sensors

MDPI

Review

Generative Design in Building Information Modelling (BIM): Approaches and Requirements

Wei Ma [1], Xiangyu Wang [2,3,*], Jun Wang [4], Xiaolei Xiang [1] and Junbo Sun [1]

1 School of Design and the Built Environment, Curtin University, Bentley, WA 6102, Australia; wei.ma3@postgrad.curtin.edu.au (W.M.); Xiaolei.Xiang@curtin.edu.au (X.X.); tunneltc@gmail.com (J.S.)
2 School of Civil Engineering and Architecture, East China Jiao Tong University, Nanchang 330013, China
3 Australasian Joint Research Centre for Building Information Modelling, School of Built Environment, Curtin University, Perth, WA 6102, Australia
4 School of Engineering, Design and Built Environment, Western Sydney University, Kingswood, NSW 2747, Australia; jun.wang@westernsydney.edu.au
* Correspondence: xiangyu.wang@curtin.edu.au

Abstract: The integration of generative design (GD) and building information modelling (BIM), as a new technology consolidation, can facilitate the constructability of GD's automatic design solutions, while improving BIM's capability in the early design phase. Thus, there has been an increasing interest to study GD-BIM, with current focuses mainly on exploring applications and investigating tools. However, there are a lack of studies regarding methodological relationships and skill requirement based on different development objectives or GD properties; thus, the threshold of developing GD-BIM still seems high. This study conducts a critical review of current approaches for developing GD in BIM, and analyses methodological relationships, skill requirements, and improvement of GD-BIM development. Accordingly, novel perspectives of objective-oriented, GD component-based, and skill-driven GD-BIM development as well as reference guides are proposed. Finally, future research directions, challenges, and potential solutions are discussed. This research aims to guide designers in the building industry to properly determine approaches for developing GD-BIM and inspire researchers' future studies.

Keywords: generative design; building information modelling; technology integration; methodological relationships; skill requirement and improvement; novel development perspectives

Citation: Ma, W.; Wang, X.; Wang, J.; Xiang, X.; Sun, J. Generative Design in Building Information Modelling (BIM): Approaches and Requirements. *Sensors* **2021**, *21*, 5439. https://doi.org/10.3390/s21165439

Academic Editor: Hossam A. Gabbar

Received: 6 July 2021
Accepted: 9 August 2021
Published: 12 August 2021

1. Introduction

Generative design (GD) as a rule-driven iterative design process is based on algorithmic and parametric modelling to automatically explore, iterate, and optimise design possibilities by defining high-level constraints and goals [1,2]. Building Information Modelling (BIM) is a collection of regulations, procedures, and technologies enabling the creation, recording, and management of buildings' digital information through their entire life cycle [3–7]. The GD-BIM integration combines a new intelligent design approach and the technology of automated construction information generation [8–11]. It can facilitate the constructability of GD's automatic design solutions, and meanwhile improve BIM's capability in the early design phase. Thus, developing GD-BIM has drawn increasing attention academically and practically [12–16].

The current research regarding GD-BIM development mainly focuses on exploring applications and investigating tools. For instance, some GD-BIM are developed and studied to creatively tackle design issues, while some research examine software and programming means to compare developing tools [17–22]. However, there are a lack of methodological relationship studies, specifically, determination of proper developing approaches, skill requirements, and improvement paths based on different development objectives or GD properties; thus, the threshold of developing GD-BIM still seems high.

It is especially difficult for those designers in the building industry with little knowledge of GD or programming. Therefore, appropriate and practicable methodological guidance is in great demand to provide designers advice on developing GD in BIM.

The aim of this review is to investigate the current approaches of developing GD in BIM to discover methodological relationships, skill requirements, and improvement, to support designers on proper method selection for developing GD-BIM. Thus, publications regarding developing GD in BIM over the last decade are searched and reviewed in this study. Section 2 clarifies the necessity and logic of developing GD in BIM by investigating background knowledge. Section 3 explains research methods. Section 4 reviews, compares, and analyses the objectives, programming language suitability, and skill learning of developing GD-BIM, to propose perspectives of objective-oriented, GD component-based, and skill-driven GD-BIM development. In this section, a set of reference guides are suggested to designers on development methods selection, skill learning, and improving paths. Section 5 discusses future research directions and challenges and recommends potential solutions.

2. Generative Design and Building Information Modelling

This section aims to clarify the necessity and logic of developing GD in BIM by investigating background knowledge of GD, GD components, BIM, and GD-BIM integration.

2.1. Generative Design

GD is a rules-driven iterative design process [1]. It is based on algorithmic and parametric modelling to automatically explore, iterate, and optimise design possibilities by defining high-level constraints and goals [2]. Shea et al. [23] stated the aim of GD is to "explore creative and constructable designs by creating new design processes using the latest computing and manufacturing capabilities". With development of computer power and growing interest of researchers and practitioners, GD has become a new intelligent design approach and has been studied and applied in many fields academically and practically [8,24–38].

The most powerful capability of GD is to automatically explore and iterate design possibilities, and permutate the best solutions to human designers for decision-making [34]. This process usually occurs at the conceptual design stage, where GD can operate while other CAD applications are unable to support [39]. In fact, GD can explore design possibilities for any type of AEC designs (e.g., architectural design, structural design, interior design, urban design, or urban planning, etc.) at the design formulation stage. Thus, it is considerably useful to implement GD at the early design stage in the AEC industry.

2.2. Components of a GD

Currently, there are different classifications of GD components from various perspectives. Krish [39] breaks down the GD process into three components from the perspective of design: (1) "a design schema", (2) "a means of creating variations", and (3) "a means of selecting desirable outcomes". Marsh [40] discussed GD components from the view of performance measurement: (1) "Configuration Variation", (2) "Performance Metric", and (3) "Decision-Making Response". Nagy and Villaggi [2] states from the angle of formulating GD: (1) "a generative model expressing broad design possibilities", (2) "an evaluative component consisting of proposed design targets", and (3) "a metaheuristic search algorithm navigating the design iterations". It is seen that the classifications are determined based on the corresponding research objectives. Thus, in this study, a GD will be decomposed from the aspect of development, to facilitate the investigation of relationships between GD properties and development methods.

Generally, creating a GD consists of several steps: define design goals, formulate design constraints, determine algorithms, program the GD, run the GD, and modify the generated parametric models based on goals and constraints until satisfied [33,39]. Among them, programming the algorithms and design constraints are the major devel-

opment process, as the design goals are pre-defined, while the parametric models are automatically generated accordingly [2]. Therefore, in this study, a GD is decomposed into two major components from the perspective of developing its: (1) algorithm, and (2) design constraints.

The algorithm is usually highly abstract as it is used to instruct the computer to generate and optimise design possibilities through search methodologies. Singh and Gu [34] summarised five commonly used algorithms, namely, generative design techniques, as: Genetic Algorithms, Shape Grammars, L-systems, Swarm Intelligence, and Cellular Automata. As for usages, the proper selection of algorithms is determined according to the characteristic of design objectives [34]. For instance, if the design purposes are to discover space layout or visual compositions exploratorily, then the Shape Grammars algorithm is most likely selected. Or, if the objective is design improvement or optimisation, then the Genetic Algorithm is usually employed.

The design constraints are relatively intuitive and used to limit the scope of design exploration and narrow the search range of optimisation. The design constraints are a series of design conditions or preferences, such as spatial or morphological requirements, dimensional constraints, materials selections, manufacturing methods, or even cost constraints, etc. [26,29,41]. They are scripted by designers in a computer recognizable format to control the design exploration, iteration, or evolution [42–44]. As for scripting, design constraints are mostly manual scripted; however, research has started to develop automatic methods. For example, approaches to extract constraints-related information from text format to computer recognisable format have been studied and developed [45–48]. Yet, manually scripting of design constraints is still more common and flexible so far.

To identify the methodological relationship between development methods and the GD components, their different characteristics and properties need to be further studied. Detailed analysis is elaborated in Section 4 (Analysis section).

A figure is proposed to indicate the relationship of GD components and the process of running a GD, as presented in Figure 1. Firstly, the design goal is inputted into the GD program in the computer along with appropriate algorithms and well-defined design constraints. Then, parametric models are generated, iterated, and permuted to designers for decision-making. Finally, designers modify parameters in the algorithm and design constraints to adjust models until design goals are achieved.

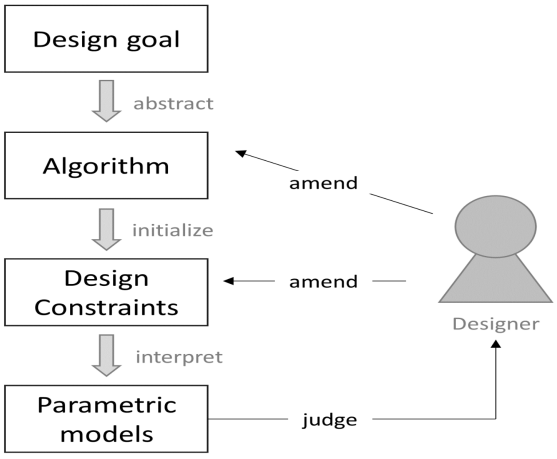

Figure 1. Relationship of GD components and the process of running a GD.

2.3. Building Information Modelling

As a revolutionary technology, BIM has rapidly changed the paradigm of a building's conception, design, construction, and operation [49–52]. Back to 1970s, the initial research of BIM began as parametric modelling research; however, the practical implementation of BIM in the building industry started from the mid-2000s [49]. Since then, BIM has quickly become the centrepiece of AEC technology [49]. Wang et al. [3] demonstrate

that BIM is a collection of regulations, procedures, and technologies enabling interacting to create a "digital representation of the projects' physical and functional characters". In BIM, digital formatting of the building's fundamental design and information data can be recorded and managed through the projects' entire life cycle [4–7]. Meanwhile, the quality of designs and datasets in BIM can be well controlled and improved by various approaches [53,54]. Therefore, it enables manipulation and maintenance of shared data and information resource for all users [55–57]. In conclusion, "BIM is not just software; It is a process and software" [49].

BIM software (e.g., Revit or ArchiCAD, etc.) provide an Application Programming Interface (API), allowing the access, extraction, selection, and modification of a building's data and information. Besides, the API provides users a platform to develop add-ons by writing a program or script to extend the application's capabilities [58,59]. For instance, the Revit API enables proficient Revit users to customize their own tools by programming with it to enhance Revit's capability, and to improve workflows [59].

2.4. Integration of GD with BIM

Although BIM is applicable throughout the project's entire life cycle, its usages in the design phase are mainly limited in the later design stages (e.g., the design development stage, structural design stage, mechanical, electrical, and plumbing design stage, etc.). In terms of design exploration in the early design stage or the creative phase (e.g., the conceptual design stage), the capability of BIM is inadequate. However, integrating GD demonstrates the potential to make up this deficiency of BIM.

GD, as a computer-aided design (CAD) method, mostly focuses on geometrical modelling [39,60] to quickly explore design. However, the "information" attribute usually cannot be generated. Differently, BIM produces components with both geometry and information attributes to facilitate buildability [60–63]. Thus, integrating BIM can improve the constructability of design solutions generated by GD.

The GD-BIM integration is beneficial to improve each other's capability by making up mutual deficiencies. The integration can support automatic and fast design explorations and enable buildability of these GD solutions, while extending BIM's capability in the early design phase.

To implement the integration, the API in BIM provides possibility and platforms. In fact, GD as a new feature is already available in Revit 2021 [64], the latest Revit version. Although the current GD availability in Revit is limited (only three design studies), the exploration and customization of more GD is allowed by using the Revit API. Therefore, developing GD in BIM (e.g., Revit) has a promising future.

3. Review Methodology

This research conducted a critical review using the content analysis-based review method [65–68] to obtain a novel understanding of methodological relationships, skill requirements, and improvement for developing GD in BIM. Scopus is selected as the searching database in this study as it is claimed the "largest abstract and citation database" [69] and provides the broadest overview of international and interdisciplinary scientific data and literature [70]. "Generative design" and "building information modelling" or "BIM" are the primary keywords for searching. However, GD is often confused with parametric design (PD) and algorithm design (AD), and used in parallel [71–74]. Therefore, it is imperative to understand the relationship among these items before determining the final searching keywords. According to Caetano et al. [71], "AD is a subset of GD", and "PD is orthogonal to AD and GD", which means AD and certain PD can be considered as GD as well. Therefore, AD and PD are added as a search key word as well to make the literature search as comprehensive as possible and avoid missing literature. To conclude, the keywords are determined as "generative design" OR "parametric design" OR "algorithm design" AND "building information modelling" OR "BIM".

The past 10 years (2010–2020) was set as the time frame for searching because developing GD in BIM is a newly emerged research area. There have been great leaps in computational design, including GD and BIM, in the last decade; therefore, review of the past 10 years period can provide a relatively sufficient overview of the present GD-BIM development [29]. The document type was limited to articles and conference papers to ensure the publication quality.

As such, the search criteria are as following:

- Scopus as the search database.
- Search within: article title, abstract, keywords.
- Search keywords: "Generative design" OR "Parametric design" OR "Algorithm design" AND "Building information modelling" OR "BIM".
- Publications published between 2010 and 2020.
- Document type: articles and conference papers.
- Publication language: English.

Initially, 114 documents were found in the Scopus database, based on the above search criteria. Non-related subject areas (e.g., Mathematics, Business, Management and accounting, Energy, Medicine, Chemical Engineering, Economics, Econometrics and Finance, Physics and Astronomy) are then excluded. Finally, 93 publications were selected as highly relevant to this study area for review; 61 out of the 93 publications are conference articles (accounting for 66%), while the rest 32 are journal papers (accounting for 34%).

The subject area of 'Engineering' makes up the largest proportion (46.2%), as shown in Figure 2, which indicates the domain emphasis. Table 1 summarizes the top 5 sources (in terms of amount and year) where the selected publications were published, including The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Automation in Construction, IOP Conference Series: Earth and Environmental Science, International Symposium on Automation and Robotics in Construction (ISARC), and Procedia Engineering. Table 1 indicates an increasing interest in this domain study.

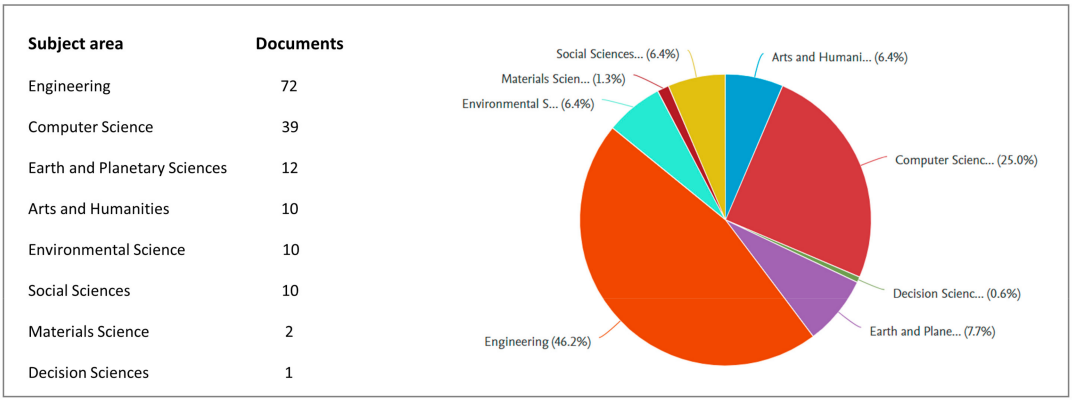

Figure 2. Distribution of subject areas of the selected 93 publications.

Table 1. A summary of the top five publication sources in terms of amount and year (2010–2020).

Source Title	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Sum
CAADRIA		1	1	1	2	1	2	3		2		13
Automation in Construction				2		1			4	2	1	10
IOP Conference Series: Earth and Environmental Science										3	3	6
ISARC				1			3	1	1			6
Procedia Engineering					2		2					4
Sum	0	1	1	4	4	2	7	4	5	7	4	39

To investigate the methodology of developing GD in BIM, it is essential to understand the development objectives, characteristics of the development objects, development tools, and skill requirement. Accordingly, the categories and characteristics of GD-BIM development objectives, characteristics of GD components, features and applications of programming languages, and skill requirement and improvement frame the detailed review in this study. Based on that, three major topics are analysed to discover the methodological relationships and skill requirements, including:

1. Relationships between programming languages and developing different GD-BIM objectives.
2. Relationships between programming languages and developing different GD components.
3. Skill requirement and learning paths of programming languages for developing GD-BIM.

Mixed methods of quantitative and qualitative research are used to review and analyze the above three topics. Firstly, qualitative research method is used to categorise different objectives of GD-BIM development, followed by quantitative research to investigate relationships between GD-BIM development objectives and applications of programming languages. Secondly, qualitative research is employed to examine the characteristics of GD components and programming languages respectively to analyze the suitability relationship between programming languages and developing GD components. Thirdly, to review discussions on skills of programming languages used to develop GD in BIM from the selected literature and qualitatively study the skill requirements and improvement paths. Accordingly, three novel perspectives of GD-BIM development and a set of reference guides are proposed. Detailed analysis is elaborated in the following section.

4. Analysis

In this section, the GD-BIM development objectives, programming languages suitability, and programming skills learning and improvement are reviewed and analysed to investigate the methodological relationships. Accordingly, perspectives of objective-oriented, GD component-based, and skill-driven GD-BIM development are proposed, as well as a set of reference guides to designers in the building industry.

4.1. Objectives of Developing GD in BIM and the Relationship to Programming Languages

4.1.1. Categorising and Comparison of Different Objectives of GD-BIM Development

Through analysing the different characteristics of the objectives, the GD-BIM development can be classified into two major categories: (1) solve specific design tasks, and (2) support design processes. Table 2 compares these two categories in detail. The development category of solving specific design tasks is usually project-based and mainly developed by practitioners aiming to solve individual design tasks creatively and efficiently within larger practical projects. To achieve this objective, GD is used to automatically explore and optimise design alternatives, while BIM generates multidimensional information to facilitate the constructability. Detailed information of the examples for this objective category can be found in [75–82], presented in Table 1. The advantages of these GD-BIM developments are targeted and practical problem-solving, lower development difficulty, and a shorter development period. However, the drawbacks of these developments are less universal (as they are conducted based on specific practical design conditions) and too preliminary for complicated tasks.

Table 2. A summary of objective categories of developing GD in BIM.

Objective Category	Description	Sub-Objectives	Characteristic	Evaluations	Examples	Programming Method	Programming Language
1. Solve specific design tasks.	This objective is to solve individual design tasks creatively and efficiently within larger practical projects.	• Cope with design changes efficiently. • Shorten the design period. • Generate, document, and fabricate designs in great detail. • Improve design efficiency by integration with traditional design workflow. • Generate parametric models efficiently. • Explore building forms automatically. • Explore and generate facade designs automatically.	• Project based. • Practical design task orientated. • Mostly developed by industry practitioners.	• Relatively easy to develop and amend. • Short development pe-riod.Relatively easy programming requirement. • Solutions are high targeted but less universal. • Currently too preliminary for complicated tasks.	Automate design and production for practical tunnel projects [75].	Grasshopper (in Rhino) and Dynamo (in Revit) used in the Application Programming Interfaces (API's).	Visual programming language (VPL)
					Digital aided façade design [76].	An add-in named GA (generative design) in Grasshopper.	VPL
					Digital workflows in contemporary architecture and construction [81].	Objected-oriented programming, functional programming, visual programming, and distributed visual programming [81], based on cases.	Textural programming language (TPL) and VPL
					An algorithmic BIM approach in a traditional design studio [82].	Python and AutoLisp are used to script algorithm; Rosetta is used to support BIM back-end.	TPL
					BIM-integration of solar thermal systems [77].	Dynamo in Revit.	VPL
					BIM-based parametric modelling to Tapered Slip-Form System [78].	SmartParts Script Language of Allplan (a BIM tool).	TPL
					Parametric design of Shanghai Tower's form and façade [79].	Microsoft Visual C# that ran between Grasshopper and Revit [79].	TPL.
					BIM façade module for diagrid twisted structures [80].	Dynamo in Revit.	VPL

Table 2. *Cont.*

Objective Category	Description	Sub-Objectives	Characteristic	Evaluations	Examples	Programming Method	Programming Language
2. Support design processes.	This objective aims to support design processes in BIM by building environments or systems integrating with GD in the context of BIM.	• Automate the design evaluation. • Automate designing, modelling, and documenting of customized designs. • Propose an early design workflow in reducing construction waste. • Enhance the applicability of BIM in various design stages (such as conceptual design, structural design, etc.). • Enhance the practicality of BIM in various types of design (such as interactive brickworks design, green building design, and adaptive facade design, etc.). • Establish a portable platform to develop GD in BIM. • Customise design tools.	• Research based. • System development orientated. • Mostly developed by researchers.	• Higher programming requirement. • Higher development difficulty. • Long development circle. • Solutions are higher universally, but at the prototype stage so far. • Currently too elementary to support complex design processes.	Generative design for building interiors using BIM [58].	domain-specific language (DSLs) is used to script the design rules.	TPL
					Generative BIM workspace for conceptual design automation [84].	C#.Net in Revit Add in.	TPL
					Generative interior design using BIM [85].	DSLs to script the design rules.	TPL
					Automatic structural design of RC wall-slab buildings in BIM [86].	Not mentioned.	N/A
					Automated design and modelling for mass-customized timber structure housing [87].	Grasshopper in Rhino.	VPL
					From layout generation to construction document production of customised apartment plans [83].	Existing grasshopper (GH) workflow nodes are used to script design rules, Python in GH is used to create new nodes and script the algorithm, and the processing language is used to develop the Graphic User Interface (GUI).	VPL and TPL
					A novel construction waste reduction workflow using parametric design and module coordination [88].	Existing nodes in GH are used to develop the algorithm.	VPL

Table 2. *Cont.*

Objective Category	Description	Sub-Objectives	Characteristic	Evaluations	Examples	Programming Method	Programming Language
					Portable generative design for BIM [60].	An Integrated Development Environment named Rosetta is used to support various TPLs as a front-end, and a series of CAD and BIM applications are connected as back-ends for GD model generation.	TPL
					Design and analysis in a generative tool with multi back-ends [98].	Same as the above.	TPL
					Towards cloud informed robotics [89].	Visual programming for parametric design.	VPL
					A framework for a dimensional customization system [90].	Not mentioned.	
					A Green-BIM approach for adaptive building facade optimisation [91].	Dynamo is used for information extraction; C# is used to develop compliance checking systems.	VPL and TPL
					Virtual generative BIM workspace for maximising conceptual design innovation in the AEC industry [92].	C#.Net programming.	TPL
					Exploit AEC conceptual design innovation by integrating GD with BIM [93].	C#.Net programming	TPL

Table 2. *Cont.*

Objective Category	Description	Sub-Objectives	Characteristic	Evaluations	Examples	Programming Method	Programming Language
					G-BIM framework and development process for design automation [94].	C#.Net programming	TPL
					Integrated generative technique for brickworks interactive design [95].	Grasshopper in rhino is used to script the algorithm. Processing is used to create the sketch tool.	VPL and TPL
					Parametric and generative methods with BIM [96].	C#	TPL
					Design of parametric software tools [99]	Grasshopper in Rhino	VPL
					Tool design for architectural design [100].	Grasshopper in Rhino	VPL
					Parametric design based on BIM for sustainable buildings [97].	C# programming in Revit API	TPL

The development category of supporting design processes is not based on projects; instead, it is mainly developed by researchers, aiming to support certain design processes by establishing environments or systems in the context of BIM. These developments can enhance the applicability, practicality, and compatibility of BIM in certain design stages or fields. For instance, some GD-BIM are created to support designs in certain phases, such as the conceptual design phase, interior design phase, structural design phase, or design customization stage, etc. [58,83–97]. Meanwhile, some studies concentrate on improving the portability of GD-BIM developments [60,98–100]. These developments have advantages of higher universality, as they can work as platforms or programs allowing use by a wider group of designers for similar design purposes. However, the disadvantages are higher programming requirement, higher developing difficulty (due to the need of multidisciplinary involvement), and long development cycle (due to the need of constant testing and iteration before practical application). Also, most of the current developments are at the prototype stage or the rhetorical stage, and too elementary to support complex design processes [84,88,94].

4.1.2. Publications of Different Objectives of GD-BIM Development

Out of the filtered 93, 28 literatures (listed in Table 1) have discussed developing GD in BIM. Figure 3 shows the distribution of publications based on the different objectives classified above. Publications about the objective of "support design processes" occupy the majority (20 in 28), which indicates the research focus and preference in the academic domain. Moreover, the increasing number over the last five years illustrates growing research in this area of study.

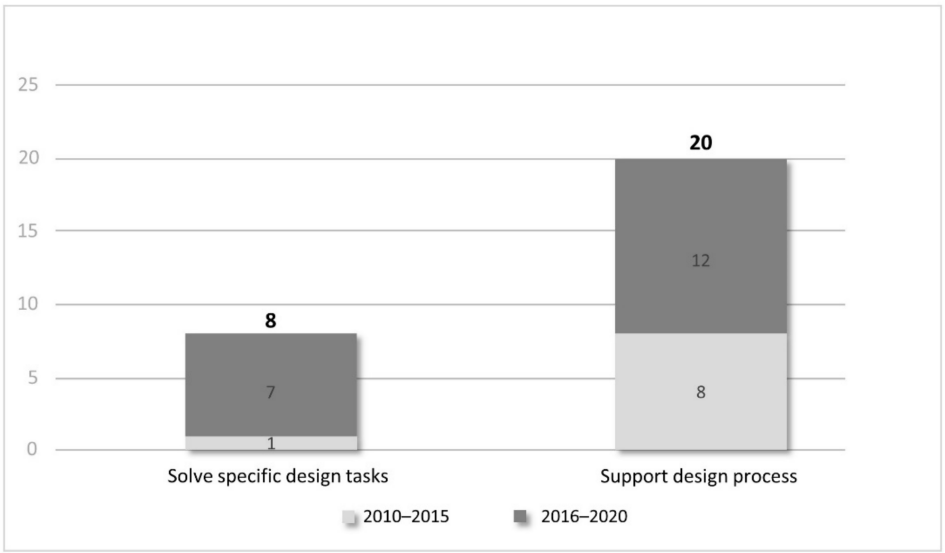

Figure 3. Distribution of publications on different objectives of GD-BIM developments (2010–2020).

4.1.3. Application of Programming Languages Based on Different Objectives

Programming methods and languages are examined and listed following each example in Table 1. Categories of textural programming languages (TPL) and visual programming languages (VPL) are adopted and investigated in this research. A TPL is a programming language that uses text, code, symbol, and predefined syntax, etc. to develop programs, while a VPL allows users to develop programs by interactively manipulating visual elements, rather than scripting texturally [60,101]. Based on Table 1, there are 6 cases using VPL and 5 cases using TPL in the Objective 1, while 8 cases using VPL and 13 cases using TPL in Objective 2. Accordingly, Figure 4 shows the application distributions of programming languages. It is seen that to develop GD-BIM for solving specific design tasks,

the use of VPLs (55%) is slightly more than that of TPLs (45%); to develop GD-BIM for supporting design processes, TPLs are more used than VPLs (62% and 38% respectively).

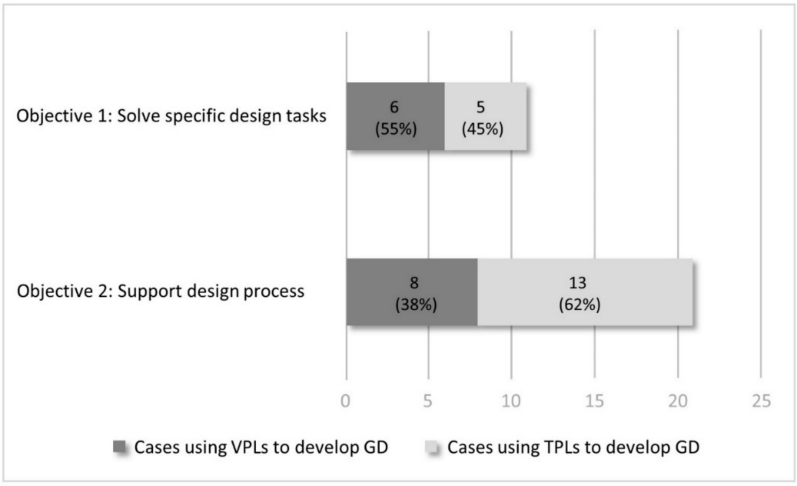

Figure 4. Application distributions of programming languages based on objectives of GD-BIM development.

Therefore, the methodological relationship can be concluded: VPLs are applied more to individual practical design tasks, for easier and quicker GD-BIM developments, while TPLs are more accepted for harder, longer, and more systematic GD-BIM developments, to establish application platforms or environments.

4.1.4. Perspective of Objectives-Oriented GD-BIM Development

A novel perspective of objective-oriented GD-BIM development is proposed based on the above analysis, which is the selection of proper development of methods according to the different objectives of developing GD-BIM. Firstly, GD-BIM developments have two major categories based on development objectives, which are "solve specific design tasks" and "support design processes". Secondly, utilisation of programming methods and languages differ according to different objectives. Thirdly, VPLs are more applicable to easier and quicker GD-BIM development for the objective of solving specific practical design tasks, while TPLs are more capable to develop harder, longer, and more systematic GD-BIM, to establish application platforms or environments for the objective of supporting design processes.

4.2. Suitability of Programming Languages for GD-BIM Development

4.2.1. Programming Languages and Software Used to Develop GD in BIM

A programming language and software (usually BIM software) are crucial and indispensable tools in GD-BIM development [17,102,103]. Table 3 compares VPLs and TPLs in detail regarding definitions, languages, advantages, and limitations [12,17,101–104]. Table 4 illustrates the popular software and applicable programming languages for scripting GD [25,103,105]. As shown in the Table 3, Revit, ArchiCAD, Grasshopper, and GenerativeComponents are the commonly used software for developing GD. Among them, Revit and ArchiCAD are BIM software, while Grasshopper and GenerativeComponents are not (they are CAD software). However, Grasshopper and GenerativeComponents can connect to BIM software by means of plug-ins or a specific edition, as indicated in the Table [102,105]. Therefore, all of them can be considered as platforms for GD-BIM development. As Grasshopper and Dynamo have abundant content in libraries and are easily accessible forums for learners, and Python is easier to learn compared to C++, Grasshopper and Revit are currently more popular for developing GD-BIM.

Table 3. Comparison between VPLs and TPLs regarding definitions, languages, advantages, and limitations.

Categories	Definitions	Languages	Advantages	Limitations
Visual programming languages (VPLs)	Any programming language that allows users to develop programs by manipulating visual elements interactively.	• Grasshopper • Dynamo • Optimizer Node • etc.	• Simpler operation. • Intuitive and interactive. • Immediate visual feedback. • Good Interactive Development Environment (IDE). • Processes can be described graphically and intuitively by a form of 'pipes-and-filters' logic. • Convenient for small program development. • Faster learning curve. • Little programming background requirement. • More productive and motivating for novices.	• Apply to very limited domains and highly specialized solutions. • Scale poorly with large and complex design tasks. • Absence of (sophisticated) abstraction mechanisms, resulting in redundancy.
Textural programming languages (TPLs)	Any programming language that uses lines of text, code, symbol, predefined syntax, etc. to develop programs.	• Python • C# • C++ • JavaScript • AutoLISP • Haskell • VisualScheme • etc.	• Various types of abstraction mechanisms. • Easy to adjust to changing requirements. • Significantly more productive for large scale and complex design tasks.	• Absence of good IDE. • Hard to learn for novices. • Steeper learning curve. • Require extensive knowledge and proficient skill.

Table 4. Commonly used software and applicable programming languages for scripting GD.

Software	Developer	BIM	Connectable to BIM	Plug-In or Stand-Alone	Applicable Programming Languages for Scripting GD	
					VPLs	TPLs
Revit	Autodesk	Yes	N/A	Stand-alone	Dynamo	Python, C# in Revit API
ArchiCAD	Graphisoft	Yes	N/A	Stand-alone	Grasshopper—Archicad Live Connection	C++ in ArchiCAD API
Grasshopper	McNeel	No	Yes; by Lyrebird, etc.	Plug-in for Rhinoceros	Grasshopper	GhPython in Grasshopper API
GenerativeComponents	Bentley	No	Yes. GenerativeComponents CONNECT Edition.	Stand-alone and Plug-in for MicroStation	Optimizer Node in CONNECTION Edition	GCScript, C#

As indicated in Table 2, TPLs and VPLs have different characteristics, pros, and cons. Based on these features, studies have examined, compared, and evaluated their usages and suitability in GD development [17,60,81,106]. For instance, Leitão et al. [17] have conducted a comparative study to demonstrate that modern TPLs are more productive in developing complex GD, as TPLs have the advantage in abstraction mechanisms. However, due to their difficulty to learn, few designers possess extensive knowledge and proficient ability of TPLs to skilfully use them [60]. In contrast, VPLs are much easier to learn and grasp for beginners because of the advantages of intuitive and interactive properties and simpler operation [107]. However, VPLs' limitation in sophisticated abstraction mechanism restricts their capability in complex GD development.

Therefore, in general, VPLs are more suitable to novice designers and simple GD-BIM development, while TPLs are more suitable to expert players and complex GD-BIM development. However, most of the current studies overlook the relationship to GD components, which are inherent properties of GD-BIM. This gap is addressed in the following section.

4.2.2. Suitability Relationship between Programming Languages and GD Component Development

The application of programming languages varies, depending on the properties of GD components. Generally, the algorithms component is highly abstract while the design constraints component is intuitive. Thus, programming languages used to script these components should have the corresponding characteristics. Leitão et al. [17] have summarised the three fundamental dimensions of a full-featured programming language: primitive, combination, and abstraction. Different programming languages have different advantages in these dimensions. Specifically, VPLs have the advantage in primitive dimension, while TPLs have strong points in combination and abstraction dimensions.

Thus, a suitability relationship is found between programming languages and GD components development, as indicated in Table 5 [12,17,101,104]. VPLs have better primitive dimensions, resulting in good performance of expressing intuitive ideas by simply connecting various primitive components. Thus, VPLs are suitable to develop the "design constraints" component and straightforward algorithms which require intuitive and explicit expression. However, due to the shortcomings in combination and abstraction mechanisms [17], VPLs are not the ideal choice for complex algorithm development (e.g., genetic algorithms, shape grammars, etc.). Due to expression and production in complex and abstract tasks, TPLs are more suitable to develop the complex algorithms component. TPLs can also be used to develop complicated design constraints that require intricate scripting in VPLs.

Therefore, the methodological relationship can be concluded: TPLs are more suitable to develop abstract components such as the Algorithm component or intricate design constraints, while VPLs are more suitable to develop intuitive components such as the design constraints component or simple algorithms.

4.2.3. Perspective of GD Component-Based GD-BIM Development

A new perspective of GD component-based GD-BIM development is proposed, which is the selection of proper development methods based on different GD components. Firstly, developing a GD is mainly composed of programming the algorithm and design constraints components. Secondly, the use of programming methods and languages differ based on the different characteristics of the GD components. Thirdly, TPLs are more suitable to develop abstract components, such as the algorithm component or intricate design constraints, while VPLs are more suitable to develop intuitive components such as the design constraints component or simple algorithms.

Table 5. Suitability relationship between programming languages and GD components development.

Programming Languages	Characteristics of Programming Languages on Fundamental Dimensions			Suitability for GD Component Development
	Primitive	Combination	Abstraction	
VPLs Such as: • Dynamo • Grasshopper • Optimizer Node	• Implement a large set of primitive components (e.g., ranges, mappings, and geometric operations). • Primitive components are with a high degree of sophistication. • Good in expressing intuitive ideas by connecting primitives.	• Provide a single combination paradigm. • Rely on an extremely simple metaphor (primitives combined by connecting outputs and inputs of components). • This metaphor is too restrictive to express: ➢ complex control structure (e.g., iteration, recursion, etc.), and ➢ algorithms without using a TPL to script customized primitive components. • Scale poorly with complex design tasks, may result in reading and maintenance issues.	• Provide 'Cluster' function, which allows to group other components or clusters as a single component. • Good for improving programs clarity and parts reuse. • Inability in centralized definition (as clusters are independent from their copies). • Not real abstraction.	• Intuitive design constraints, • or straightforward algorithms.
TPLs Such as: • Python • C# • C++ • JavaScript • AutoLISP • Haskell • VisualScheme	• Currently not as many primitives as VPLs.	• Provide various combination paradigms, such as functional, imperative, and object-oriented. • Including expression composition, and various control and data structures. • Paradigms are extendable. • Relay on underlying languages. • Expressive in complex design tasks.	• Provide various types of procedurals, data, and control abstraction. • Good in simplifying solutions. • Much easier to adjust to changing requirements. • Significantly more abstract than VPLs.	• Abstract algorithms, • or intricate design constraints.

4.3. Programming Skill Learning & Improving for GD-BIM Development

4.3.1. Designers' Learning of VPLs and TPLs

Research has revealed that VPLs have a faster learning curve, while TPLs have a steeper one [17,101,102,104]. VPLs are easy to learn and use as they were invented to enable easy interaction with computers and became educational tools to teach beginners programming [108]. In terms of conducting design tasks, VPLs enable simpler and more straightforward operation due to their sophisticated IDE and intuitive elements [17]. Thus, VPLs are a good starter for designers to learn programming and create simple programs, despite shortcomings in dealing with complex tasks.

TPLs are much harder to learn and master because they rely on hard coding and a complicated syntax system, especially for creative professionals such as designers who are not used to linear thinking [104]. However, additional time and effort spent on learning TPLs can be "quickly recovered once the complexity of problems becomes significantly large" [17,81]. Owning knowledge and experience of TPLs can even effectively enhance the productivity of VPL users [17]. Therefore, although hard to learn, TPLs can be a quality weapon for designers to improve design capability and productivity by developing complex programs and customising design tools.

4.3.2. Influence from Portable Development Environments

Some studies have demonstrated that portable development environments can reduce the requirement of learning new programming skills [17,60,81,82,98,102,109]. For instance, Rosetta as a portable development environment enables a user proficient in one TPL to create GD in various BIM environments without learning extra new programming languages. Rosetta plays as the medium or converter role and allows users to script GD using certain TPLs (as front-end languages) and generate identical GD models in multiple software (as back-end environments).

Rosetta is beneficial to learning cost and design efficiency because of the portability and interoperability. A study conducted by Caetano and Leitão [82] has demonstrated this, in which Rosetta took advantage of both CAD and BIM to create outstanding design solutions under a very tight deadline. Usually, CAD is more efficient in design exploration than BIM, while BIM is much better at generating construction information. In this project, designers first used a familiar TPL in Rosetta to explore designs and generate design models in a CAD software. Then, once the design was confirmed, designers used the same scripts in Rosetta, but connected to a BIM software to generate an identical model equipped with construction information. Finally, a satisfying design solution with detailed construction guide was produced in a very short time. Hence, learning and use of different programming languages was avoided due to the portability of Rosetta between BIM and CAD, which greatly saved time and improved design efficiency.

Despite the advantage in reducing learning effort for technically capable users, Rosetta is not usable for all levels of designers. Studies have pointed out the drawbacks. Firstly, back-end software that Rosetta currently support is inadequate [102]. Secondly, specialised TPL programming knowledge is required [82]. Proficiency in at least one applicable TPL is currently necessary to use Rosetta to either script GD or add new back-end; however, few designers possess TPL skills. To conclude, Rosetta can provide positive effects in programming skill learning and GD-BIM development to designers skilled in TPLs.

4.3.3. Recommendations to Designers on Skill Learning and Improving

Based on the analysis above, paths of skill learning and improvement of GD-BIM development are recommended to designers, as shown in Figure 5. Firstly, designers without programming skills can start by learning VPLs (e.g., Dynamo for Revit, Grasshopper for Rhino, etc.) because VPLs are intuitive and easier for novices. Secondly, after mastering a certain VPL, designers can develop simple GD with the existing workflow nodes of VPLs to improve skills. Thirdly, learning one TPL is advised as the next step because modern

TPLs are more productive in developing complex tasks [17]. Python is suggested as it is easier to learn and apply compared to other complex TPLs such as C++ [81], and widely accepted by the majority of applications, including Revit and Grasshopper. Afterwards, the path can follow two directions once skilled in TPLs: (1) using TPLs to improve GD-BIM development ability and programming skills in VPLs, and (2) using TPLs and Rosetta to improve GD-BIM development ability and efficiency. Refer to the figure for detailed information.

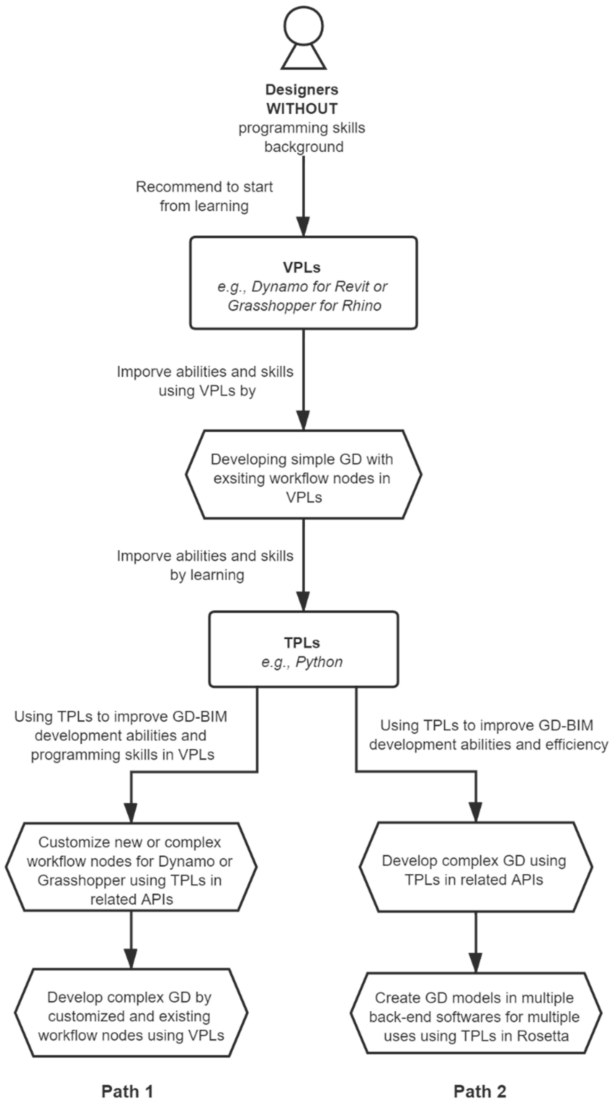

Figure 5. Paths of skill learning and improving GD-BIM development for designers.

4.3.4. Perspective of Skill-Driven GD-BIM Development

An original perspective of skill-driven GD-BIM development is proposed. Firstly, skill in at least one type of programming language (either a VPL or TPL) is necessary for designers to develop GD-BIM. Secondly, learning and proficiency in a VPL or TPL requires different investments of time and energy; thus, they are applicable to different stages and levels of GD-BIM development. Thirdly, the ability and efficiency of GD-BIM development can be progressively enhanced by strategically learning and improving VPL and TPL skills.

5. Discussion

There are two major aspects worth discussing based on the review and analysis. First, the GD-BIM developments to support design process have dominated the research

according to the investigation of publication distribution, as shown in Figure 2. Increasingly, researchers have explored the integration of GD into BIM to establish applications or systems, allowing use by a wider group of designers. However, current attempts for this objective are still at the prototype stage and too preliminary to solve complicated design issues. Therefore, more sophisticated and systematic GD-BIM developments to support more design processes is one future research direction.

Second, skill requirement for developing complex GD-BIM is high, because proficiency in TPLs is necessary and difficult; it requires significant time to learn and practice, which is unfriendly to most designers who are not used to programming, thus restricting wider development and application of GD-BIM. Therefore, reducing programming difficulty for designers will be another future research direction to facilitate GD-BIM development.

As such, future directions, challenges, and potential solutions are mapped in Table 6 [110] and discussed in the following section.

Table 6. Future directions of GD-BIM development and corresponding challenges and potential solutions.

Future Directions	Challenges	Potential Solutions
Develop more sophisticated and systematic GD-BIM to support more design processes.	• Lack of diversity of GD design scenes. • Lack of consideration for comprehensive design elements. • Lack of mature GD-BIM environments convenient to use for complicated design. • Lack of wider integration to form a more extensive intelligent system. • High development skill requirement.	• Explore more sophisticated design scenes. • Introduce detailed and realistic design elements as GD constraints. • Built-in complex search algorithms for convenient selection and use. • Develop more industry-usable GD-BIM applications for complex design. • Integrate with upstream or downstream technologies such as computer visioning or intelligent output.
Reduce programming difficulties for designers.	• High programming requirement for designers. • Lack of prior knowledge and resource of GD.	• Develop more diversified, and novice-friendly, compatible, and portable development environments. • Create more domain-specific and easy-to-use programming languages. • Establish expandable GD library to leverage prior knowledge.

5.1. Develop More Sophisticated and Systematic GD-BIM to Support More Design Processes

To develop and apply more sophisticated GD-BIM applications, the information below discusses challenges and methods. The current GD-BIM applications only consider limited design elements, as GD attempts to quickly resolve simple design tasks, and presents design scenes as research backgrounds too simply to facilitate prototype study. As a result, the current drawbacks are the lack of comprehensive design elements, diversity of design scenes, and mature and useful applications. Therefore, challenges are mainly three aspects: comprehensiveness of design elements, diversity of design scenes, and maturity of applications. To address these, firstly, detailed and realistic design elements such as locations, dimensions, materials, multiple types of cost, energy performance, etc. should be introduced to form complex GD constraints. Secondly, more design scenes covering diverse design processes (e.g., from conceptual design to detailed design) are advised to be studied, abstracted, and scripted into GD-BIM applications as research backgrounds. Thirdly, various complex search algorithms (e.g., genetic algorithms, shape grammars,

cellular automata, etc.) can be built-in for easy selection and use, based on practical tasks to facilitate wider application of GD-BIM.

To implement broader integration, challenges and solutions are discussed below. The intelligence extent of GD-BIM integration has much room to expand because their current applications are mostly during the post-site analysis and pre-construction stages. Therefore, a more extensive intelligent system can largely enhance GD-BIM's capacity to solve complicated problems. However, the largest challenge is to conduct wider integrations based on GD-BIM, to form an entire intelligent system from pre-design (site analysis) to construction. To address this, BIM can be a bridge to connect upstream and downstream technologies. A framework of an extensive intelligent system formed by broad technology integration based on GD-BIM is proposed, as presented in Figure 6. Firstly, upstream technologies such as computer visioning or sensor technology [110–113] can be integrated to intelligently capture site information and form design constraints. Secondly, downstream technologies such as 3D printing or robotic arms in construction [114–121] can be integrated to automatically construct the GD solutions generated beforehand. Moreover, GD-BIM demonstrates promising potential in smart engineering composites [122–124] as well as artificial intelligence aided optimization [125–128]. Thus, an entire automated process from site analysis, design exploration, construction information generation, to design construction would be established, which could considerably save manpower and improve the buildability of designs. This broad integration grown out of GD-BIM would have great potential to solve comprehensive and complicated design issues and support multiple design processes.

5.2. Reduce Programming Difficulties for Designers to Facilitate GD-BIM Development

Another direction is to reduce programming difficulties for designers to enable easier development. Usually, designers in the building industry are not well equipped with programming skills, and the existing programming languages are not perfect for all level designers. Despite this, there have been instances of successful GD-BIM applications to familiarise and master programming; this is still a huge threshold for designers. Therefore, if the barriers of programming could be reduced, more designers would be willing to develop GD-BIM, which is beneficial for both technology and industry. To reduce the barriers, three potential solutions are suggested: (1) develop more compatible and portable development environments, (2) create more domain-specific and easy-to-use programming languages, and (3) establish GD library to leverage prior knowledge.

Firstly, compatible and portable development environments, enabling designers proficient in one TPL to develop GD in multiple software for multiple usages, can reduce programming difficulty. One example of a portable development environment is Rosetta; it can reduce the requirement of learning new programming languages. However, there are shortcomings with Rosetta, such as insufficient back-end software, and high TPL skill requirements [82,102]. Thus, the current Rosetta is not usable by all levels of designers, especially those who have little TPL knowledge. To address this, more back-end connections or novice-friendly development environments accepting VPLs should be developed in the future.

Secondly, programming languages more conforming to design languages can lower the difficulty of learning brand new languages for designers, thereby reducing programming difficulty. The text format domain-specific languages (DSLs) have this characteristic, and "modern TPLs coupling with domain-specific primitives become better alternatives to current VPLs for developing GD" [17]. They have already been applied in some GD-BIM developments for easier use of designers [58,78,85]. However, the usage of current text DSLs are still close to professional TPLs; thus, knowledge of textural programming is required. Therefore, creating more DSLs conforming to design languages and improving the availability to beginners are recommended.

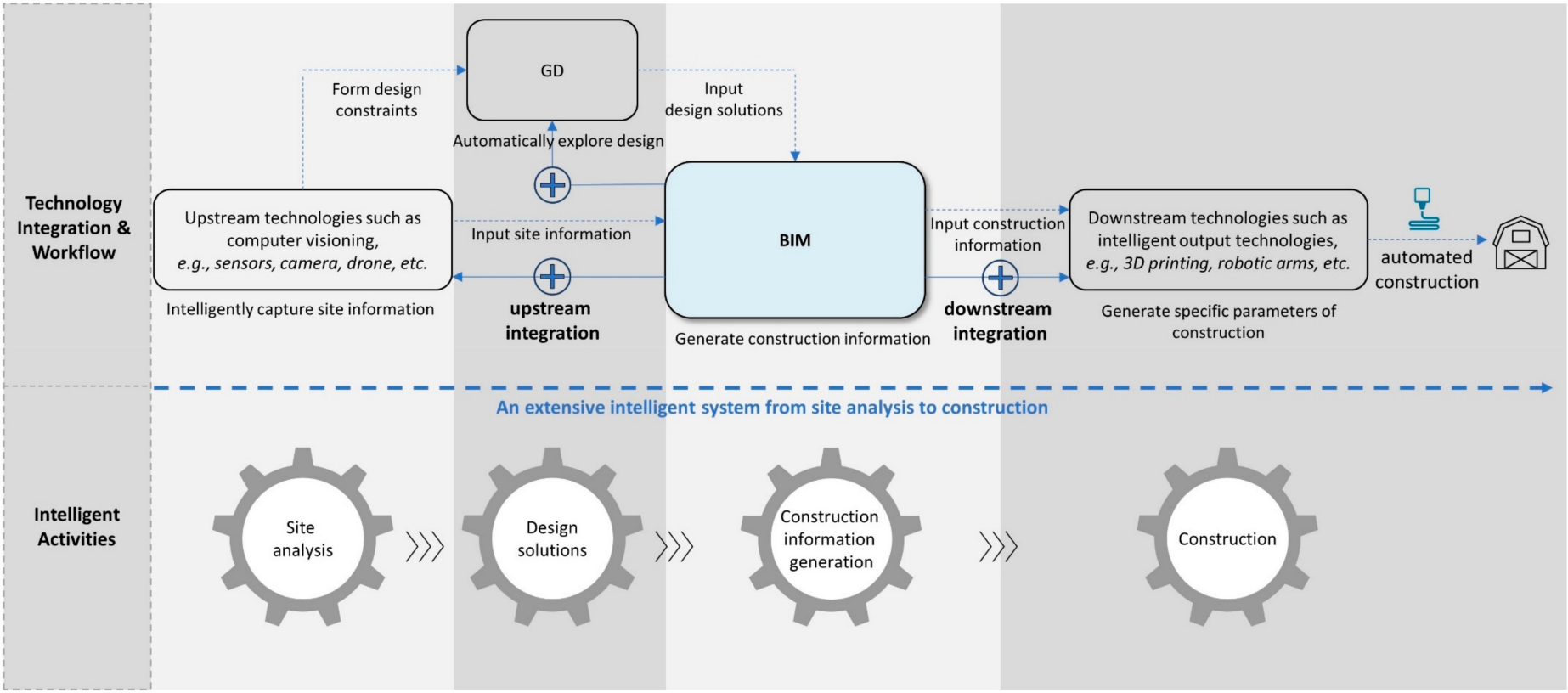

Figure 6. A framework of an extensive intelligent system (from site analysis to construction) formed by broad technology integration based on GD-BIM.

Thirdly, decreasing the reliance on programming by leveraging prior knowledge or resources can contribute to reduced programming difficulty. If the prior knowledge or resource of GD were well documented, accessed, and reused, then less programming would be required when developing new GD. For instance, three simple GD are available in Revit 2021 [64], which eliminates the demands of developing extra GD with similar functions, thus reducing programming necessity. Therefore, it inspires the recommendation of establishing expandable GD libraries in BIM that allow simple and flexible storage, access, use, and modification to decrease the reliance on programming.

6. Conclusions

The purpose of this review is to investigate the current approaches to developing GD in BIM by reviewing publications over the past decade. This research demonstrates a novel understanding of methodological relationships, skill requirement, and improvement for developing GD in BIM. The significance is to support designers in the building industry on the proper methods selection for developing GD-BIM. It is clear from the review and analysis that programming skills are necessary for designers to develop GD-BIM, and different types of programming languages have different suitability based on development objectives and GD components. Specifically, VPLs are more applicable to develop simple GD-BIM to solve specific practical design tasks, while TPLs are more capable of developing complex and systematic GD-BIM to establish application platforms for supporting design processes. Regarding developing GD components, TPLs are more suitable to develop abstract components such as an algorithm component or intricate design constraints, while VPLs are more suitable to develop intuitive components such as the design constraints component or simple algorithms. The skills of programming and developing can be progressively enhanced by strategically learning and improving. Accordingly, three novel perspectives of objective-oriented, GD component-based, and skill-driven GD-BIM development, as well as a set of reference guides, are proposed regarding development method selection, skill learning, and improvement.

It is also found that the GD-BIM developments to support design process have dominated the research. However, most of the current attempts are still at the prototype stage and too preliminary to solve complicated design issues. Also, the skill requirement for developing complex GD-BIM is high, as proficiency in TPLs is necessary and difficult, increasing the challenges in developing methods. Therefore, future research directions are discussed, regarding: (1) development of more sophisticated and systematic GD-BIM to support more design processes, and (2) facilitation of GD-BIM development by reducing programming difficulties for designers. The main challenges are identified, and potential solutions are recommended.

In conclusion, this review aims to guide designers in the building industry to select proper methods or formulate skill-improving paths to develop GD-BIM and provides an inspired map for researchers to explore new knowledge.

Author Contributions: Conceptualization, W.M. and X.W.; methodology, W.M.; software, W.M.; validation, W.M.; formal analysis, W.M.; investigation, W.M.; resources, W.M., X.W. and J.S.; data curation, W.M. and X.X.; writing—original draft preparation, W.M.; writing—review and editing, W.M., J.W., J.S.; supervision, X.W. and J.W.; funding acquisition, X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was co-funded by the Australian Research Council's Linkage Projects, grant number LP180100222.

Acknowledgments: I would like to extend my sincere thanks to Yongze Song for his valuable advising and discussion.

Conflicts of Interest: On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Meintjes, K. '"Generative Design"—What's That?—CIMdata', CIMdata. 2017. Available online: https://www.cimdata.com/en/news/item/8402-generative-design-what-s-that (accessed on 13 June 2020).
2. Nagy, D.; Villaggi, L. 'Generative Design for Architectural Space Planning'. Autodesk University. 2020. Available online: https://www.autodesk.com/autodesk-university/article/Generative-Design-Architectural-Space-Planning-2020 (accessed on 15 June 2020).
3. Wang, X.; Love, P.E.D.; Kim, M.J.; Park, C.-S.; Sing, C.P.; Hou, L. A conceptual framework for integrating building information modeling with augmented reality. *Autom. Constr.* **2013**, *34*, 37–44. [CrossRef]
4. Zhu, J.; Wang, X.; Wang, P.; Wu, Z.; Kim, M.J. Integration of BIM and GIS: Geometry from IFC to Shapefile Using Open-Source Technology. *Autom. Constr.* **2019**, *102*, 105–119. [CrossRef]
5. Wang, T.; Wang, J.; Wu, P.; Wang, J.; He, Q.; Wang, X. Estimating the environmental costs and benefits of demolition waste using life cycle assessment and willingness-to-pay: A case study in Shenzhen. *J. Clean. Prod.* **2018**, *172*, 14–26. [CrossRef]
6. Zhang, C.; Gholipour, G.; Mousavi, A.A. Nonlinear dynamic behavior of simply-supported RC beams subjected to combined impact-blast loading. *Eng. Struct.* **2019**, *181*, 124–142. [CrossRef]
7. Mou, B.; Li, X.; Bai, Y.; Wang, L. Shear Behavior of Panel Zones in Steel Beam-to-Column Connections with Unequal Depth of Outer Annular Stiffener. *J. Struct. Eng.* **2019**, *145*, 04018247. [CrossRef]
8. Deshmukh, M.; Mahatme, C. Generative Design: A New Intelligent Design & Manufacturing Approach. *Chemik* **2020**, *2*, 193–198.
9. Abedini, M.; Zhang, C. Performance Assessment of Concrete and Steel Material Models in LS-DYNA for Enhanced Numerical Simulation, A State of the Art Review. *Arch. Comput. Methods Eng.* **2021**, *28*, 2921–2942. [CrossRef]
10. Alam, Z.; Sun, L.; Zhang, C.; Su, Z.; Samali, B. Experimental and numerical investigation on the complex behaviour of the localised seismic response in a multi-storey plan-asymmetric structure. *Struct. Infrastruct. Eng.* **2021**, *17*, 86–102. [CrossRef]
11. Zenggang, X.; Zhiwen, T.; Xiaowen, C.; Xue-Min, Z.; Kaibin, Z.; Conghuan, Y. Research on Image Retrieval Algorithm Based on Combination of Color and Shape Features. *J. Signal. Process. Syst.* **2021**, *93*, 139–146. [CrossRef]
12. Toth, B.; Janssen, P.; Stouffs, R.; Chaszar, A.; Boeykens, S. Custom Digital Workflows: A New Framework for Design Analysis Integration. *Int. J. Arch. Comput.* **2012**, *10*, 481–499. [CrossRef]
13. Liu, J.; Liu, Y.; Wang, X. An environmental assessment model of construction and demolition waste based on system dynamics: A case study in Guangzhou. *Environ. Sci. Pollut. Res.* **2020**, *27*, 37237–37259. [CrossRef]
14. Gao, N.; Tang, L.; Deng, J.; Lu, K.; Hou, H.; Chen, K. Design, fabrication and sound absorption test of composite porous metamaterial with embedding I-plates into porous polyurethane sponge. *Appl. Acoust.* **2021**, *175*, 107845. [CrossRef]
15. He, L.; Shao, F.; Ren, L. Sustainability appraisal of desired contaminated groundwater remediation strategies: An information-entropy-based stochastic multi-criteria preference model. *Environ. Dev. Sustain.* **2021**, *23*, 1759–1779. [CrossRef]
16. Song, Y.; Wang, X.; Tan, Y.; Wu, P.; Sutrisna, M.; Cheng, J.C.P.; Hampson, K. Trends and Opportunities of BIM-GIS Integration in the Architecture, Engineering and Construction Industry: A Review from a Spatio-Temporal Statistical Perspective. *ISPRS Int. J. Geo. Inf.* **2017**, *6*, 397. [CrossRef]
17. Leitão, A.; Santos, L.; Lopes, J.D.A. Programming Languages for Generative Design: A Comparative Study. *Int. J. Arch. Comput.* **2012**, *10*, 139–162. [CrossRef]
18. Chen, Y.; Li, J.; Lu, H.; Yan, P. Coupling system dynamics analysis and risk aversion programming for optimizing the mixed noise-driven shale gas-water supply chains. *J. Clean. Prod.* **2021**, *278*, 123209. [CrossRef]
19. Zhu, L.; Kong, L.; Zhang, C. Numerical Study on Hysteretic Behaviour of Horizontal-Connection and Energy-Dissipation Structures Developed for Prefabricated Shear Walls. *Appl. Sci.* **2020**, *10*, 1240. [CrossRef]
20. Ma, H.-J.; Xu, L.-X.; Yang, G.-H. Multiple Environment Integral Reinforcement Learning-Based Fault-Tolerant Control for Affine Nonlinear Systems. *IEEE Trans. Cybern.* **2021**, *51*, 1913–1928. [CrossRef] [PubMed]
21. Liu, Y.; Zhang, B.; Feng, Y.; Lv, X.; Ji, D.; Niu, Z.; Yang, Y.; Zhao, X.; Fan, Y. Development of 340-GHz Transceiver Front End Based on GaAs Monolithic Integration Technology for THz Active Imaging Array. *Appl. Sci.* **2020**, *10*, 7924. [CrossRef]
22. Li, B.-H.; Liu, Y.; Zhang, A.-M.; Wang, W.-H.; Wan, S. A Survey on Blocking Technology of Entity Resolution. *J. Comput. Sci. Technol.* **2020**, *35*, 769–793. [CrossRef]
23. Shea, K.; Aish, R.; Gourtovaia, M. Towards integrated performance-driven generative design tools. *Autom. Constr.* **2005**, *14*, 253–264. [CrossRef]
24. Wu, C.; Wang, X.; Chen, M.; Kim, M.J. Differential Received Signal Strength Based RFID Positioning for Construction Equipment Tracking. *Adv. Eng. Inform.* **2019**, *42*, 100960. [CrossRef]
25. Computational Design Software, Buildings-GenerativeComponents. Available online: https://www.bentley.com/en/products/product-line/modeling-and-visualization-software/generativecomponents (accessed on 21 January 2021).
26. Zhu, J.; Wang, X.; Chen, M.; Wu, P.; Kim, M.J. Integration of BIM and GIS: IFC Geometry Transformation to Shapefile Using Enhanced Open-Source Approach. *Autom. Constr.* **2019**, *106*, 102859. [CrossRef]
27. Frazer, J. Creative Design and the Generative Evolutionary Paradigm. In *Creative Evolutionary Systems*; Elsevier BV: Amsterdam, The Netherlands, 2002; pp. 253–274.
28. Luisa, C.; José, D. Implementation Issues in Generative Design Systems, First International Conference on Design Computing and Cognition. 2004. Available online: http://home.fa.utl.pt/~jduarte/dcc08_workshop/2004_Proceedings_Workshop_3_DCC04.pdf (accessed on 21 January 2021).

29. BuHamdan, S.; Alwisy, A.; Bouferguene, A. Generative systems in the architecture, engineering and construction industry: A systematic review and analysis. *Int. J. Arch. Comput.* **2020**. [CrossRef]
30. Tsai, Y.-H.; Wang, J.; Chien, W.-T.; Wei, C.-Y.; Wang, X.; Hsieh, S.-H. A BIM-Based Approach for Predicting Corrosion under Insulation. *Autom. Constr.* **2019**, *107*, 102923. [CrossRef]
31. Monizza, G.P.; Bendetti, C.; Matt, D.T. Parametric and Generative Design techniques in mass-production environments as effective enablers of Industry 4.0 approaches in the Building Industry. *Autom. Constr.* **2018**, *92*, 270–285. [CrossRef]
32. Touloupaki, E.; Theodosiou, T. Energy Performance Optimization as a Generative Design Tool for Nearly Zero Energy Buildings. *Procedia Eng.* **2017**, *180*, 1178–1185. [CrossRef]
33. Nagy, D.; Lau, D.; Locke, J.; Stoddart, J.; Villaggi, L.; Wang, R.; Zhao, D.; Benjamin, D. Project Discover: An Application of Generative Design for Architectural Space Planning. In Proceedings of the Symposium on Simulation for Architecture and Urban Design, Toronto, ON, Canada, 22–24 May 2017; Society for Modeling and Simulation International (SCS): San Diego, CA, USA; New York, NY, USA, 2017; p. 7.
34. Singh, V.; Gu, N. Towards an integrated generative design framework. *Des. Stud.* **2012**, *33*, 185–207. [CrossRef]
35. Zheng, J.; Zhang, C.; Li, A. Experimental Investigation on the Mechanical Properties of Curved Metallic Plate Dampers. *Appl. Sci.* **2019**, *10*, 269. [CrossRef]
36. Zhang, C.; Wang, H. Robustness of the Active Rotary Inertia Driver System for Structural Swing Vibration Control Subjected to Multi-Type Hazard Excitations. *Appl. Sci.* **2019**, *9*, 4391. [CrossRef]
37. Gholipour, G.; Zhang, C.; Mousavi, A.A. Nonlinear numerical analysis and progressive damage assessment of a cable-stayed bridge pier subjected to ship collision. *Mar. Struct.* **2020**, *69*, 102662. [CrossRef]
38. Alam, Z.; Zhang, C.; Samali, B. The role of viscoelastic damping on retrofitting seismic performance of asymmetric reinforced concrete structures. *Earthq. Eng. Eng. Vib.* **2020**, *19*, 223–237. [CrossRef]
39. Krish, S. A practical generative design method. *Comput. Des.* **2011**, *43*, 88–100. [CrossRef]
40. Marsh, A. Generative and Performative Design: A Challenging New Role for Modern Architects. In *The Oxford Conference 2008*; WIT Press: Oxford, UK, 2008; p. 5.
41. Abedini, M.; Mutalib, A.A.; Zhang, C.; Mehrmashhadi, J.; Raman, S.N.; Alipour, R.; Momeni, T.; Mussa, M.H. Large deflection behavior effect in reinforced concrete columns exposed to extreme dynamic loads. *Front. Struct. Civ. Eng.* **2020**, *14*, 532–553. [CrossRef]
42. Medjdoub, B.; Chenini, M.B. A constraint-based parametric model to support building services design exploration. *Arch. Eng. Des. Manag.* **2013**, *11*, 123–136. [CrossRef]
43. Buonamici, F.; Carfagni, M.; Furferi, R.; Volpe, Y.; Governi, L. Generative Design: An Explorative Study. *Comput. Des. Appl.* **2020**, *18*, 144–155. [CrossRef]
44. Sun, L.; Yang, Z.; Jin, Q.; Yan, W. Effect of Axial Compression Ratio on Seismic Behavior of GFRP Reinforced Concrete Columns. *Int. J. Struct. Stab. Dyn.* **2020**, *20*, 2040004. [CrossRef]
45. Wu, C.; Wu, P.; Wang, J.; Jiang, R.; Chen, M.; Wang, X. Ontological Knowledge Base for Concrete Bridge Rehabilitation Project Management. *Autom. Constr.* **2021**, *121*, 103428. [CrossRef]
46. Gholipour, G.; Zhang, C.; Mousavi, A.A. Numerical analysis of axially loaded RC columns subjected to the combination of impact and blast loads. *Eng. Struct.* **2020**, *219*, 110924. [CrossRef]
47. Abedini, M.; Zhang, C.; Mehrmashhadi, J.; Akhlaghi, E. Comparison of ALE, LBE and pressure time history methods to evaluate extreme loading effects in RC column. *Structures* **2020**, *28*, 456–466. [CrossRef]
48. Alam, Z.; Zhang, C.; Samali, B. Influence of seismic incident angle on response uncertainty and structural performance of tall asymmetric structure. *Struct. Des. Tall Spéc. Build.* **2020**, *29*, 1750. [CrossRef]
49. Azhar, S.; Khalfan, M.; Maqsood, T. Building information modelling (BIM): Now and beyond. *Constr. Econ. Build.* **2015**, *12*, 15–28. [CrossRef]
50. Zhu, J.; Wu, P.; Chen, M.; Kim, M.J.; Wang, X.; Fang, T. Automatically Processing IFC Clipping Representation for BIM and GIS Integration at the Process Level. *Appl. Sci.* **2020**, *10*, 2009. [CrossRef]
51. Hardin, B. *BIM and Construction Management: Proven Tools, Methods, and Workflows*, 1st ed.; Wiley: New York, NY, USA, 2009.
52. Ju, Y.; Shen, T.; Wang, D. Bonding behavior between reactive powder concrete and normal strength concrete. *Constr. Build. Mater.* **2020**, *242*, 118024. [CrossRef]
53. Choi, J.; Lee, S.; Kim, I. Development of Quality Control Requirements for Improving the Quality of Architectural Design Based on BIM. *Appl. Sci.* **2020**, *10*, 7074. [CrossRef]
54. Ariza-López, F.J.; Rodríguez-Avi, J.; Reinoso-Gordo, J.F.; Ariza-López, Í.A.; López, A. Quality Control of "As Built" BIM Datasets Using the ISO 19157 Framework and a Multiple Hypothesis Testing Method Based on Proportions. *ISPRS Int. J. Geo. Inf.* **2019**, *8*, 569. [CrossRef]
55. Zhang, C.; Abedini, M.; Mehrmashhadi, J. Development of pressure-impulse models and residual capacity assessment of RC columns using high fidelity Arbitrary Lagrangian-Eulerian simulation. *Eng. Struct.* **2020**, *224*, 111219. [CrossRef]
56. Zhang, W.; Tang, Z.; Yang, Y.; Wei, J. Assessment of FRP–Concrete Interfacial Debonding with Coupled Mixed-Mode Cohesive Zone Model. *J. Compos. Constr.* **2021**, *25*, 04021002. [CrossRef]
57. Zhang, C.; Wang, H. Swing vibration control of suspended structures using the Active Rotary Inertia Driver system: Theoretical modeling and experimental verification. *Struct. Control. Heal. Monit.* **2020**, *27*, 2543. [CrossRef]

58. Sydora, C.; Stroulia, E. Rule-based compliance checking and generative design for building interiors using BIM. *Autom. Constr.* **2020**, *120*, 103368. [CrossRef]
59. Mason, M. Extending BIM Design Value Using the Revit Api | AUGI-The World's Largest CAD & BIM User Group. 2009. Available online: https://www.augi.com/articles/detail/extending-bim-design-value-using-the-revit-api (accessed on 1 February 2021).
60. Feist, S.; Barreto, G.; Ferreira, B.; Leitão, A. Portable generative design for building information modelling. In *Living Systems and Micro-Utopias: Towards Continuous Designing*; The Association for Computer-Aided Architectural Design Research in Asia: Hong Kong, China, 2016; p. 10.
61. Huang, H.; Huang, M.; Zhang, W.; Pospisil, S.; Wu, T. Experimental Investigation on Rehabilitation of Corroded RC Columns with BSP and HPFL under Combined Loadings. *J. Struct. Eng.* **2020**, *146*, 04020157. [CrossRef]
62. Huang, H.; Guo, M.; Zhang, W.; Zeng, J.; Yang, K.; Bai, H. Numerical investigation on the bearing capacity of RC columns strengthened by HPFL-BSP under combined loadings. *J. Build. Eng.* **2021**, *39*, 102266. [CrossRef]
63. Huang, H.; Huang, M.; Zhang, W.; Yang, S. Experimental study of predamaged columns strengthened by HPFL and BSP under combined load cases. *Struct. Infrastruct. Eng.* **2021**, *17*, 1210–1227. [CrossRef]
64. David, S. Generative Design in Revit Now Available, Revit Official Blog. 2020. Available online: https://blogs.autodesk.com/revit/2020/04/08/generative-design-in-revit/ (accessed on 12 June 2020).
65. Wu, C.; Wu, P.; Wang, J.; Jiang, R.; Chen, M.; Wang, X. Critical Review of Data-Driven Decision-Making in Bridge Operation and Maintenance. *Struct. Infrastruct. Eng.* **2020**, 1–24. [CrossRef]
66. Zhang, C.; Gholipour, G.; Mousavi, A.A. Blast loads induced responses of RC structural members: State-of-the-art review. *Compos. Part. B: Eng.* **2020**, *195*, 108066. [CrossRef]
67. Gao, N.; Wang, B.; Lu, K.; Hou, H. Complex band structure and evanescent Bloch wave propagation of periodic nested acoustic black hole phononic structure. *Appl. Acoust.* **2021**, *177*, 107906. [CrossRef]
68. Li, A.; Spano, D.; Krivochiza, J.; Domouchtsidis, S.; Tsinos, C.G.; Masouros, C.; Chatzinotas, S.; Li, Y.; Vucetic, B.; Ottersten, B. A Tutorial on Interference Exploitation via Symbol-Level Precoding: Overview, State-of-the-Art and Future Directions. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 796–839. [CrossRef]
69. Welzenbach, R. Research Guides: Research Impact Metrics: Citation Analysis: Scopus. Available online: https://guides.lib.umich.edu/citation/Scopus (accessed on 8 October 2020).
70. Elsevier. Content—How Scopus Works. Available online: https://www.elsevier.com/solutions/scopus/how-scopus-works/content (accessed on 25 November 2020).
71. Caetano, I.; Santos, L.; Leitão, A. Computational design in architecture: Defining parametric, generative, and algorithmic design. *Front. Arch. Res.* **2020**, *9*, 287–300. [CrossRef]
72. Zhang, C.; Wang, H. Swing Vibration Control of Suspended Structure Using Active Rotary Inertia Driver System: Parametric Analysis and Experimental Verification. *Appl. Sci.* **2019**, *9*, 3144. [CrossRef]
73. Zhu, J.; Yang, K.; Chen, Y.; Fan, G.; Zhang, L.; Guo, B.; Guan, X.; Zhao, R. Revealing the substitution preference of zinc in ordinary Portland cement clinker phases: A study from experiments and DFT calculations. *J. Hazard. Mater.* **2021**, *409*, 124504. [CrossRef]
74. Zhu, J.; Chen, Y.; Zhang, L.; Guo, B.; Fan, G.; Guan, X.; Zhao, R. Revealing the doping mechanism of barium in sulfoaluminate cement clinker phases. *J. Clean. Prod.* **2021**, *295*, 126405. [CrossRef]
75. Roovers, K.; Raucroix, X.; Wyns, K.; Meerkerk, A.; van Steirteghem, J. Design and production automation for the A16 tunnel in Rotterdam. In *IABSE Congress, New York, New York 2019: The Evolving Metropolis*; International Association for Bridge and Structural Engineering (IABSE): Zürich, Switzerland, 2019; pp. 1844–1850.
76. Ma, C.; Zhu, C.; Xiang, K. Digital Aided Façade Design Introduced in A Traditional Design Workflow. In Proceedings of the 24th CAADRIA Conference, Wellington, New Zealand, 15–18 April 2019; pp. 675–684.
77. Castro, A.B.; Bío, U.D.B.; Alvarado, R.G. BIM-Integration of solar thermal systems in early housing design. *Rev. Constr.* **2017**, *16*, 323–338. [CrossRef]
78. Yoo, B.; Yoon, H.; Kim, Y.; Lee, K.M. Stepwise Application of BIM-based Parametric Modeling to Tapered Slip-Form System. *Procedia Eng.* **2016**, *145*, 112–119. [CrossRef]
79. Xia, J.; Peng, M. The parametric design of Shanghai Tower's form and façade. In Proceedings of the CTBUH 2012 9th World Congress, Shanghai, China, 19–21 September 2012.
80. Akkoyunlu, T. Parametric BIM Façade Module Development for Diagrid Twisted Structures. In Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC), International Association for Automation and Robotics in Construction (IAARC), Berlin, Germany, 20–25 July 2018; pp. 1056–1061.
81. Wortmann, T.; Tuncer, B. Differentiating parametric design: Digital workflows in contemporary architecture and construction. *Des. Stud.* **2017**, *52*, 173–197. [CrossRef]
82. Caetano, I.; Leitão, A. Integration of an algorithmic BIM approach in a traditional architecture studio. *J. Comput. Des. Eng.* **2018**, *6*, 327–336. [CrossRef]
83. Veloso, P.; Celani, G.; Scheeren, R. From the generation of layouts to the production of construction documents: An application in the customization of apartment plans. *Autom. Constr.* **2018**, *96*, 224–235. [CrossRef]
84. Abrishami, S.; Goulding, J.; Rahimian, F. Generative BIM workspace for AEC conceptual design automation: Prototype development. *Eng. Constr. Arch. Manag.* **2020**, *28*, 482–509. [CrossRef]

85. Sydora, C.; Stroulia, E. Generative Interior Design using BIM. In Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, Association for Computing Machinery (ACM), New York, NY, USA, 13–14 November 2019; pp. 354–355.
86. Tafraout, S.; Bourahla, N.; Mebarki, A. Automatic structural design of RC wall-slab buildings using a genetic algorithm with application in BIM environment. *Autom. Constr.* **2019**, *106*, 102901. [CrossRef]
87. Bianconi, F.; Filippucci, M.; Buffi, A. Automated design and modeling for mass-customized housing. A web-based design space catalog for timber structures. *Autom. Constr.* **2019**, *103*, 13–25. [CrossRef]
88. Banihashemi, S.; Tabadkani, A.; Hosseini, M.R. Integration of parametric design into modular coordination: A construction waste reduction workflow. *Autom. Constr.* **2018**, *88*, 1–12. [CrossRef]
89. Stumm, S.; Neu, P.; Brell-Cokcan, P.N.A.S. Towards Cloud Informed Robotics. In Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC); International Association for Automation and Robotics in Construction (IAARC), Taipei, Taiwan, 28 June–1 July 2017; pp. 59–64.
90. Khalili-Araghi, S.; Kolarevic, B. Development of a framework for dimensional customization system: A novel method for customer participation. *J. Build. Eng.* **2016**, *5*, 231–238. [CrossRef]
91. Chen, J.-Y.; Huang, S.-C. Adaptive Building Facade Optimisation. In Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2016, Melbourne, Australia, 30 March–2 April 2016; pp. 259–268.
92. Abrishami, S.; Goulding, J.; Rahimian, F.; Ganah, A. Virtual generative BIM workspace for maximising AEC conceptual design innovation. *Constr. Innov.* **2015**, *15*, 24–41. [CrossRef]
93. Abrishami, S.; Goulding, J.; Pour Rahimian, F.; Ganah, A. Integration of BIM and generative design to exploit AEC conceptual design innovation. *Electron. J. Inf. Technol. Constr.* **2014**, *19*, 350–359.
94. Abrishami, S.; Goulding, J.; Pour Rahimian, F.; Ganah, A.; Sawhney, A. G-BIM Framework and Development Process for Integrated AEC Design Automation. *Procedia Eng.* **2014**, *85*, 10–17. [CrossRef]
95. Afsari, K.; Swarts, M.E.; Gentry, T.R. Integrated Generative Technique for Interactive Design of Brickworks. *J. Inf. Technol. Constr.* **2014**, *19*, 225–247.
96. Fernando, R.; Drogemuller, R. Parametric and Generative Methods with Building Information Modelling: Connecting BIM with explorative design modelling. In Proceedings of the Beyond Codes and Pixels, Hong Kong, China, 4 April 2012; pp. 537–546.
97. Wang, J.; Li, J.; Chen, X. Parametric Design Based on Building Information Modeling for Sustainable Buildings. In Proceedings of the 2010 International Conference on Challenges in Environmental Science and Computer Engineering; Institute of Electrical and Electronics Engineers (IEEE), Wuhan, China, 6–7 March 2010; Volume 2, pp. 236–239.
98. Leitão, A.; Castelo-Branco, R.; Cardoso, C. Algorithmic-Based Analysis-Design and Analysis in a Multi Back-end Generative Tool. In Proceedings of the 22nd CAADRIA Conference, Xi'an Jiaotong-Liverpool University, Suzhou, China, 5–8 April 2017; pp. 137–146.
99. Sabra, J.B.; Mullins, M.F. Design of parametric software tools: Optimizing future health care performance by integrating evidence-based knowledge in architectural design and building processes. *Light Eng. Archit. Environ.* **2011**, *121*, 37–49.
100. González-Márquez, R.J.; Esparza, R.M. Tool Design as a Strategy for Architectural Design. In Proceedings of the Presentation for the XXXVII IAHS World Congress on Housing, Santander, Spain, 26–29 October 2010; p. 9.
101. Noone, M.; Mooney, A. Visual and textual programming languages: A systematic review of the literature. *J. Comput. Educ.* **2018**, *5*, 149–174. [CrossRef]
102. Ferreira, B.; Leitão, A. Generative Design for Building Information Modeling. In Proceedings of the 33rd eCAADe, Vienna, Austria, 16–18 September 2015; pp. 635–644.
103. Loomis, M. Rhino Grasshopper VS Generative Components, Designplaygrounds. 2011. Available online: http://designplaygrounds.com/deviants/rhino-grasshopper-vs-generative-components/ (accessed on 13 February 2021).
104. Boshernitsan, M.; Downes, M. *Visual Programming Languages: A Survey*; University of California: Berkley, CA, USA, 2004.
105. An Overview of Generative Components-Generative Components Community Wiki-Generative Components-Bentley Communities. Available online: https://communities.bentley.com/products/products_generativecomponents/w/generative_components_community_wiki (accessed on 15 February 2021).
106. Leitão, A.; Cabecinhas, F.; Martins, S. Revisiting the Architecture Curriculum: The programming perspective. In Proceedings of the 28th eCAADe, Zurich, Switzerland, 15–18 September 2010; pp. 81–88.
107. Zhao, R.; Zhang, L.; Fan, G.; Chen, Y.; Huang, G.; Zhang, H.; Zhu, J.; Guan, X. Probing the exact form and doping preference of magnesium in ordinary Portland cement clinker phases: A study from experiments and DFT simulations. *Cem. Concr. Res.* **2021**, *144*, 106420. [CrossRef]
108. CHEN, V. A History of Visual Programming: From Basic to Bubble, Bubble Blog-The Best Way to Build Web Apps without Code. 2020. Available online: https://bubble.io/blog/visual-programming/ (accessed on 20 February 2021).
109. Leitão, A.; Caetano, I.; Correia, H. Processing architecture. *Int. J. Arch. Comput.* **2016**, *14*, 147–157. [CrossRef]
110. Xu, S.; Wang, J.; Shou, W.; Ngo, T.; Sadick, A.-M.; Wang, X. Computer Vision Techniques in Construction: A Critical Review. *Arch. Comput. Methods Eng.* **2021**, *28*, 3383–3397. [CrossRef]
111. Sun, L.; Li, C.; Zhang, C.; Liang, T.; Zhao, Z. The Strain Transfer Mechanism of Fiber Bragg Grating Sensor for Extra Large Strain Monitoring. *Sensors* **2019**, *19*, 1851. [CrossRef]

112. Li, C.; Sun, L.; Xu, Z.; Wu, X.; Liang, T.; Shi, W. Experimental Investigation and Error Analysis of High Precision FBG Displacement Sensor for Structural Health Monitoring. *Int. J. Struct. Stab. Dyn.* **2020**, *20*, 2040011. [CrossRef]
113. Ma, H.-J.; Xu, L.-X. Decentralized Adaptive Fault-Tolerant Control for a Class of Strong Interconnected Nonlinear Systems via Graph Theory. *IEEE Trans. Autom. Control.* **2021**, *66*, 3227–3234. [CrossRef]
114. Ma, G.; Sun, J.; Wang, L.; Aslani, F.; Liu, M. Electromagnetic and microwave absorbing properties of cementitious composite for 3D printing containing waste copper solids. *Cem. Concr. Compos.* **2018**, *94*, 215–225. [CrossRef]
115. Ma, G.; Sun, J.; Aslani, F.; Huang, Y.; Jiao, F. Review on electromagnetic wave absorbing capacity improvement of cementitious material. *Constr. Build. Mater.* **2020**, *262*, 120907. [CrossRef]
116. Ma, G.; Zhang, J.; Wang, L.; Li, Z.; Sun, J. Mechanical characterization of 3D printed anisotropic cementitious material by the electromechanical transducer. *Smart Mater. Struct.* **2018**, *27*, 075036. [CrossRef]
117. Sun, J.; Huang, Y.; Aslani, F.; Ma, G. Properties of a double-layer EMW-absorbing structure containing a graded nano-sized absorbent combing extruded and sprayed 3D printing. *Constr. Build. Mater.* **2020**, *261*, 120031. [CrossRef]
118. Sun, J.; Huang, Y.; Aslani, F.; Ma, G. Electromagnetic wave absorbing performance of 3D printed wave-shape copper solid cementitious element. *Cem. Concr. Compos.* **2020**, *114*, 103789. [CrossRef]
119. Ma, H.-J.; Yang, G.-H. Adaptive Fault Tolerant Control of Cooperative Heterogeneous Systems with Actuator Faults and Unreliable Interconnections. *IEEE Trans. Autom. Control.* **2016**, *61*, 3240–3255. [CrossRef]
120. Sun, J.; Huang, Y.; Aslani, F.; Wang, X.; Ma, G. Mechanical enhancement for EMW-absorbing cementitious material using 3D concrete printing. *J. Build. Eng.* **2021**, *41*, 102763. [CrossRef]
121. Sun, J.; Aslani, F.; Wei, J.; Wang, X. Electromagnetic absorption of copper fiber oriented composite using 3D printing. *Constr. Build. Mater.* **2021**, *300*, 124026. [CrossRef]
122. Aslani, F.; Sun, J.; Bromley, D.; Ma, G. Fiber-reinforced lightweight self-compacting concrete incorporating scoria aggregates at elevated temperatures. *Struct. Concr.* **2019**, *20*, 1022–1035. [CrossRef]
123. Aslani, F.; Hou, L.; Nejadi, S.; Sun, J.; Abbasi, S. Experimental analysis of fiber-reinforced recycled aggregate self-compacting concrete using waste recycled concrete aggregates, polypropylene, and steel fibers. *Struct. Concr.* **2019**, *20*, 1670–1683. [CrossRef]
124. Sun, J.; Lin, S.; Zhang, G.; Sun, Y.; Zhang, J.; Chen, C.; Morsy, A.M.; Wang, X. The effect of graphite and slag on electrical and mechanical properties of electrically conductive cementitious composites. *Constr. Build. Mater.* **2021**, *281*, 122606. [CrossRef]
125. Sun, J.; Ma, Y.; Li, J.; Zhang, J.; Ren, Z.; Wang, X. Machine learning-aided design and prediction of cementitious composites containing graphite and slag powder. *J. Build. Eng.* **2021**, *43*, 102544. [CrossRef]
126. Hou, L.; Wu, S.; Zhang, G.; Tan, Y.; Wang, X. Literature Review of Digital Twins Applications in Construction Workforce Safety. *Appl. Sci.* **2021**, *11*, 339. [CrossRef]
127. Sun, J.; Aslani, F.; Lu, J.; Wang, L.; Huang, Y.; Ma, G. Fresh and mechanical behaviour of developed fibre-reinforced lightweight engineered cementitious composites for 3D concrete printing containing hollow glass microspheres. *Ceram. Int.* **2021**. [CrossRef]
128. Sun, J.; Wang, Y.; Yao, X.; Ren, Z.; Zhang, G.; Zhang, C.; Chen, X.; Ma, W.; Wang, X. Machine-Learning-Aided Prediction of Flexural Strength and ASR Expansion for Waste Glass Cementitious Composite. *Appl. Sci.* **2021**, *11*, 6686. [CrossRef]