limitations			
DSM-based metrics	Reference	Dependent on size (i & ii)	Binary state of connections (iii)	Ignores discrete architecture (iv)	Single module metric (v)
Propagation cost	MacCormack, Baldwin, & Rusnak (2012) MacCormack, Rusnak, & Baldwin (2006)		•	•	
Clustered cost	MacCormack, Rusnak, & Baldwin (2006)	•			
Relative clustered cost	Milev, Muegge, & Weiss (2009)		•		
Module strength indicator	Whitfield, Smith, & Duffy (2002)				•
MS	Sosa, Eppinger, & Rowles (2003)		•		
MG&G	Guo & Gershenson (2004)				
Minimum description length	Yu, Yassine, & Goldberg (2007)	•			
MJ&S	Jung & Simpson (2016)				

MG&G metric is dependent on number of modules. MJ&S metric is composed from three separate indices arbitrarily weighted.

(ii) The absence of fixed value boundaries results in indeterminacy about what is modular in absolute terms.

(iii) The modelling of the interface is over-simplistic (Binary DSMs). Most DSM modularity metrics are based on a binary state of connection between elements which may be connected or unconnected. However, in real-world hardware, most interfaces are more complicated and feature connections of varying dependency. Beyond hardware, collaborative networks are shaped with connections of surprisingly variable weight (Bonvoisin *et al.* 2018). The assumption that neither pair of developers has collaborated more than once in an OSS project, for example, is extremely flattening.

(iv) Calculation of modifiability instead of modularity. The propagation cost method emphasises dependency among all the nodes of the entire network rather than network-discrete architecture. While modifiability is one of the products of modularity, the inverse is not always true.

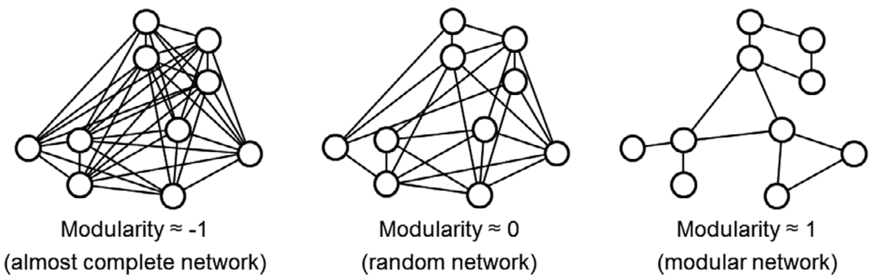


Figure 2. Indicative examples approaching the boundary values of the modularity index in network science (Newman & Girvan 2004; Blondel *et al.* 2008).[5]

(v) Calculation of single module modularity rather than that of the entire architecture.

In network-based methods, the modularity index is clearly defined and used for a wide range of physical and intangible structures (Newman & Girvan 2004; Blondel *et al.* 2008). The modularity index as formulated in network science has rarely been used in OSS or proprietary hardware (see Paparistodimou *et al.* 2020) and even less so in OSH. The modularity index measures the quality of the division, that is, the density and weight of coupling inside modules compared to that among modules (Newman 2004; Newman & Girvan 2004; Blondel *et al.* 2008). Modularity is a scalar value ranging between -1 and 1 (Blondel *et al.* 2008). The value of -1 expresses the state where all 'module' parts are interconnected within a monolithic system; the 0 value represents a state of random connectedness lacking any substantial modularisation; and the value of 1 describes a highly modular system (Figure 2).

The approach proposed for calculating modularity in OSH concentrates the following advantages, which are critical in comparison to most of the existing methods (Table 1):

(i) The size-independent metric allows for comparison between different modularity types of the same projects or similar types between different projects, or even studying the evolution of a specific modularity type over time.
(ii) The index is an absolute scalar value delineating what is actually modular.
(iii) The evaluation of weighted dependencies is critical in the precise analysis of connectivity for many subjects relating to modularity, including collaborative networks.
(iv) The evaluation of inter- and intra-module dependencies leads to a thorough understanding of the qualitative property of modularity beyond mere divisibility and detected number of modules.

4. Types of modularity in OSH

4.1. Presentation of the cases

Following the qualitative and quantitative generic definitions of modularity above, this section introduces the selected cases and discusses the types of modularity observed in practice.

[5]Image processing: authors, image source: authors.

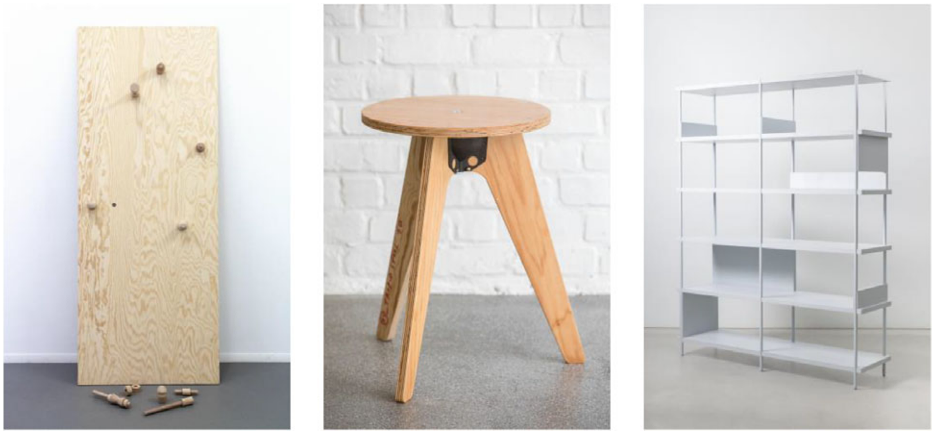


Figure 3. OpenStructures hardware. From left to right: coat rack, stool and bookcase.[6]

4.1.1. OpenStructures

The first selected case is OpenStructures, an open-source project for making modular furniture and other generic hardware (Figure 3). OpenStructures began in 2006 as a student project at the Institute without Boundaries in Toronto, Canada. In September 2009, Thomas Lommee, designer and founder, organised an exhibition showcasing the concept and some initial prototypes, and presented an open call for the collaborative development of the project. OpenStructures is self-defined as an open modular system for building hardware, inspired by the modularity of OSS (OpenStructures 2019).

The centerpiece of the OpenStructures system is the OpenStructures grid, a shared geometrical grid built up from the superposition of a rectilinear grid, diagrid and polar grid. The OpenStructures grid is a shared geometric platform for designing parts and their interfaces, which then can be reassembled in different combinations to form new hardware. The featured parts are the lowest-order elements, and usually do not have any functional independence.

In 2019, OpenStructures underwent a major reform, withdrawing most of the older projects from the online database and adding newer projects and designers. The former vertical organisation of artifacts as parts, components and structures was converted to parts that, assembled, comprise apps. In OpenStructures, apps describe objects, appliances, furniture or any other hardware assembly from two or more compatible parts. Macroscopically, the restructuring resulted in a greater level of interaction between designers: in the previous phase most assemblies were composed of parts from the same designer.

4.1.2. WikiHouse

Our second selected case, WikiHouse, was initiated by a nonprofit foundation in 2011 as 'an open-source project that re-invents the way we make homes' (WikiHouse 2018a). Since then, architects, builders and users have been locally

[6]Source: https://www.openstructures.net/applications, license: OS noncommercial 1.0 (downloadable on June 1st, 2020).

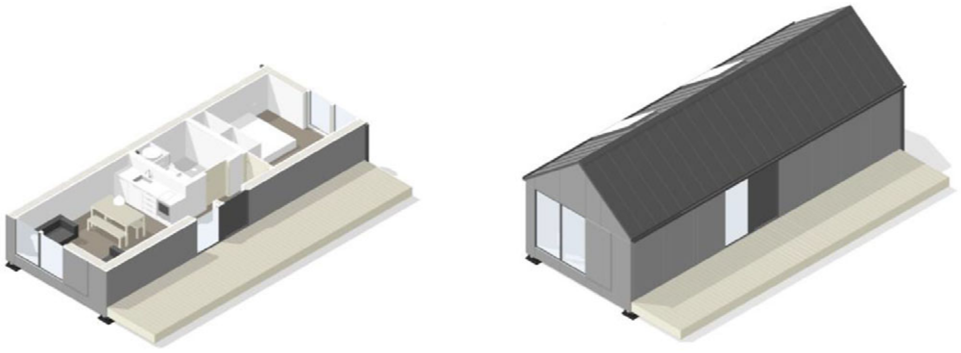


Figure 4. A housing type that can be reconfigured by Wren technology.[7]

constructing pilot wikihouses all over the world, based on the distribution of digital fabrication media.

WikiHouse draws a clear parallelisation between OSS and architectural design, as ‘digital design allows every home to be designed as code; instantly customised to its site and user’ (WikiHouse 2018b). It is a collaborative project ‘by everyone for everyone’ (WikiHouse 2018b). WikiHouse is predicated upon the concept ‘Design Global, Manufacture Local’ (Priavolou & Niaros 2019), in which design is developed, shared and improved openly and globally, while manufacturing takes place locally (Kostakis *et al.* 2015).

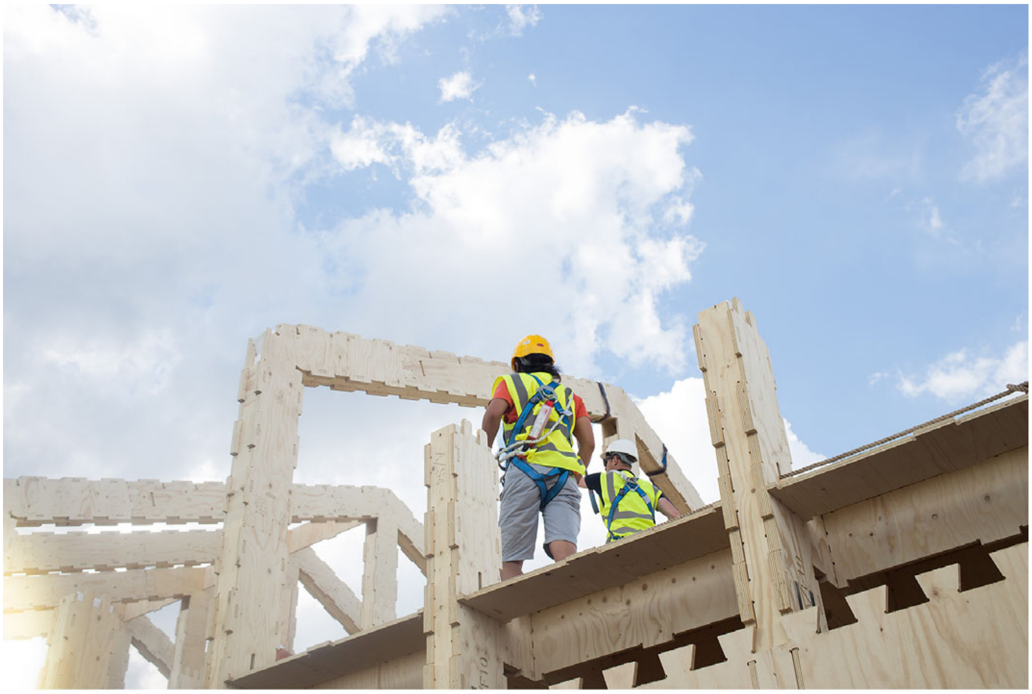
WikiHouse’s design knowledge is organised from lower- to higher-order elements in tools, technologies and types. Types are building typologies, designed and configured by technologies (Figure 4). Tools are accessories for assembly and construction, discussion of which is beyond the scope of this article. The most impactful category of WikiHouse design knowledge is technologies.

The core technology in the WikiHouse ecosystem is known as Wren. Wren is a design algorithm, developed in a visual programming environment, aiming at two goals. The first goal is the parametric design and customisation of housing types. For instance, the user can adjust the outer dimensions (width, length, height and roof height) of the house, or the number and location of openings (doors and windows) (Figure 4). Based on the outcome of the first goal, the second is to automate the production of detailed fabrication data regarding structural chassis and cladding. In the case of the chassis, the algorithm forms a three-dimensional (3D) structural frame based on the designed outer shell of the first step. Subsequently, that frame is automatically subdivided into indexed interlocking flat-cut elements (Figure 5).

4.2. Typology of modularity

Before exploring the modularity types observed in the selected cases, we now introduce our starting point, drawing from the literature on product design and management studies.

[7]Source: https://github.com/wikihouseproject/Microhouse, license: CC BY-SA 3.0 (downloadable on June 1st, 2020).



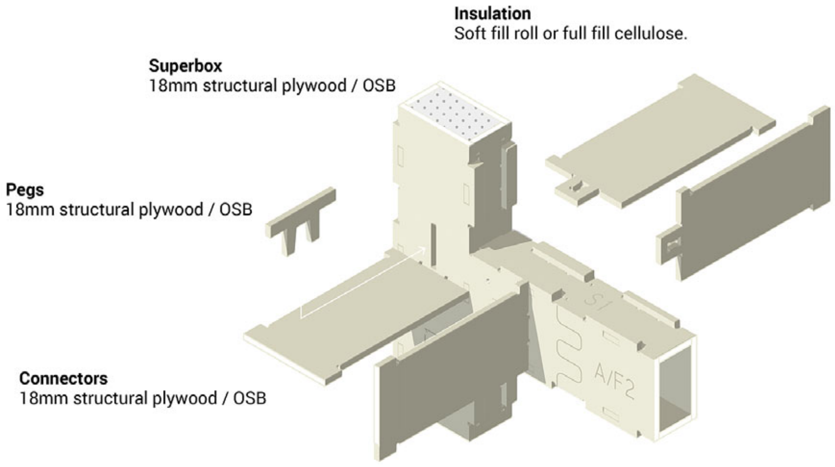

Insulation
Soft fill roll or full fill cellulose.
Superbox
18mm structural plywood / OSB
Pegs
18mm structural plywood / OSB
Connectors
18mm structural plywood / OSB
S1
A/F2

Figure 5. The structural chassis is automatically designed by the Wren algorithm as a three-dimensional puzzle of interlocking flat elements.[8]

4.2.1. *Modularity types in product design literature*

Kostakis (2019: 9), building on Baldwin & Clark (2003), distinguishes three types of modularity pertaining to OSH development:

(i) 'Modularity in artifact design: It refers to the decomposability of an object into smaller subsystems that may be designed independently but still function together as a whole'.

[8]Source: https://github.com/wikihouseproject/Wren/raw/master/Images/Connectors-01.png, license: MPL 2.0 (downloadable on June 1st, 2020).

(ii) ‘Modularity in production processes: It refers to the way that the artifact is produced. Production includes the whole value chain of an artifact, from its design to its manufacturing and distribution. Modularity in production is often a result of increased modularity in design (Brusoni & Prencipe 2001). Modularity in design is connected to the outsourcing of tasks; however, it is not clear which begets the other (Campagnolo & Camuffo 2010)’.

(iii) ‘Modularity in use: It refers to the possibility that the users may have to mix and match modules so that the artifact suits their needs as well as their ability to maintain them’.

Modularity in use is closely related to the ideas of separability and combinability, which build the concept of product system modularity (Salvador 2007). The difference lies with the actor: the user or the firm. Modularity in use is also similar to the concept of open-architecture products by Koren *et al.* (2013). Fixson (2007: 90) describes another type of modularity related to (ii):

(iv) Modularity in organisation: ‘the term organisation here is intended to include both intra-firm and inter-firm organisational structures’.

The types described here are also parts, or, better, phases in the Life Phases Modularisation method developed by Krause *et al.* (2014). The abovementioned modularity types are not mutually exclusive and their uncoordinated combination can lead to conflicts in product development (Greve, Rennpferdt, & Krause 2020). Greve, Rennpferdt, & Krause (2020) have synthesised a method to resolve conflicts identified through the Module Process Chart. Conflicts are resolved by the harmonised amalgamation of the different modularisations of each phase.

The types described above reveal the strong hegemonic position of product modularity. Erixon, Von Yxkull, & Arnström (1996) define product modularity as a precondition for modularity in the production process itself. According to Sanchez & Mahoney (1996), product modularity facilitates design modularity and organisation modularity. Even the method proposed by Greve, Rennpferdt, & Krause (2020) to resolve modularisation conflicts is based on the conception of different physical product modularisations corresponding to each modularity type or phase. In a holistic hierarchical interpretation, any other type can be modular as long as the product is modular (Figure 6).

4.2.2. Modularity types derived from the selected OSH cases

In addition to the modularity types determined by the literature, the examination of the selected cases of the present study reveals new types of modularity, or redefines existing ones. WikiHouse exemplifies design modularity beyond physical modules. For instance, the components of the Wren design algorithm defining the outer form of the housing type do not correspond exclusively to any single physical module, while their algorithmic structure is potentially modular. Generally, it is common in the building industry for a function (e.g., waterproofing) to be distributed across many physical modules (roof, cladding, doors and windows, etc.). It is also common for a physical module (e.g., a window) not to single-handedly perform a whole function (daylighting, rainproofing, ventilation and visibility). An additional example of potential intangible design modularity, and

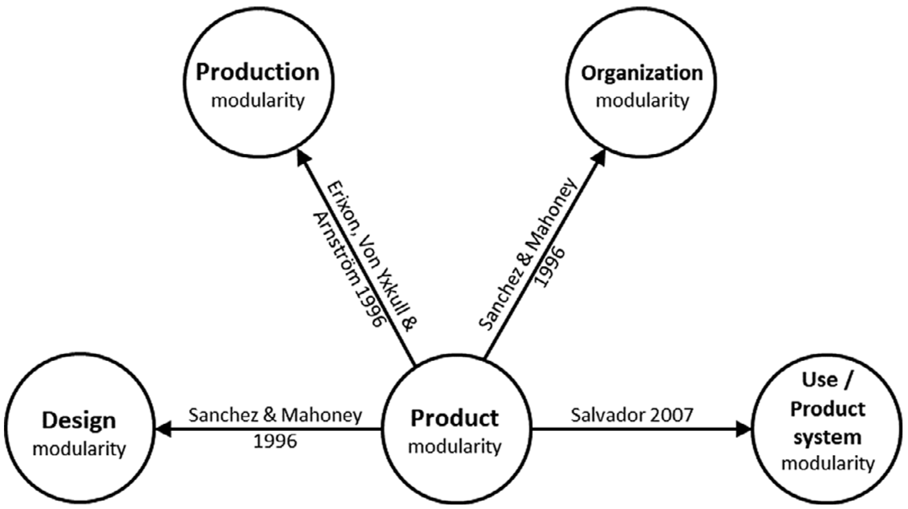


Figure 6. Schematic deployment of modularity types according to their relation in the product design literature (Erixon, Von Yxkull, & Arnström 1996; Sanchez & Mahoney 1996; Salvador 2007).[9]

one of the most promising plans of the WikiHouse team, is PlanX[10]. PlanX is a beta-version design algorithm that automatically checks conformity with building regulations for each instance of a given house type. Consequently, intangible design is a new subject of modularity, bringing OSH closer to the data-driven code structure of OSS. Intangible design modularity – as defined in this paragraph – is wider in scope than merely looking at the level of granularity in the information content of physical modules (Maier, Eckert, & Clarkson 2017).

Next, the compound nature of the Wren algorithm reveals the absence and importance of another modularity type. For Wren, the processes of conceptual design and design for fabrication are combined and merged into a single algorithm. The amalgamation roughly follows the way different modularisation phases are combined by Greve, Rennpferdt, & Krause (2020) in commercial product development. However, any conceptual design can be materialised using numerous different construction methods: integral, differential, composite or modular (Pahl *et al.* 2007), and various fabrication technologies: 3D printers, computerized numerical control (CNC) routers, traditional mechanical tools, or a combination of them.

For example, the Wren algorithm designs a modular chassis out of structural frames and transverse elements. Both are composed from smaller CNC flat-cut plywood panels using a differential construction method (Figure 5). A possible alternative could be to construct the frames out of solid timber beams, but this minor alteration would render the whole Wren algorithm useless. More radical alternatives may include the potential to construct the same conceptual building plan out of precast or cast-*in-situ* concrete, or even steel. Each construction method requires a totally different dataset: watertight 3D models, flat-cut profile drawings or conventional 2D drawings, sketches and diagrams. Additionally, each of the construction methods results in different levels of hardware modularity. Hence, the

[9]Image processing: authors, image source: authors.
[10]https://www.planx.uk/

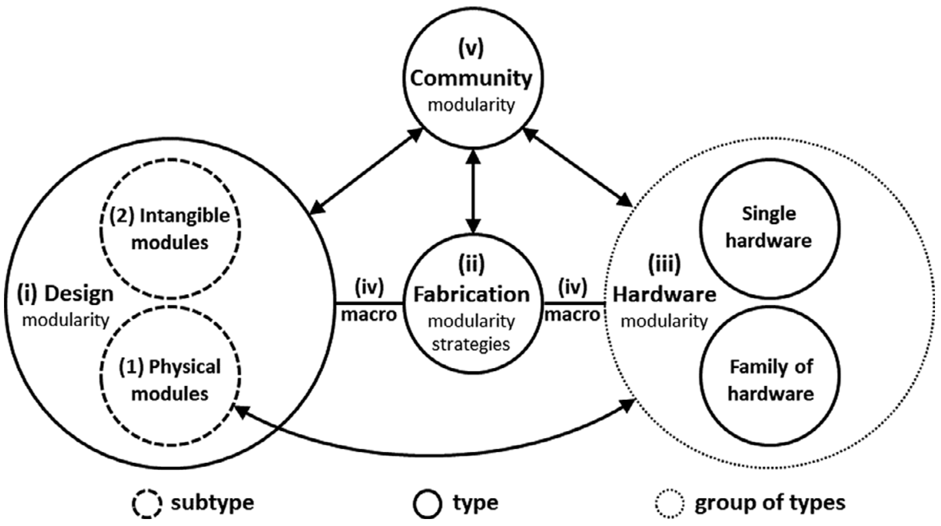


Figure 7. Modularity typology drawn from practice and literature. Arrows express potential interdependence and simple lines express succession.[11]

disentanglement of design from construction and the simplification of the interface between them, whenever possible, may allow the end-user or maker to combine different designs with different construction methods, and vice versa.

Fabrication features a possibility for another modularity type highlighted by the Wren design algorithm. The Wren design algorithm subdivides structural frames into smaller components for fabrication. Those components are profiles flat cut from plywood sheets that come in the standard size of 1220 $\times$ 2440 mm. Thus, the structural chassis is, by default, fabricated using an industrial-grade CNC machine of similar or greater bed size (i.e., greater than 1220 $\times$ 2440 mm). This technological dependency limits the 'Design Global, Manufacture Local' (Kostakis *et al.* 2015) potential of OSH. Nevertheless, the Wren design algorithm allows the user to input the available CNC bed size and plywood length, width, and thickness, and subsequently all fabrication drawings are adjusted automatically. In this example, the fabrication is modular, as the interfaces between fabrication media, stock material and fabrication drawings are simple enough to allow independent adjustments. The modularisation of the fabrication process is important, especially in complex assemblies where a simple modification in one parameter (e.g., panel thickness) may result in a substantial volume of design work. The content and target of fabrication modularity are wholly different than modularity in production processes, because the latter aims to outsource the production of components by mirroring product modularisation.

Based on the literature and the types contributed by the cases explored above, we propose the following OSH modularity typology (Figure 7):

(i) Design modularity, which can be based on:

(1) Physical modularisation. The design of the object is decomposed into the design of separate hardware modules. Physical design modularity is a

[11]Image processing: authors, image source: authors.

prerequisite for hardware modularity. The OpenStructures (2020) project exemplifies this in featuring custom parts that, assembled, synthesise into higher-scale objects, called apps.

(2) Intangible modularisation. The design of the object is decomposed into simpler cognitive tasks that form discrete design modules. Modules are generic blocks of design knowledge that are potentially reusable. No overarching one-to-one correspondence is observed between intangible design and physical modules. Moreover, intangible design modularisation can coexist with physical modularisation as well as with non-modular construction methods. The Wren algorithm and PlanX are instances of intangible design subjected to modularisation. Another example is the structural analysis of a 3D chassis composed of several physical modules (columns and beams).

(ii) Fabrication modularity, which denotes the adaptation of each manufacturing method to varying parameters within the boundaries of the specific method. The function of the Wren algorithm in modularising fabrication data, CNC machine capabilities and stock material dimensions has already been discussed. An example of other manufacturing methods is the customisation of parts and assembly in relation to the maximum printing volume of each available 3D printer. In a broader view, fabrication modularity refers to a design process concerning the final phase of the design: design for fabrication. However, the incentives are different; intangible design modularity aims to 'design global' and fabrication modularity aims to 'manufacture local'.

(iii) Hardware modularity, which is the outcome of design modularity based on physical modules. Bonvoisin *et al.* (2016) completed a systematic literature review of modularisation principles that may be based on material, function, service frequency and reusability. Hardware modularity may refer to a single instance of hardware, or a family of hardware. In the latter case, an ideally highly modular family features modularity in use. In OSH, hardware modularisation remains important for many reasons, but not for outsourcing production tasks.

(iv) Macro-modularity, which is the opposite of the harmonisation of different modularity types in a single project as described by Greve, Rennpferdt, & Krause (2020). Macro-modularity is actuated at the level of the whole OSH ecosystem, based on the simplification or standardisation of interfaces among the different phases of hardware development (e.g., conceptual design and design for fabrication use). A counterexample is the fusion of conceptual design and design for fabrication in the Wren algorithm. A potential example of macro-modularity is the option given to the user or maker to utilise an alternative fabrication method from another OSH housing project in order to construct a WikiHouse-designed house.

(v) Community modularity, which addresses the collaborative structure among contributors, beta testers and users. Community organisation is related to design and hardware modularisation. However, the direction of causality is unclear (Colfer & Baldwin 2016). In contrast with the organisation of the firm, open-source communities are based on voluntary participation, the self-selection of tasks and a lack of top-down authority. In any case, the measure of community modularity cannot separate the distributed from the hierarchical organisation, as both can be modular (Sarkar & Dong 2011). Further analysis of network properties reveals node inheritance (Yu & Ramaswamy

2007) or node centrality (Piccolo, Lehmann, & Maier 2018). Both properties are distinctive features of hierarchical organisations.

We next draw an initial scheme of interdependencies and independencies among the modularity types. Hardware modularity cannot exist without physical design modularity. On the other hand, a design based on intangible modules and fabrication modularity, as described so far, may not necessarily provoke any other type of physical modularity. According to the 'mirroring hypothesis', community modularity is positively related to design and hardware modularity; however, further research is required to provide systematic evidence of the same in OSH.

An observation by Bonvoisin *et al.* (2016) – with greater extensions in OSH – is that design modularity can be either *ex ante* or *ex post. Ex post* modularity approximates the hierarchical division of a whole into parts after the completion of the design. *Ex ante* modularity is bottom-up, more susceptible to change, and closer to a distributed form of organisation. *Ex post* modularity resembles what Raymond (2000) describes as a cathedral structure, and *ex ante* modularity is closer to a bazaar structure. The modular structure of a bazaar is described by Gentile (2013) as architecture of additive functionality. Therefore, modularity type and degree aside, a modularisation strategy may influence modularity magnitude and effectiveness.

5. Quantifying modularity

This section quantifies the types of modularity observed in the selected cases that were introduced in Section 4.

5.1. OpenStructures

OpenStructures is a mature OSH project that features design modularity (based on physical modules) and hardware and community modularity. OpenStructures' network involves 16 designers, 39 apps and 45 unique parts utilised 102 times. The modularity analysis is performed at four different levels (Table 2) of successively enhanced abstraction. Our aim is to capture all possible types of modularity, as well as intermediate conditions that may offer useful meanings.

The first level refers to spatial attachment relations among the parts within a single app: the one made of the greater number of unique parts (Figure 8). The app A.563 was selected from two apps consisting of seven unique parts used several times in each app.

The second level simulates the relation of containment between all apps and parts. When a part is used to assemble an app, those elements are linked (Figure 9). This level is an intermediate-hybrid condition situated between the described types of modularity in OSH, as nodes represent objects of different classes. It is important to note that only custom parts are taken into consideration at all analysis levels, and that ready-made modules are excluded.

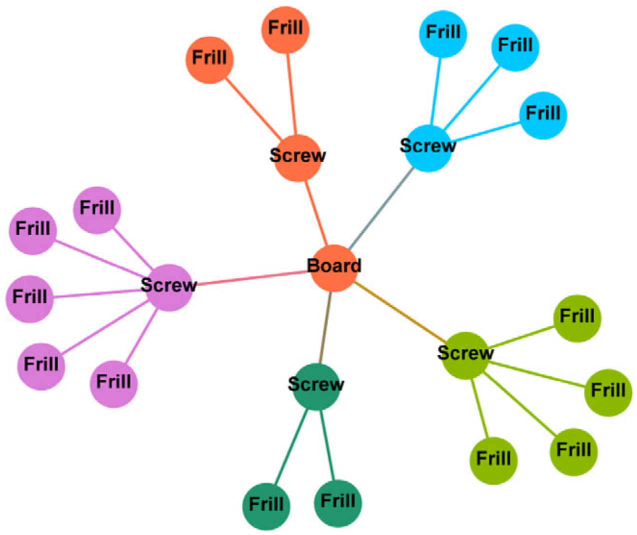
The third level is that of the family of hardware or the representation of weighted relations between apps with one or more common parts (Figure 10).

The last level (Figure 11) outlines the relations in the community between designers; the minimum collaboration act (minimal edge weight) is the indirect and asynchronous contribution of one designer to an app of another designer.

Beyond the community modularity index, an overview of the community is meaningful for meta-analysis. Specifically, the distribution of collaboration 'units'

Table 2. OpenStructures modularity analysis overview

		Modularity analysis (Blondel *et al.* 2008)	
Nodes	Edges	Modularity index [−1,1]	Number of modules
Parts of A.563	Physical assembly linkage between parts of an app	0.596	5
All apps and parts	Drawn between apps and parts when the former contains the latter	0.770	13
All apps	Drawn between apps that share at least one part	0.533	11
Designers	Drawn between designers: one designer sharing his/her part in another designer's app	0.112	6



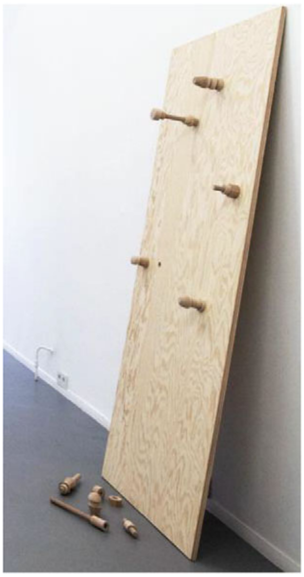


Figure 8. Left: Parts of App A.563 (modularity index: 0.596). An example of design–hardware modularity in a single piece of hardware. Node labels stand for part category. Node colors separate the bigger modules.[12] Right: App A.563. Coat Rack.[13]

[13]Source: https://openstructures.net/apps/a563, license: OS non-commercial 1.0 (downloadable on December 12th, 2019).

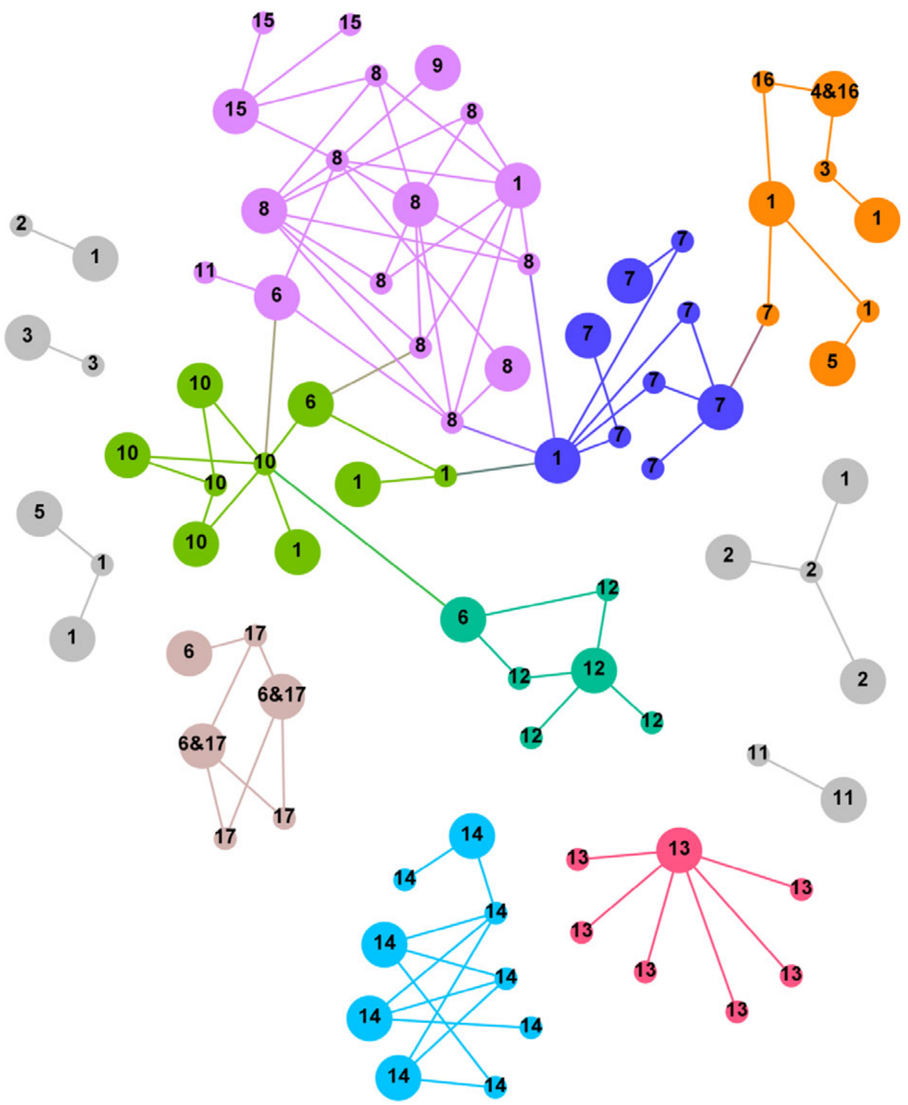


Figure 9. Apps-Parts network (Modularity index: 0.770). The analysis refers to an intermediate-hybrid condition between described types of modularity in open-source hardware (OSH), as nodes represent objects of different classes. Big nodes represent the Apps; small nodes represent the contained Parts. Node labels are unique identifiers of designers' names. Node colors separate the bigger modules.[14]

among designers (Table 3) is a useful, though not absolute, indicator of *ex ante* or *ex post* modularisation strategies. The distribution of Table 3 concerns both inbound and outbound collaborative 'units'. Otherwise, the distribution records both the times that a designer used part(s) designed by others and the times that others used his/her part(s). Any modularisation strategy, whether *ex post* or *ex ante*, is hard to recognise by the examination of a single artifact. However, when a part is used in more apps of different designers, initial as well as subsequent

[14]Data collection, analysis and visualisation: authors, software: Gephi.

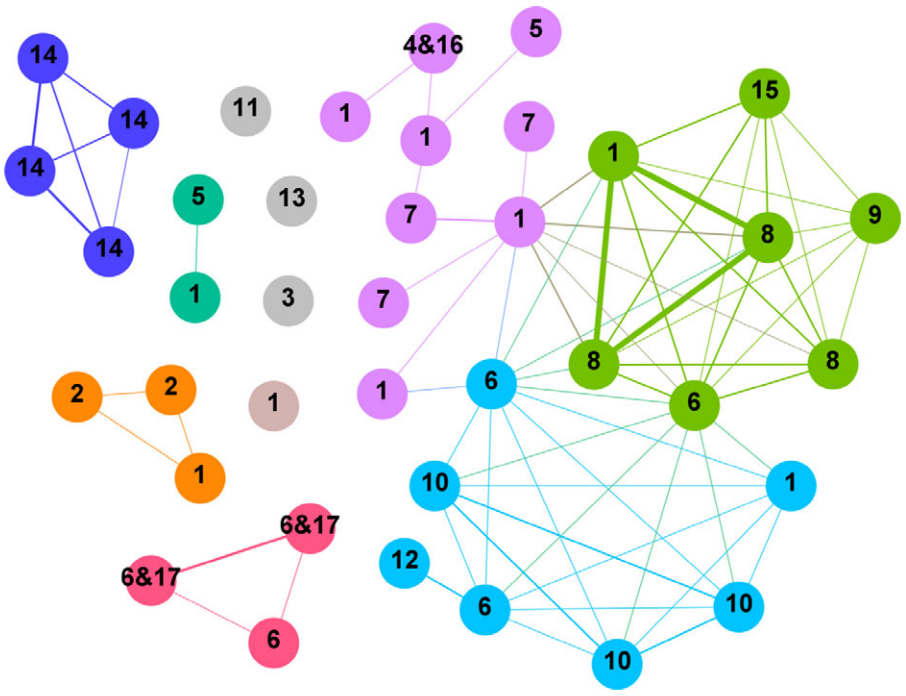


Figure 10. Apps-Apps network (Modularity index: 0.533). An example of family of hardware modularity. Edge weight represents the number (one or more) of parts in common. Node labels are unique identifiers of designers' names. Node colors separate the bigger modules.[15]

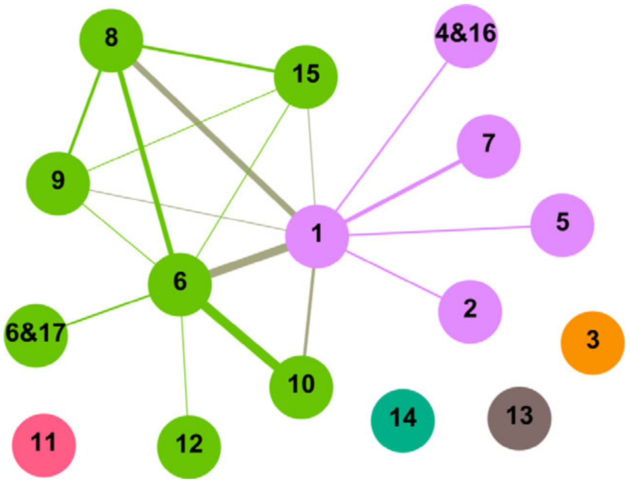
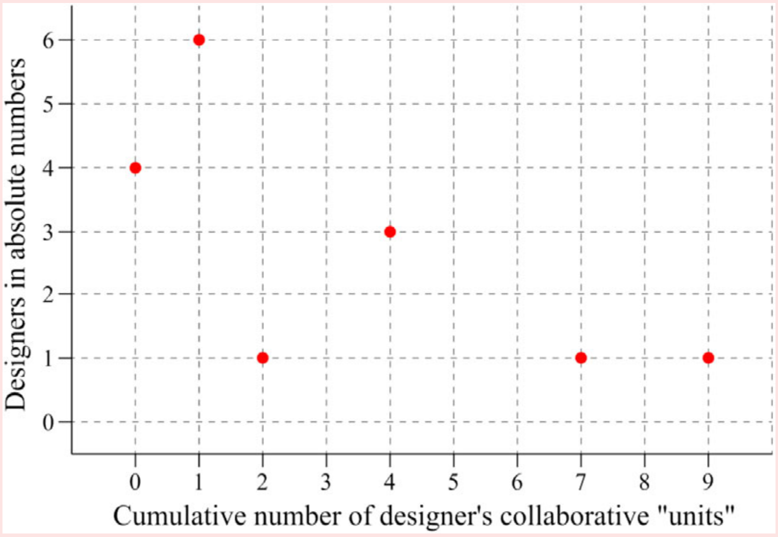


Figure 11. Designers network (Modularity index: 0.112). An example of community modularity. Edge weight represents the number of contributions of one designer to another designer's Apps. Node labels are unique identifiers of designers' names. Node colors separate the bigger modules (communities).[16]

[15]Data collection, analysis and visualisation: authors, software: Gephi.
[16]Data collection, analysis and visualisation: authors, software: Gephi.

Table 3. Distribution of cumulative (inbound and outbound) collaboration 'units' among designers



modularisations are more likely to be *ex ante* strategies. The meaning of the distribution will be discussed further in Section 6.1.2 (ii).

5.2. WikiHouse (Wren)

The Wren algorithm is the core of the WikiHouse project, as well as a characteristic example of intangible design modularity and fabrication modularity. The Wren visual programming script consists of 1086 nodes representing design components, either single or clustered, and 1618 relations (edges) between them. Also, despite the fact that WikiHouse is the result of cooperative efforts in general, according to the GitHub repository[17] and the current authors' personal communication, the Wren script was developed by a single designer-programmer.

The modularity analysis of the Wren algorithm (Table 4 and Figure 12) is performed at the level of design components, marking their explicitly presented interrelation (Figure 13). In terms of simplicity, the script, which was created by the authors to automatically collect the required data, recognises the clustered design components as single-node design components. Clusters are like global functions in programming, capable of being infinitely reused in a programme.

Irrespective of cluster organisation, most design components are grouped by the developer in coloured bounding rectangles (Figure 13) for the purposes of clarity. While this kind of grouping does not imply any functional consequences, groups are indicative of the developer's notion of design modules. To that end, a second level of analysis was performed (Figure 14) to juxtapose the developer's

[17]https://github.com/wikihouseproject/Wren/blob/master/WikiHouse_WREN_(v4.3).gh

Table 4. Wren modularity analysis overview

		Modularity analysis (Blondel *et al.* 2008)	
Nodes	Edges	Modularity index [−1,1]	Number of modules
Design components	Drawn between the output of one component and the input of another	0.763	35

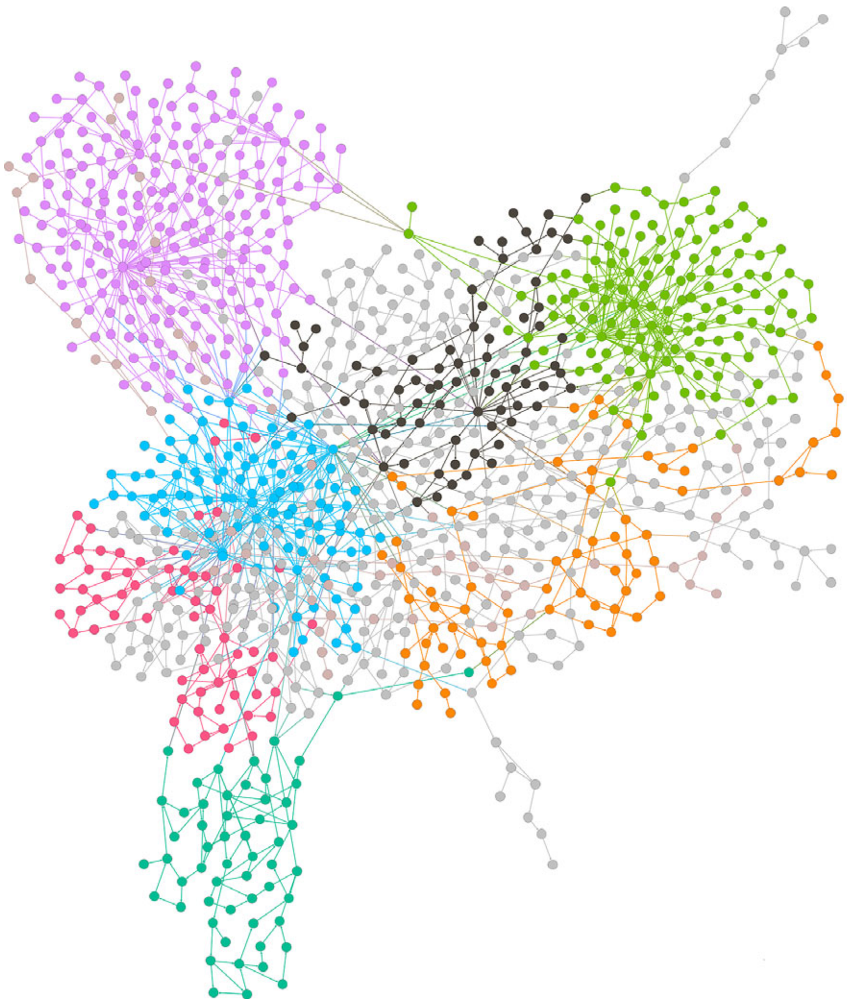
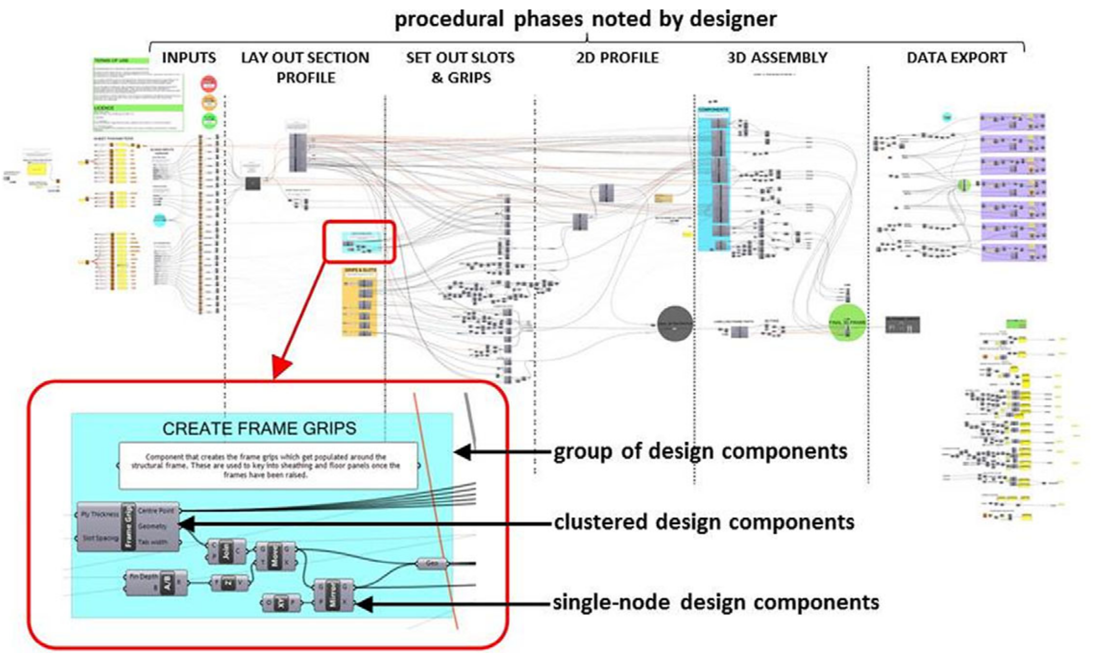


Figure 12. The modules of the Wren design algorithm. Nodes represent design components. Node colors separate the bigger modules.[18]

[18]Data collection, analysis and visualisation: authors, software: Gephi.



procedural phases noted by designer
INPUTS
LAY OUT SECTION PROFILE
SET OUT SLOTS & GRIPS
2D PROFILE
3D ASSEMBLY
DATA EXPORT
CREATE FRAME GRIPS
Component that creates the frame grips which get populated around the structural frame. These are used to key into sheathing and floor panels once the frames have been raised.
group of design components
clustered design components
single-node design components

Figure 13. The Wren algorithm in its native visual programming environment (Grasshopper software). The magnified call out indicates single components, clustered components and groups of components. There are more actual connections between design components than the wired connections displayed.[19]

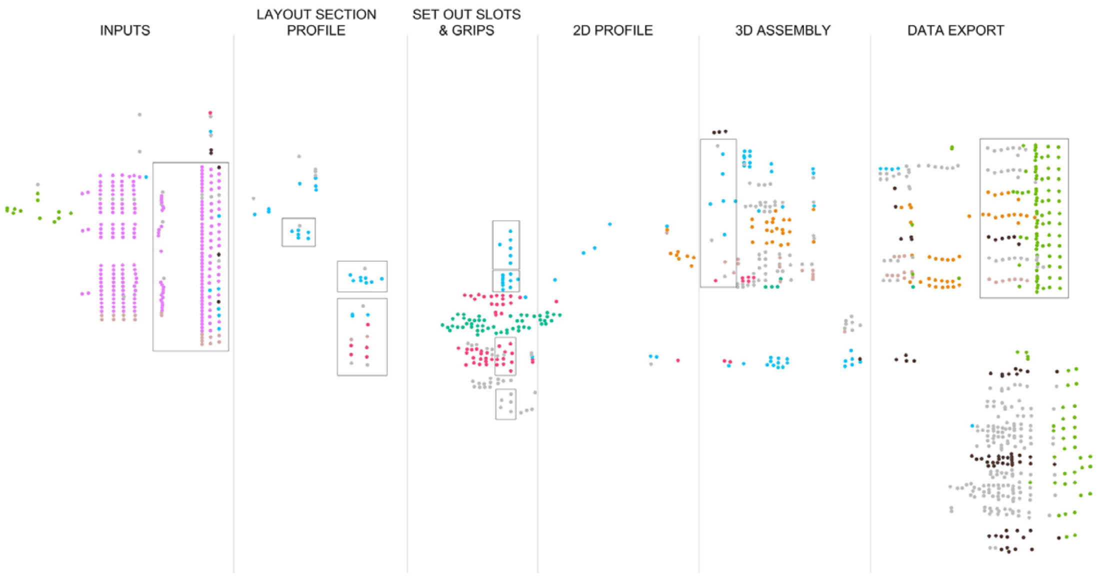


Figure 14. Groups of design components (rectangles) and design phases (labels) as set by the Wren developer to order 1086 design components, compared with the modules (colour) as recognised by the modularity algorithm in Figure 12.[20]

[19]Source: https://github.com/wikihouseproject/Wren/blob/master/WikiHouse_WREN_(v4.3).gh, license: MPL 2.0 (downloadable on June 1st, 2020), edited by the authors.

[20]Data collection, analysis and visualisation: authors, software: Gephi and Grasshopper.

intended modules (Figure 13) with those detected from the algorithmic modularity analysis (Figure 12). While no one-to-one coincidence exists between intended–detected design modules, a certain degree of coincidence above randomness is observed at the higher level (Figure 14). Also, the detected modules (except cyan-coloured modules) show a similar coincidence with the procedural design phases noted by the designer-programmer.

6. Discussion

This section discusses the results presented in Section 5 under the framework described in Section 2.

6.1. Interpretation of the results

The presented results delineate a wide range of modularity levels: from near random connectivity (0.112) to fairly modular structures (0.763). According to Newman & Girvan (2004), most real-world networks lie between 0.3 and 0.7, while higher values are rare. Therefore, the value of 0.763 corresponds in absolute terms to a highly modular structure. In general, the performed modularity analysis offers two different kinds of metrics: the modularity index and the number of modules.

The first metric – described in Section 3 – is not directly affected by the size of the sample but only by the qualitative attributes of connectivity. An example regarding the scale-free nature of the modularity index is the relative closeness between the levels of a single app (0.596) and of Wren (0.763). The former consists of 22 nodes (parts) and the latter of 1086 nodes (design components). Thus, different modularity types of the same project can be compared regarding possible coincidence (the 'mirroring hypothesis'). Moreover, alternatives (intangible versus physical) of design modularity, across different projects, can be weighed in regard to their magnitude.

The number of modules, a secondary metric of modularity analysis, is not scale-free. Additionally, the number of modules is important as an indicator of the collaborative potential of each OSH project. According to Benkler (2006), the greater the number of modules, the greater the potential to attract many contributors from the community. As outlined in Section 1, the number of modules is also connected to parallel work and complexity management (Baldwin & Clark 2002), as well as coordination minimisation (Collopy, Adar, & Papalambros 2020). However, the number of modules itself is not an adequate metric for assessing modularity.

Regarding the comparability of modularity types between the cases, two points should be noted beyond the absolute nature of the modularity index already analysed. First, the Wren algorithm concerns both the design of a single wikihouse and a family of wikihouses, proportionate to the OpenStructures family. The difference lies in the extent and variation of the wikihouse family, which, based on each user input, is almost infinite. Consequently, the hardware modularity levels of the output of the Wren algorithm, being either a single wikihouse or a family of wikihouses, cannot be evaluated without significant assumptions. Second, the size of a wikihouse, measured by the number of unique hardware modules, is similar to the size of A.563 and other OpenStructures apps. Analytically, each wikihouse features identical but generally custom frames in one direction and transverse connectors in the other, with cladding and floor modules between the frames differentiated only in the case of window or door. To summarise, most wikihouses consist of approximately six modules in a repeated arrangement. The difference is that the Wren algorithm includes extra complexity in its intangible design functions and the subdivision of physical modules into smaller intra-module components (Figure 5).

6.1.1. Inter-case design modularity levels

Design modularity is the only common modularity type between the cases. The comparative analysis provides preliminary evidence that intangible design modularisation is likely to produce more modules than physical design modularisation. A comparison between Wren and A.563 provides an example of this: both were designed by a single designer and feature 35 and 5 modules, respectively. Certainly, further research is required to compare more pairs of similar size and complexity before a universal principle can be recognised.

However, intangible design modularisation potentially features increasingly more modularisation levels compared to the physical modularisation of hardware. Whether the hardware modularisation strategy is based on material, function, service frequency or reusability (Bonvoisin *et al.* 2016), a different but finite upper limit of modules is anticipated. Conversely, in the case of intangible design modularisation, the upper limit is potentially more flexible due to the intangible nature of the design modules. For example, the designer of a wikihouse can keep adding almost infinite intangible design functions, embedded in a fixed number of physical modules, whereas s/he faces an upper limit when adding chassis physical modules for a wikihouse of a specific size.

The comparison between the modularity indices enhances the previously described outcome in a number of ways. First, the Wren algorithm features greater design modularity than the A.563 (0.763 versus 0.596). Second, even when comparing the Wren design algorithm with the entire OpenStructures family of hardware, featuring 102 parts, intangible design modularisation is prevalent (0.763 versus 0.533). These comparisons emphasise the tendency that intangible compared to physical design modularisation strategies are more effective in yielding more modules and qualitatively more modular structures. Also, the relative coincidence among Wren-detected modules and the horizontal (phases) and vertical (groups) ordering of 1086 design components by the developer is surprising (Figure 14). Even in complex intangible modularisations, the developer can safely predict actual modules. Gopsill, Snider, & Hicks (2019) have revealed a significant deviation between intended and designed modular architectures within CAD environments.

6.1.2. Intra-case modularity levels and the 'mirroring hypothesis'

Following the comparative evaluation of these cases, we focus on the interrelation of modularity types in each project independently. Interestingly, for OpenStructures it may be observed that community modularity is unusually low (0.112), very close to random connectedness. Community modularity is lower than hardware modularity, either examined as a single case (0.596) or as a family of hardware (0.533). Unfortunately, the same comparisons cannot be repeated for Wren. While WikiHouse consists of a small team based in London, the Wren algorithm was created by a single developer. However, in this case the fact that a single developer produced a highly modular design file (0.763) is surprising.

Consequently, each case features an inverse relation between the levels of community modularity and the produced modularity either being design or hardware. Not even a preliminary pattern of correlation among modularity types was detected. On the contrary, the juxtaposition of the available modularity levels in each OSH project reveals an interesting but uncommon contradiction relating to the analogy observed in the software realm, also known as the 'mirroring hypothesis' (MacCormack, Baldwin, & Rusnak 2012).

Before expanding on the possible reasoning for the deviation, we note that MacCormack, Baldwin, & Rusnak (2012) have not tested exactly what is known as the 'mirroring hypothesis' despite a common misunderstanding. First, MacCormack, Baldwin, & Rusnak (2012) axiomatically supposed that firms and OSS communities are tightly and loosely coupled organisations, respectively, no matter what the actual or comparative extent of the property is in each case. Next, the researchers compared proprietary code with OSS in terms of their modifiability, proving that OSS is more 'modular' by a factor of six. Therefore, no quantified modularity mirroring exists between the examined software and the community. It is likely that the lack of a unified qualitative and quantitative definition of modularity, as well as the limitations of utilised metrics – outlined in Section 3 – has resulted in the vagueness and ambiguity described above.

Regarding our cases, possible explanations for the deviation among modularity types include the following:

(i) The 'mirroring hypothesis' is inaccurate near extreme values and boundary conditions, as in the case of a single designer or developer.

(ii) *Ex post* modularisation strategies may result in modular hardware or families of hardware, yet also monolithic collaboration networks. Top-down modularised hardware may be highly modular, but such modules are less useful (closed) in another designer's hardware. Consider the example of the cathedral and the bazaar: one can approximately build both out of cellular Lego bricks, but one can build only the cathedral out of pinnacles, triforiums and buttresses. The potential interoperability of modules in practice is not predicted or captured by the modularity metrics. In OpenStructures, the assumption of *ex post* modularisation strategies is supported to some extent by the analysis performed. A considerable portion of the designers (4/16) have not collaborated with anyone, and few (6/16) have collaborated with more than one (Figure 11 and Table 3). At the level of parts, almost half of the parts (20/45) have not been reused in another designer's app (Figure 9), and very few have been reused in three or more designers' apps (Figure 10).

(iii) Asynchronous and distance-based collaboration as well as the self-selection of tasks may be critical for community modularity to emerge. According to the OpenStructures website (2020), the design of a significant portion of hardware had been commissioned by museums, galleries or even the OpenStructures studio itself, revealing a top-down organisation of tasks. Also, the monolithic and well-connected core of designers (Figure 11) is located mainly in Brussels (the base of the OpenStructures studio) and nearby cities. In contrast, designers from Ramallah or Taipei are less well connected to the central core.

(iv) Beyond all other parameters, design modularity may also depend on the design tool. Visual programming – in the case of Wren – may provoke more modular design than conventional CAD or other design tools. Beeker L, Pringle, & Pearce (2018) and Oberloier & Pearce (2018) report that the parametric nature of the code leads to easily customisable designs. For Wren, the integral distribution of detected modules in procedural phases (Figure 14) highlights that algorithmic design, due to its inherently procedural nature, is closer to an inherently modular structure. In contrast, sketching is a highly nonmodular knowledge structuration process (Brun, Le Masson, & Weil 2016). Bonvoisin *et al.* (2018) have reported significantly lower distributed collaboration based on CAD files compared with other design documentation in OSH. Even in software, the programming paradigm (procedural or object-oriented language) affects code structure (Baldwin, MacCormack, & Rusnak 2014). The Million Penguins collaborative writing project forms a counter-example (Shirky 2005), indicating that the open-source model cannot be applied to any creative realm without first transforming each conventional workflow (Troxler 2019).

6.2. Limitations

Most limitations in the present article stem from the type of research adopted. The case studies analysed here offer a plethora of useful and detailed insights, but results cannot be safely generalised without further systematic investigation. As the OSH realm is still immature, the selection of the sample is crucial for any future systematic research.

Additionally, the unconditional positive impact of modularity levels on OSH development was an axiomatic assumption in this article. In proprietary hardware, optimal modularity is considered preferable to maximum modularity. In contrast with OSH and OSS, in the proprietary realm modularity is a tool to manage limited resources efficiently (Colfer & Baldwin 2016). Further research is required to investigate the potential drawbacks of high modularity levels in OSH development.

A limitation concerning OpenStructures is that the app design files were unavailable. The only available design files were the CAD files for single parts. Therefore, at the lowest analysis level, that of the selected single app A.563, the detection of dependencies among parts was based on spatial attachment relations interpreted from photographic documentation. The relative simplicity of the artifact and the absence of other dependencies (apart from spatial) among modules ensured the accuracy of data collection and analysis. Even if the app design files were available, dependencies between CAD entities (lines, arcs, solids or other) are rarely explicit. Regarding the levels of family of hardware and community analysis, data collection was restricted to the relations of containment between apps and parts as explicitly recorded on the project website (OpenStructures 2019). Explicit relations may not cover the full range of engagement among designers, as described by Kohtala, Hyysalo, & Whalen (2019).

A less important limitation concerns the recognition of clustered design components as single-node design components in the Wren case. The simplification of the script that collects connectivity data was rated as more important than accuracy loss. As there are 9 clustered design components in a total of 1086 design components (translating to less than 1%), the accuracy loss is acceptably low.

7. Conclusions

The application of an open-source approach to hardware inaugurates a new research field regarding modularity. This article critically discussed widely used definitions of modularity to argue that proprietary hardware and OSH are phenomena with distinct motives and theoretical frameworks. First, we provided a qualitative and quantitative framework to assess modularity in OSH, drawing mainly from the literature of network science. Second, we presented a tentative classification of modularity types, attempting to address the full range of OSH development. Interestingly, intangible design, fabrication and macro modularity types are unique in the OSH realm and are presented and defined herein, to our knowledge, for the first time. Such a framework and classification may be integral for future systematic research in OSH modularity. Moreover, our quantitative analysis introduced preliminary evidence to argue that intangible design modularisation is more effective than exclusively physical modularisation strategies. Our final remark is that the 'mirroring hypothesis' is not a universal pattern in OSH. The selection of design tools as well as the modularisation strategy (*ex ante* or *ex post*) may affect the levels of modularity types.

It would be interesting to explore whether other network structural properties such as hierarchy counteract or enhance the impact of modularity. Existing approaches in literature (Baldwin, MacCormack, & Rusnak 2014), which correlate modularity with hierarchy, have implemented arbitrary thresholds in order to classify structures. Therefore, the quantification methodology and the meaning of other structural metrics (hierarchy, commonality and combinability, among others) in each OSH modularity type is a promising research topic *per se*. Another promising direction is the mapping of modularity levels along the phases of the development of successful OSH projects. As even the larger projects begin locally with a small team, the recognition of a common pattern may reveal the stages by which quantitative change provokes or requires qualitative reform. By mapping the different types of modularity in OSH, we hope that this paper may prompt a discussion into a wide but less-explored subset in OSH and beyond.

Glossary

CAD	Computer Aided Design
CNC	Computerized Numerical Control
DSM	Design Structure Matrix
OSH	Open-Source Hardware
OSS	Open-Source Software

Acknowledgements

Network analyses and visualisations were generated with the free and open-source Gephi software. We thank three anonymous reviewers whose comments/suggestions helped improve and clarify this manuscript. V.K. acknowledges funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant No. 802512). Authors acknowledge financial support from the Hellenic Academic Libraries Link (HEAL) regarding article processing charge (APC).

References

Albers, A., **Bursac, N.**, **Scherer, H.**, **Birk, C.**, **Powelske, J.** & **Muschik, S.** 2019 Model-based systems engineering in modular design. *Design Science* **5** (e17), 1–33; doi:10.1017/dsj.2019.15.

Asikoglu, O. 2012 *A new method for evaluating design dependencies in product architectures* (PhD Thesis). Department of Industrial and Manufacturing Engineering, The Pennsylvania State University.

Baldwin, C. Y. & **Clark, K. B.** 2000 *Design Rules, Volume 1: The Power of Modularity*. MIT Press.

Baldwin, C. Y. & **Clark, K. B.** 2002 The option value of modularity in design: An example from Design rules, volume 1: The power of modularity. *Harvard NOM Working Paper No. 02–13, Harvard Business School Working Paper No. 02–078*, pp. 1–15; doi:10.2139/ssrn.312404.

Baldwin, C. Y. & **Clark, K. B.** 2003 Managing in an age of modularity. In *Managing in the Modular Age: Architectures, Networks, and Organizations* (**ed**. R. Garud, A. Kumaraswamy & R. Langlois), pp. 84–93. Blackwell Publishers.

Baldwin, C. Y. & **Clark, K. B.** 2006 The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science* **52** (7), 1116–1127; doi:10.1287/mnsc.1060.0546.

Baldwin, C. Y., **MacCormack, A.** & **Rusnak, J.** 2014 Hidden structure: Using network methods to map system architecture. *Research Policy* **43** (8), 1381–1397; doi:10.1016/j.respol.2014.05.004.

Balka, K., **Raasch, C.** & **Herstatt, C.** 2009 Open source enters the world of atoms: A statistical analysis of open design. *First Monday* **14** (11); doi:10.5210/fm.v14i11.2670.

Ball, Z. & **Lewis, K.** 2018 Observing network characteristics in mass collaboration design projects. *Design Science* **4** (e4), 1–31; doi:10.1017/dsj.2017.26.

Beeker L, Y., **Pringle, A.** & **Pearce, J.** 2018 Open-source parametric 3-d printed slot die system for thin film semiconductor processing. *Additive Manufacturing* **20**, 90–100; doi:10.1016/j.addma.2017.12.004.

Benkler, Y. 2006 *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.

Blondel, V. D., **Guillaume, J.-L.**, **Lambiotte, R.** & **Lefebvre, E.** 2008 Fast unfolding of communities in large networks. *Statistical Mechanics Theory and Experiment* **2008** (10), 1–12; doi:10.1088/1742-5468/2008/10/P10008.

Boisseau, É., **Omhover, J.-F.** & **Bouchard, C.** 2018 Open-design: A state of the art review. *Design Science* **4** (e3), 1–44; doi:10.1017/dsj.2017.25.

Bonarini, A., **Matteucci, M.**, **Migliavacca, M.** & **Rizzi, D.** 2014 R2P: An open source hardware and software modular approach to robot prototyping. *Robotics and Autonomous Systems* **62** (7), 1073–1084; doi:10.1016/j.robot.2013.08.009.

Bonvoisin, J., **Buchert, T.**, **Preidel, M.** & **Stark, R.** 2018 How participative is open source hardware? Insights from online repository mining. *Design Science* **4** (e19), 1–31; doi:10.1017/dsj.2018.15.

Bonvoisin, J., **Halstenberg, F.**, **Buchert, T.** & **Stark, R.** 2016 A systematic literature review on modular product design. *Engineering Design* **27** (7), 488–514; doi:10.1080/09544828.2016.1166482.

Bonvoisin, J. & **Mies, R.** 2018 Measuring openness in open source hardware with the open-o-meter. *Procedia CIRP* **78**, 388–393; doi:10.1016/j.procir.2018.08.306.

Bonvoisin, J., **Molloy, J.**, **Haeuer, M.** & **Wenzel, T.** 2020 Standardisation of practices in open source hardware. *Open Hardware* **4** (1), 1–11; doi:10.5334/joh.22.

Boujut, J.-F. 2019 The emerging phenomenon of open source hardware design. In *JAIST World Conference (JWC2019) Advanced Design Creativity Track*. Kanasawa: Japan.

Braha, D. & **Bar-Yam, Y.** 2007 The statistical mechanics of complex product development: Empirical and analytical results. *Management Science* **53** (7), 1127–1145; doi:10.1287/mnsc.1060.0617.

Brun, J., **Le Masson, P.** & **Weil, B.** 2016 Designing with sketches: The generative effects of knowledge preordering. *Design Science* **2** (e13), 1–26; doi:10.1017/dsj.2016.13.

Brusoni, S. & **Prencipe, A.** 2001 Unpacking the black box of modularity: Technologies, products and organizations. *Industrial and Corporate Change* **10** (1), 179–205; doi:10.1093/icc/10.1.179.

Cabigiosu, A., **Zirpoli, F.** & **Camuffo, A.** 2013 Modularity, interfaces definition and the integration of external sources of innovation in the automotive industry. *Research Policy* **42** (3), 662–675; doi:10.1016/j.respol.2012.09.002.

Campagnolo, D. & **Camuffo, A.** 2010 The concept of modularity in management studies: A literature review. *Management Review* **12** (3), 259–283; doi:10.1111/j.1468-2370.2009.00260.x.

Colfer, L. J. & **Baldwin, C. Y.** 2016 The mirroring hypothesis: Theory, evidence, and exceptions. *Industrial and Corporate Change* **25** (5), 709–738; doi:10.1093/icc/dtw027.

Collopy, A. X., **Adar, E.** & **Papalambros, P. Y.** 2020 On the use of coordination strategies in complex engineered system design projects. *Design Science* **6** (e32), 1–43; doi:10.1017/dsj.2020.29.

Emanuel, A. W. R., **Wardoyo, R.** & **Istiyanto, J. E.** 2011 Modularity index metrics for java-based open source software projects. *Advanced Computer Science and Applications (IJACSA)* **2** (11), 52–58; doi:10.14569/IJACSA.2011.021109.

Eppinger, S. D., **Whitney, D. E.**, **Smith, R. P.** & **Gebala, D. A.** 1994 A model-based method for organizing tasks in product development. *Research in Engineering Design* **6** (1), 1–13; doi:10.1007/BF01588087.

Erens, F. & **Verhulst, K.** 1997 Architectures for product families. *Computers in Industry* **33** (2–3), 165–178; doi:10.1016/S0166-3615(97)00022-5.

Erixon, G., **Von Yxkull, A.** & **Arnström, A.** 1996 Modularity – The basis for product and factory reengineering. *CIRP Annals* **45** (1), 1–6; doi:10.1016/S0007-8506(07)63005-4.

Fellini, R., **Kokkolaras, M.**, **Michelena, N.**, **Papalambros, P.**, **Perez-Duarte, A.**, **Saitou, K.** & **Fenyes, P.** 2004 A sensitivity-based commonality strategy for family products of mild variation, with application to automotive body structures. *Structural and Multidisciplinary Optimization* **27**, 89–96; doi:10.1007/s00158-003-0356-x.

Fixson, S. K. 2007 Modularity and commonality research: Past developments and future opportunities. *Concurrent Engineering* **15** (2), 85–111; doi:10.1177/1063293X07078935.

Fixson, S. K. & **Park, J.-K.** 2008 The power of integrality: Linkages between product architecture, innovation, and industry structure. *Research Policy* **37** (8), 1296–1316; doi:10.1016/j.respol.2008.04.026.

Frigant, V. & **Talbot, D.** 2005 Technological determinism and modularity: Lessons from a comparison between aircraft and auto industries in Europe. *Industry and Innovation* **12**, 337–355; doi:10.1080/13662710500195934.

Gavras, K. 2019 Open source beyond software: Re-invent open design on the common's ground. *Peer Production* **13**, http://peerproduction.net/editsuite/issues/issue-13-open/peer-reviewed-papers/open-source-beyond-software/.

Gentile, P. D. 2013 Theory of modularity, a hypothesis. *Procedia Computer Science* **20**, 203–209; doi:10.1016/j.procs.2013.09.262.

Gopsill, J. A., **Snider, C.** & **Hicks, B. J.** 2019 The emergent structures in digital engineering work: What can we learn from dynamic DSMs of near-identical systems design projects? *Design Science* **5** (e28), 1–29; doi:10.1017/dsj.2019.20.

Greve, E., **Rennpferdt, C.** & **Krause, D.** 2020 Harmonizing cross-departmental perspectives on modular product families. *Procedia CIRP* **91**, 452–457; doi:10.1016/j.procir.2020.02.198.

Guo, F. & **Gershenson, J. K.** 2003 Comparison of modular measurement methods based on consistency analysis and sensitivity analysis. In *Proceedings of the ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. **3b**: 15th International Conference on Design Theory and Methodology,* Chicago, IL, USA, September 2–6, pp. 393–401. ASME; doi:10.1115/DETC2003/DTM-48634.

Guo, F. & **Gershenson, J. K.** 2004 A comparison of modular product design methods based on improvement and iteration. In *Proceedings of the ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. **3a**: 16th International Conference on Design Theory and Methodology,* Salt Lake City, UT, USA, September 28–October 2, pp. 261–269. ASME; doi:10.1115/DETC2004-57396.

Hackl, J., **Krause, D.**, **Otto, K.**, **Windheim, M.**, **Moon, S. K.**, **Bursac, N.** & **Lachmayer, R.** 2019 Impact of modularity decisions on a firm's economic objectives. *Mechanical Design* **142** (4), 403–414; doi:10.1115/1.4044914.

Halman, J. I. M., **Voordijk, J. T.** & **Reymen, I. M. M.** 2008 Modular approaches in Dutch house building: An exploratory survey. *Housing Studies* **23** (5), 781–799; doi:10.1080/02673030802293208.

Hölttä-Otto, K., **Chiriac, N. A.**, **Lysy, D.** & **Suh, E. S.** 2012 Comparative analysis of coupling modularity metrics. *Engineering Design* **23** (10–11), 790–806; doi:10.1080/09544828.2012.701728.

Hölttä-Otto, K. & **de Weck, O.** 2007 Degree of modularity in engineering systems and products with technical and business constraints. *Concurrent Engineering* **15** (2), 113–126; doi:10.1177/1063293X07078931.

Huang, C.-C. & **Kusiak, A.** 1998 Modularity in design of products and systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans* **28** (1), 66–77; doi:10.1109/3468.650323.

Johnson, T. H. & **Broms, A.** 2000 *Profit beyond Measure*. The Free Press.

Jung, S. & **Simpson, T. W.** 2016 New modularity indices for modularity assessment and clustering of product architecture. *Engineering Design* **28** (1), 1–22; doi:10.1080/09544828.2016.1252835.

Kohtala, C., **Hyysalo, S.** & **Whalen, J.** 2019 A taxonomy of users' active design engagement in the 21st century. *Design Studies* **67**, 27–54; doi:10.1016/j.destud.2019.11.008.

Koren, Y., **Hu, S. J.**, **Gu, P.** & **Shpitalni, M.** 2013 Open-architecture products. *CIRP Annals* **62** (2), 719–729; doi:10.1016/j.cirp.2013.06.001.

Kostakis, V. 2019 How to reap the benefits of the "digital revolution"? Modularity and the commons. *Halduskultuur: The Estonian Journal of Administrative Culture and Digital Governance* **20** (1), 4–19; doi:10.32994/hk.v20i1.228.

Kostakis, V., **Niaros, V.**, **Dafermos, G.** & **Bauwens, M.** 2015 Design global, manufacture local: Exploring the contours of an emerging productive model. *Futures* **73**, 126–135; doi:10.1016/j.futures.2015.09.001.

Kostakis, V. & **Papachristou, M.** 2014 Commons-based peer production and digital fabrication: The case of a reprap-based, lego-built 3d printing-milling machine. *Telematics and Informatics* **31** (3), 434–443; doi:10.1016/j.tele.2013.09.006.

Krause, D., **Beckmann, G.**, **Eilmus, S.**, **Gebhardt, N.**, **Jonas H.** & **Rettberg R.** 2014 Integrated development of modular product families: A methods toolkit. In *Advances in Product Family and Product Platform Design* (**ed.** T. Simpson, J. Jiao, Z. Siddique & K. Hölttä-Otto), pp. 245–269. Springer; doi:10.1007/978-1-4614-7937-6_10.

Li, Z., **Seering, W.**, **Ramos, J. D.**, **Yang, M.** & **Robert, W. D.** 2017 Why open source? Exploring the motivations of using an open model for hardware. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* **1**: *37th Computers and Information in Engineering Conference*, Cleveland, OH, USA, August 6–9. ASME; doi:10.1115/DETC2017-68195.

MacCormack, A., **Baldwin, C. Y.** & **Rusnak, J.** 2012 Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis. *Research Policy* **41**, 1309–1324; doi:10.1016/j.respol.2012.04.011.

MacCormack, A., **Rusnak, J.** & **Baldwin, C. Y.** 2006 Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science* **52** (7), 1015–1030; doi:10.1287/mnsc.1060.0552.

Maier, J. F., **Eckert, C. M.** & **Clarkson, J. P.** 2017 Model granularity in engineering design – Concepts and framework. *Design Science* **3** (e1), 1–29; doi:10.1017/dsj.2016.16.

Milev, R., **Muegge, S.** & **Weiss, M.** 2009 Design evolution of an open source project using an improved modularity metric. In *Open Source Ecosystems: Diverse Communities Interacting* (**ed.** C. Boldyreff, K. Crowston, B. Lundell & A.I. Wasserman), OSS 2009, IFIP Advances in Information and Communication Technology, **299**, pp. 20–33. Springer; doi:10.1007/978-3-642-02032-2_4.

Newcomb, P. J., **Bras, B.** & **Rosen, D. W.** 1998 Implications of modularity on product design for the life cycle. *Mechanical Design* **120** (3), 483–490; doi:10.1115/1.2829177.

Newman, M. E. J. 2004 Analysis of weighted networks. *Physical Review E* **70** (5); doi: 10.1103/PhysRevE.70.056131.

Newman, M. E. J. & **Girvan, M.** 2004 Finding and evaluating community structure in networks. *Physical Review E* **69** (2); doi:10.1103/PhysRevE.69.026113.

Oberloier, S. & **Pearce, J.** 2018 General design procedure for free and open-source hardware for scientific equipment. *Designs* **2** (1), 1–15; doi:10.3390/designs2010002.

Open Source Hardware Association 2020 Open Source Hardware (OSHW) Statement of Principles 1.0, online document (downloadable on December 10th 2020) http://www.oshwa.org/definition/.

OpenStructures 2019 Online document (downloadable on December 10th 2020) https://web.archive.org/web/20190526015049/http://beta.openstructures.net/pages/2, archived: 26 May 2019.

OpenStructures 2020 Online document (downloadable on December 10th 2020) http://openstructures.net/.

Pahl, G., **Beitz, W.**, **Feldhusen, J.** & **Grote, K.-H.** 2007 *Engineering Design: A Systematic Approach*, 3rd edn. Springer.

Pandremenos, J., **Paralikas, J.**, **Salonitis, K.** & **Chryssolouris, G.** 2008 Modularity concepts for the automotive industry: A critical review. *CIRP Manufacturing Science and Technology* **1** (3), 148–152; doi:10.1016/j.cirpj.2008.09.012.

Papalambros, P. Y. 2015 Design science: Why, what and how. *Design Science* **1** (e1), 1–38; doi:10.1017/dsj.2015.1.

Paparistodimou, G., **Duffy, A.**, **Whitfield, R. I.**, **Knight, P.** & **Robb, M.** 2020 A network tool to analyse and improve robustness of system architectures. *Design Science* **6** (e8), 1–40; doi:10.1017/dsj.2020.6.

Parnas, D. L. 1972 On the criteria to be used in decomposing systems into modules. *Communications of the ACM* **15** (12), 1053–1058; doi:10.1145/361598.361623.

Pearce, J. M. 2015 Quantifying the value of open source hardware development. *Modern Economy* **6**, 1–11; doi:10.4236/me.2015.61001.

Persson, M. & **Åhlström, P.** 2006 Managerial issues in modularising complex products. *Technovation* **26** (11), 1201–1209; doi:10.1016/j.technovation.2005.09.020.

Piccolo, S. A., **Lehmann, S.** & **Maier, A.** 2018 Design process robustness: A bipartite network analysis reveals the central importance of people. *Design Science* **4** (e1), 1–29; doi:10.1017/dsj.2017.32.

Priavolou, C. & **Niaros, V.** 2019 Assessing the openness and conviviality of open source technology: The case of the wikihouse. *Sustainability* **11** (17), 1–16; doi:10.3390/su11174746.

Raasch, C., **Herstatt, C.** & **Balka, K.** 2009 On the open design of tangible goods. *R&D Management* **39** (4), 382–393; doi:10.1111/j.1467-9310.2009.00567.x.

Raymond, E. S. 2000 The cathedral and the bazaar, online document (downloadable on December 10th 2020) http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Salvador, F. 2007 Toward a product system modularity construct: Literature review and reconceptualization. *IEEE Transactions on Engineering Management* **54** (2), 219–240; doi:10.1109/TEM.2007.893996.

Sanchez, R. & **Mahoney, J. T.** 1996 Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management* **17** (S2), 63–76; doi:10.1002/smj.4250171107.

Sarkar, S. & **Dong, A.** 2011 Characterizing modularity, hierarchy and module interfacing in complex design systems. In *Proceedings of the ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* **9**: *23rd International Conference on Design Theory and Methodology; 16th Design for Manufacturing and the Life Cycle Conference*, Washington, DC, USA, August 28–31, pp. 375–384. ASME; doi:10.1115/DETC2011-47992.

Sharman, D., **Yassine, A.** & **Carlile, P.** 2002 Characterizing modular architectures. In *Proceedings of the ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* **4**: *14th International Conference on Design Theory and Methodology, Integrated Systems Design, and Engineering Design and Culture*, Montreal, Quebec, Canada, September 29–October 2, pp. 265–278. ASME; doi:10.1115/DETC2002/DTM-34024.

Shirky, C. 2005 Epilogue: Open source outside the domain of software. In *Perspectives on Free and Open Source Software* (**ed.** J. Feller, B. Fitzgerald, S. A. Hissam & K. R. Lakhani), pp. 483–488. The MIT Press.

Simon, H. A. 1962 The architecture of complexity. *Proceedings of the American Philosophical Society* **106** (6), 467–482.

Sosa, M., **Eppinger, S.** & **Rowles, C.** 2003 Identifying modular and integrative systems and their impact on design team interactions. *Mechanical Design* **125** (2), 240–252; doi:10.1115/1.1564074.

Sosa, M., **Eppinger, S.** & **Rowles, C.** 2004 The misalignment of product architecture and organizational structure in complex product development. *Management Science* **50** (12), 1674–1689; doi:10.1287/mnsc.1040.0289.

Sosa, M., **Eppinger, S.** & **Rowles, C.** 2007 A network approach to define modularity of components in complex products. *Mechanical Design* **129** (11), 1118–1129; doi:10.1115/1.2771182.

Sosa, M. E., **Eppinger, S. D.** & **Rowles, C. M.** 2000 Designing modular and integrative systems. In *Proceedings of the ASME 2000 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* **4**: *12th International Conference on Design Theory and Methodology*, Baltimore, Maryland, USA, September 10–13, pp. 303–312. ASME; doi:10.1115/DETC2000/DTM-14571.

Stake, R. 1995 *The Art of Case Study Research*. Sage.

Starr, M. K. 1965 Modular production – A new concept. *Harvard Business Review* **43** (6), 131–142.

Steward, D. V. 1981 The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management* **28** (3), 71–74; doi: 10.1109/TEM.1981.6448589.

Troxler, P. 2019 Building open design as a commons. In *The Critical Makers Reader: (Un) Learning Technology* (**ed.** L. Bogers & L. Chiappini), pp. 218–226. Institute of Network Cultures.

Ulrich, K. 1994 Fundamentals of product modularity. In *Management of Design* (**ed.** S. Dasu & C. Eastman), 219–231. Springer; doi:10.1007/978-94-011-1390-8_12.

Ulrich, K. 1995 The role of product architecture in the manufacturing firm. *Research Policy* **24** (3), 419–440; doi:10.1016/0048-7333(94)00775-3.

Ulrich, K. & **Seering, W. P.** 1990 Function sharing in mechanical design. *Design Studies* **11** (4), 223–234; doi:10.1016/0142-694X(90)90041-A.

Whitfield, R. I., **Smith, J. S.** & **Duffy, A. B.** 2002 Identifying component modules. In *Artificial Intelligence in Design '02* (**ed.** J. S. Gero), pp. 571–592. Springer; doi: 10.1007/978-94-017-0795-4_27.

WikiHouse 2018a About Wikihouse, online document (downloadable on December 10th 2020) https://web.archive.org/web/20180413230102/https://wikihouse.cc/about, archived: 13 Apr 2018.

WikiHouse 2018b Homepage, online document (downloadable on December 10th 2020). https://web.archive.org/web/20180212183830/https://wikihouse.cc/, archived: 12 Feb 2018.

WikiHouse 2020 Homepage, online document (downloadable on December 10th 2020). https://wikihouse.cc/.

Yu, L. & **Ramaswamy, S.** 2007 Verifying design modularity, hierarchy, and interaction locality using data clustering techniques. In *Proceedings of the 45th Annual Southeast Regional Conference (ACM-SE 45)*, pp. 419–424. ACM: New York, NY, USA. http://dx.doi.org/10.1145/1233341.1233417.

Yu, T.-L., **Yassine, A. A.** & **Goldberg, D. E.** 2007 An information theoretic method for developing modular architectures using genetic algorithms. *Research in Engineering Design* **18** (2), 91–109; doi:10.1007/s00163-007-0030-1.