AI and Ethics (2021) 1:273–281
https://doi.org/10.1007/s43681-020-00034-z

ORIGINAL RESEARCH

Moral responsibility for computationally designed products

David M. Douglas[1] · David Howard[2] · Justine Lacey[1]

Received: 18 September 2020 / Accepted: 3 December 2020 / Published online: 20 January 2021

Abstract
Computational design systems (such as those using evolutionary algorithms) can create designs for a variety of physical products. Introducing these systems into the design process risks creating a 'responsibility gap' for flaws in the products they are used to create, as human designers may no longer believe that they are wholly responsible for them. We respond to this problem by distinguishing between causal responsibility and capacity responsibility (the ability to be morally responsible for actions) for creating product designs to argue that while the computational design systems and human designers are both casually responsible for creating product designs, the human designers who use these systems and the developers who create them have capacity responsibility for such designs. We show that there is no responsibility gap for products designed using computational design systems by comparing different accounts of moral responsibility for robots and AI (instrumentalism, machine ethics, and hybrid responsibility). We argue that all three of these accounts of moral responsibility for AI systems support the conclusion that the product designers who use computational design systems and the developers of these systems are morally responsible for any flaws or faults in the products designed by these systems. We conclude by showing how the responsibilities of accountability and blameworthiness should be attributed between the product designers, the developers of the computational design systems.

Keywords Computational design · Moral responsibility · Machine ethics · Generative design · Parametric design · Artificial intelligence · Evolutionary algorithms

1 Introduction

Artificial intelligence (AI) offers many opportunities for automating manufacturing processes. While industrial robots have already assisted or replaced human workers in a variety of manufacturing tasks, AI is increasingly employed in product design and development. This application of AI goes under several names, including computational design, generative design, and parametric design [4]. Combining computational design and 3D printing (or additive manufacturing) creates new possibilities for designing and producing physical products without necessarily involving human designers or manufacturers in the process. A major application of computational design is topology optimisation, where elements are removed from a design to optimise it towards certain criteria (such as reducing weight) while maintaining other characteristics (such as strength) [35]. Topology optimisation and 3D printing methods have been applied to design aircraft components and car parts that are considerably lighter than existing parts with equivalent strength [16].

Computational design is an application of AI that is currently underrepresented in the AI ethics literature. Replacing or augmenting human designers with computational design challenges our existing notions of responsibility for design. While it might appear straightforward that the responsibilities of human designers are unchanged if they use computational design, this conclusion should not automatically be accepted. The question of responsibility (in terms of attributing authorship) for designs created by AI systems has already been raised in the context of computer-generated art [11, 24, 31]. Advances in computational design and the

✉ David M. Douglas
David.Douglas@csiro.au

David Howard
David.Howard@data61.csiro.au

Justine Lacey
Justine.Lacey@csiro.au

[1] Responsible Innovation Future Science Platform, CSIRO, Brisbane, Australia

[2] Robotics and Autonomous Systems Group, CSIRO, Brisbane, Australia

use of the products created using this technology have the potential to create significant societal impact, both positive and otherwise [32]. This has sharpened our focus on the nature of responsibility in the science, technology, and innovation process itself [29], and led to questions about new distributions of responsibility, particularly in the case of system or design failure [10]. The responsibility for the outcome of a machine's operation is usually ascribed to its operator if the machine performs as expected, or to its manufacturer if the machine malfunctions [23]. Similarly, design flaws in a machine are attributed to its designers. Computational design disturbs these established norms by replacing the human designer's role in making decisions about aspects of the design. This uncertainty creates a risk that the designers using these systems may believe that they are not wholly responsible for the products they create [17]. Designers may see the computational system itself as having full or even partial responsibility for the designed products. This uncertainty may lead to the belief that no specific person is responsible for the design, creating a 'responsibility gap' [23]. Clarifying whether the users of a computational design system, the system's developers, or the system itself are responsible for products created using this approach will remove this uncertainty.

Our focus here is on moral responsibility which concerns whether someone should receive moral praise or condemnation for their action or inaction [12]. Questions of legal responsibility (such as liability) for computationally designed products are beyond our scope.

This paper begins with an overview of computational design and describes one method (evolutionary algorithms) in detail. It then describes how the major positions on responsibility for the output of AI systems (instrumentalism, machine ethics, and hybrid responsibility) all converge on attributing moral responsibility to the human users and developers of computational design systems. No responsibility gap therefore exists if computational design systems are used to create physical products. Furthermore, this paper goes on to explain how the responsibilities of accountability and blameworthiness for design flaws in the products created using computational design may be attributed between the users and developers of these systems, such as computational design systems.

2 Computational, parametric, and generative design

Product design traditionally involved drawing sketches and detailed plans of design concepts on paper and creating physical models or prototypes that go through many iterations until the designer is satisfied. Computer-aided design (CAD) (or digital design) introduced computer technology into the design process, either as a replacement for the designer's sketchpad or drawing board, or as a method of simulating the physical characteristics of designs before they are physically created. Broadly, computational design involves computer technology assisting or replacing human designers in creating and modifying design concepts.

In the context of architecture, Caetano et al. [4] describe computational design as a design process that harnesses computation through automating design procedures, performing design tasks in parallel, adding and reflecting design changes quickly in representations of the design, and using automated feedback to assist designers in form-finding processes. These characteristics also apply to computational design in other contexts, such as art or product design.

Specific forms of computational design are parametric design and generative design. Parametric design uses physically meaningful parameters, often provided by a domain expert. Varying those parameters within reasonable (expert-set) ranges creates new designs in a comparatively constrained process. Parametric designs have the benefit of being human-understandable. The meaning of design characteristics can be gleaned from examining the optimised final parameters as they have human meaning, such as width, which allow human designers to conceptualise the extents of the parametric design space and operate effectively within it. Parameters may be varied interactively by hand, and modern design software reflects parameter changes instantly in a visualisation of the design to allow for experimentation. Many design packages also come with inbuilt parametric optimisers, which operate on a software representation of the design (a number string with one number per design parameter), and run without human input to maximise one of more fitness functions over a number of iterations, that calculates how well the design performs its intended purpose [19]. As these optimisers are only guided at a high level by human-defined performance goals, they have some capacity to provide surprising and non-intuitive solutions (parameter combinations), albeit within a restricted parametric design space.

Generative design generates a plethora of solutions over numerous iterations. Unlike parametric designs, there is a level of indirection between the software representation of the design (the number string), and the complexity of the final design. Generative design typically harnesses complexity-generating measures including symmetries, repetition, and scaling, to produce complex designs from a comparatively small number string. Generative designs provide more complex, intricate designs with a smaller number string, and the string does not necessarily have to grow if an increase in design complexity is required. Unlike parametric design, these numbers have no physically grounded meaning—they may, for example, be the weights of a neural network that is

run to produce an intricate geometry [33]. Other implementations include graphs and shape grammars [20].

Generative design spaces are often unintuitive to navigate for human designers, necessitating the use of computational optimisation to find good designs. Modern generative design is implemented in software and run in parallel on computers. In generative design, users input some high-level design goals (e.g., a fitness function), and the system attempts to generate designs that meet all those goals in the best way possible (maximise fitness across all objectives). Because of the relationship between number string and final design, and the lack of physically grounded parameters, the design process is much more free-form, with the algorithm searching a wider space than parametric design methods.

3 Evolutionary algorithms in design

Evolutionary algorithms are one of the main approaches to machine learning [8]. As black box optimisers, they are popular for design problems which may be high-dimensional, multi-modal, or lack direct gradient information. This is particularly true for generative representations. Evolutionary algorithms apply an abstract model of evolution to find solutions to complex optimisation problems [30]. The algorithms operate within a defined search space to find solutions that meet all the specified requirements or as many of them as possible [30].

Successful examples of evolutionary design include designing spacecraft antennas [22], structural elements of construction projects [19], and even robots [3]. These designs are often unconventional as they are "not limited by conventions, aesthetic considerations, or ungrounded preferences for symmetry" [9].

While a full account of how evolutionary algorithms function is beyond the scope of this paper, here, we present a general outline of the stages these algorithms perform: initialisation, evaluation, selection, variation, and replacement [9, 19].

The initialisation stage creates a population where each member is a randomly chosen binary or numerical value or set of values that encodes a possible solution to the design problem. These values must all remain within the defined search space. The member's value is called a genotype, while the encoded solution it represents is its phenotype [9].

The evaluation stage uses a fitness function that determines the suitability of individual genotypes as solutions for the intended goal. If one or more members has a fitness function result that meets the solution's requirements, or if a certain number of iterations of the algorithm have occurred, the genotype with the highest fitness function is converted to its phenotype, which is the problem solution. The algorithm then finishes. However, if none of the members produce the required value for the fitness function or if the maximum number of iterations has yet to be reached, the algorithm continues onto the selection stage.

The selection stage chooses population members to use as a basis for creating new members. The chosen members, called parents, will usually have a high fitness function result, although some members may also be randomly chosen to ensure that some variety remains in the population [9]. The algorithm now progresses to the variation stage, where new members are created by modifying the parent members. This modification occurs via recombination and mutation. Recombination merges data from two parent members together to produce one or more child genotypes [9]. A mutation is a random change in a genotype. Mutation and recombination may be used singularly or together to produce new members.

The replacement stage involves removing either some or all the existing population with the new members created in the variation stage. If only some of the existing members are to be replaced, the survival of the existing members is determined by some form of environmental selection, which usually involves removing the members with the lowest fitness functions. Once the population has been updated, the algorithm returns to the evaluation stage, and continues again until it either completes a given number of iterations or it finds a genotype with the required fitness function. The genotype and the corresponding phenotype can be considered the decision made by the evolutionary algorithm to solve the problem it has been given.

From this description of evolutionary algorithms, we can already identify two points where human design affects the system's output:

- The selection of representation and associated bounds
- The definition of the fitness function.

As the representation defines the accessible design space, it also affects the types of designs (e.g., geometries) and level of fitness that can be achieved by a given evolutionary algorithm. Similarly, if the fitness function (mathematical equations that 'score' each design) do not capture the intended end goal, the final designs will not be fit for purpose.

Designs are frequently simulated to assess fitness, and physical simulation provides an abstraction of reality that can be informative, cheap, and easy to work with. However, simulation does not model the entirety of physics, and the potential mismatch between the simulation and the physical environment is the 'reality gap' [34], meaning that performance in the simulated and physical design is notably different. Unexpected and undesirable results from an evolutionary algorithm may also be due to 'misspecified fitness functions' (where loopholes in the fitness function allows for unintended or undesirable solutions gain high results)

or 'unintended debugging' (where software or hardware errors incorrectly provide high fitness results, while the corresponding phenotype will be ineffective or unsuitable for use) [21]. Either type of error may mean that the physical design may be dangerously flawed.

Given that the human developer designs and implements these aspects of the system, it appears straightforward that they are responsible for the flaws in the resulting physical design. Nonetheless, computational design systems will produce novel, surprising, and unexpected results [21]. The system is creating something new, and flaws in the system that cause unexpected results may only be identifiable after the system has produced them [21]. We still need to establish whether the system itself may have any moral responsibility for the solutions it produces, or whether this responsibility can only be held by humans associated with it in some way.

4 Moral responsibility and computational design

Moral responsibility for computationally designed products is a limited case of the larger question of AI or robot responsibility. While designing products via computational design does not require a robot or automated means of manufacture, it still concerns the question of whether systems that produce outputs without human direction are morally responsible for that output.

Responsibility takes many forms [6, 36]. As our question concerns the problem of whether a computational design system may be held morally responsible for products it designs, we will focus on the *capacity to be* morally responsible. This notion of responsibility may be called capacity responsibility [15]. Possessing capacity responsibility means that someone (or something) has the capability to act responsibly and to hold responsible for their actions. They are able "to reflect on the consequences of one's actions, to form intentions, to deliberately choose an action and act upon it" [36]. Capacity responsibility may also be described as fulfilling two conditions: the control condition of having the ability to control one's own actions, and the epistemic condition of being aware of one's actions and their consequences [5]. The control condition highlights the significance of causal responsibility, which is the attribution of actions, events and outcomes to people, objects, or forces of nature [15]. Capacity responsibility also creates the possibility of ascribing other forms of responsibility, such as accountability and blameworthiness, to its possessor.

The usual basis for attributing moral responsibility for what a machine does is causal responsibility: the less control the user has over the machine, the less causally responsible she is for the actions caused by its decisions. Machine learning systems, such as those used in computational design, use processing methods that the users or developers did not incorporate into it themselves [23]. The output produced by these systems involve either new rules that the system itself has developed from analysing input data, or through processes that incorporate random elements, such as evolutionary algorithms. As a result, the developers have less influence over its output that they would over a system where the processing rules are hard coded by the developers, and where randomness plays no role in the algorithmic process. This loss of influence over the system's decision-making creates uncertainty about whether the developer fulfills the epistemic condition for capacity responsibility for the machine's output. This uncertainty, combined with the lack of a clear alternative attribution of responsibility, creates the possibility of a responsibility gap where no one is responsible for the system's output [23]. Addressing the responsibility gap requires an account of how moral responsibility (i.e., capacity responsibility) should be attributed when the causal responsibility for an AI system's output is uncertain.

Gunkel [13] describes three possible responses to the responsibility gap: instrumentalism, machine ethics, and hybrid responsibility. Each response resolves the gap by proposing a different distribution of responsibility between the developers, users, and the system. Instrumentalism places capacity responsibility solely on the humans involved in the system. In contrast, machine ethics places capacity responsibility (of a sort) onto the system, while hybrid responsibility distributes capacity responsibility between the humans and the system.

Instrumentalism denies that a responsibility gap exists: responsibility for the system and its output rests on the people associated with it. It views robots and AI as products like any other, and so should not be treated differently regarding moral responsibility. For computational design systems, this view presents us with a clear response: only the people associated with the system possess capacity responsibility. The remaining difficulties are determining the relevant people associated with the system, and how different forms of responsibility should be distributed among them. If we accept that computationally designed products are like any other, then we may use the same norms for attributing accountability and blameworthiness that we would use for human-designed products as a starting point for determining how to attribute them here.

Machine ethics calls for ethical decision-making to be incorporated into robots and AI systems [1, 3]. This ability to make ethical evaluations would allow such systems to act responsibly [2]. This is a pragmatic form of attributing responsibility that is similar to the legal personhood attributed to corporations [13]. How ethical evaluations are made by the system depends on whether their evaluations are constrained by the developer to prevent harmful outcomes occurring, or if they are programmed with an ethical theory

and methods for using that theory in decision-making [25]. These may be called implicit and explicit ethical agents, respectively [25]. Computational design systems are implicit ethical agents, as they are not programmed with ethical theories and methods of ethical decision-making, but rather make evaluations within ranges set by the developer that ideally will prevent them from producing outputs with harmful consequences. When the system is an implicit ethical agent, machine ethics provides a similar response to that given by instrumentalism: the developers have capacity responsibility for the output of computational design systems.

Hybrid responsibility states that some combination of the system and the people associated with it bear responsibility [13]. There are several accounts of hybrid responsibility between people and artefacts. Hanson's [14] account of joint responsibility equates causal responsibility with moral responsibility: the responsible 'subject' is the combination of people and the physical objects necessary to perform an action. It rejects the claim that only persons can have agency, and instead builds on accounts of extended agency that incorporate both persons and the things they use to perform actions (such as tools, machines, or even animals) as being the proper subject for moral responsibility [14]. Extended agency therefore situates capacity responsibility in the collection of people and things that are causally responsible for an action.

Johnson's [18] account of hybrid responsibility describes three relevant entities for a computer system's actions: the human users, the human designers, and the computational system itself. As with Hanson's account, all three components share causal responsibility for the system's actions: the user initiated the action, the system performed it, and the developer gave the system the capability to do so [18]. However, Johnson [18] argues that computer systems lack moral agency as they do not have mental states: they fail to fulfill the epistemic condition for capacity responsibility as they are not aware of their actions or their consequences. They may have intentionality as a result of their programming (in the sense that they can act without immediate human intervention), but this intentionality reflects that of the system's users and designers. Computer systems do not have intentions independent of their human developers and users [18]. As a result, the system does not have capacity responsibility. Johnson's account therefore accepts that the combination of an artefact and the humans associated with it is causally responsible for what the artefact does, but rejects the claim that the collection of humans and artefacts shares capacity responsibility.

Gunkel's [13] concern about hybrid responsibility centres on how responsibility is divided between the entities involved. Entities within the network of responsibility may also deny that they have any responsibility, or that the outcome was so complex that no human role or group of humans fulfilling a role are responsible for the outcome. While this is an important question, it does not mean that hybrid responsibility should be rejected. Proponents of hybrid responsibility themselves have emphasised this point as an argument in favour of this approach. Hanson [14] argues that systems incorporating machine learning may become so complex that determining which part is responsible for any decision is effectively impossible. However, this does not prevent capacity responsibility from being assigned to the combination of the system and its users [14].

For computational design systems, Hanson's and Johnson's accounts agree that the combination of the humans associated with the system and the system itself is causally responsible for the products it designs. While they differ in whether the design system itself shares some degree of capacity responsibility, they both attribute that responsibility to the humans associated with it. While this resolves the responsibility gap, it places a new emphasis on resolving the 'problem of many hands' in determining how the various types of moral responsibility should be attributed when multiple humans or groups contributed to an outcome [27]. This point is addressed in the next section.

5 Moral responsibilities of human users and developers

For computational design systems, the three approaches to resolving the AI responsibility gap (instrumentalism, machine ethics, and hybrid responsibility) converge on attributing capacity responsibility to the humans associated with the system. However, as Gunkel [13] rightly observes in his discussion of hybrid responsibility, there is still the need to determine who among these human users and developers is responsible and for what. The collection of human roles that share capacity responsibility may be defined in multiple ways. The responsibility for an AI system could be distributed across the developers of the system, the users who input data into it, those who commissioned its use, the designers of the hardware that it operates on, and so on. Hanson [14] admits that there is "a point of diminishing returns" in specifying the elements of capacity responsibility associated with an AI system, and that constant and inconsequential elements (such as gravity and the specific model of a computer, respectively) may be disregarded. Johnson's account sidesteps this problem by limiting her discussion to three entities (human users, human developers, and the computational system itself). We will follow Johnson in

highlighting the human users and developers as the primary focus of capacity responsibility.[1]

This distinction between users and developers is justified by the role responsibilities they have in their association with the system. Role responsibilities are the duties that accompany specific social roles [15]. The computational design system's human users use it as a tool for achieving the goal of creating new product designs. The users therefore have the role of human product designers, which introduces the expectation that they will consider the purpose, physical characteristics, potential uses, and likely risks of the products they design. The developers design and create the computational design system as a whole and the system components specifically designed and incorporated into it. They are expected to consider the functionality and performance of the software they create and ensure that it is as error-free as possible.

The role responsibilities of designers (as users) and developers also serve as the basis for what they are accountable and blameworthy for. Accountability is the requirement for someone with capacity responsibility to provide an explanation or justification for actions they are causally responsibility for where there is the possibility of wrongdoing [36]. Accountability may also serve as the basis for establishing blameworthiness. To avoid being blameworthy, the explanation provided by those held accountable must show that either no wrongdoing occurred, that they were unaware of the potential outcome of their actions (the knowledge condition), that they were unable to do otherwise (the freedom condition), that they were not casually responsible, or that they lacked capacity responsibility [36].

For human product designers, the potential forms of wrongdoing are that the design is unfit for its intended purpose and that it is unnecessarily dangerous to its users. For the purposes of accountability for using computational design systems, the relevant problems here are how to ensure that the design created by the system for the human designer is safe to use and fit for purpose. Creating a design that is unfit for purpose may be due to the designer having an inaccurate or confused conception of the product's intended purpose, or by making errors in specifying the materials to be used to create it. Inappropriate materials and design flaws may also make the designed product unsafe to use. If we assume that the designer rather than the system chooses the materials and the requirements for the created design, then the designer is relying on the developer creating the computational design system, ensuring that the products it designs are likely to be safe. This may be achieved by incorporating the safety requirements and regulations that apply to human-designed products into the computational design system.

Noorman and Johnson [28] rightly observe that the developers and users of robotic and AI systems impose constraints on their possible outputs. The users set the design requirements for the products the system will design. The developers define the search space available to the system to search for potential designs and define the parameters it uses to determine the fitness of the potential designs. The changeable parameters and the relationships between them are defined by the developers of parametric design systems. These constraints allow the system's designs to be within the known performance characteristics of the intended production materials of the selected design. As suggested in our earlier discussions of instrumentalism and machine ethics, the regulatory requirements for the type of product to be designed may also be defined as constraints within the search space. For parametric systems, regulatory requirements (such as strength and weight tolerances, for example) could be specified as limits for the parameters. The system would reject potential designs that exceed these constraints as unsuitable solutions, in the same way that it would reject other potential designs that fail to meet the requirements defined by the user.

The potential forms of wrongdoing are similar for developers: that their creation is unfit for purpose, that it causes unnecessary harm to its user (through damaging data or hardware, for example), or by unfairly copying the software developed by others without permission. We will focus on the problem that the computational design system itself is unfit for purpose: it does not create designs that function as expected as physical products as they are predicted to do by the system.

Historically human developers have been reluctant to accept accountability for computer systems. Nissenbaum [26] describes four such rationalisations: the problem of 'many hands' contributing to the system, the acceptance that bugs will always exist within complex computer systems, attributing causal responsibility for errors to the system itself rather than to its developers, and the developers' own rejection of liability for the software they create. For systems incorporating machine learning techniques (such as computational design systems), de Laat [7] adds 'opacity' to this list: the lack of transparency in how the system produces outputs. Four of these rationalisations may be defensible for computational design systems (as our focus is on moral rather than legal responsibility, we will not consider the legitimacy of rejecting liability). Computational design systems are complex software that itself runs on complex software and hardware platforms, each of which may contain

[1] The product manufacturers might also considered be relevant if they are distinct from the system's human users. However, the duties of performing the manufacturing process successfully and using the correct materials that accompany the role of manufacturer are unchanged by the introduction of computational design.

errors that affect their operation. Debugging and testing will help to identify the source of errors and where they exist within the system, even if they cannot guarantee that all errors are known or removed. Each response to the responsibility gap acknowledges the causal responsibility of the computational design system. The problem of opacity will depend on the form of machine learning used within the computational design system. While the outcomes of evolutionary algorithms are effectively unpredictable, the methods by which these outcomes are determined and evaluated are known to the developers. These include the fitness function and simulation used to evaluate the generated genotypes and how the phenotype representing the physical product designs are represented as genotypes. For parametric design systems, potential errors may exist in how the relationship between different parameters is defined, and in how the parameters are optimised by the system.

In cases where the developers are accountable but not blameworthy for the harm caused by their system creates a responsibility for them to modify it to avoid such harm occurring in the future. Using computational design is a form of experimentation. While the developers can predict with a reasonable level of confidence the range of possible system outputs, there will necessarily be some uncertainty about the output. The purpose of these systems is to produce solutions to problems that are too complex or impractical to resolve using traditional methods where the output is (in principle) predictable. The computational design system is effectively an experimenter within the boundaries of the search space defined for it to find possible designs. The flaws it uncovers through the designs it creates are counterexamples to the developers' claims to have accurately represented the materials and physical constraints to the system. Correcting these flaws will reduce the reality gap between the simulation and fitness function, and the actual context where the design will be used. While they are not blameworthy for these flaws if they had not been previously discovered or predicted, developers of computational design systems have an obligation to correct these flaws in their systems that are exposed by any faulty product designs the computational system creates.

Some uncertainty will always remain with the output of any machine learning system, regardless of the constraints incorporated into it. Malicious users may game the system to deliberately produce harmful outputs. The system may use unforeseen and untested interactions in materials that cause harmful failures in the products it creates. Product designs may fall into a 'reality gap' between the simulation and the product's actual use context. Establishing the specific cause of the fault in the product should indicate whether the fault was due to an error in defining the properties of the search space, a problem in the physical manufacturing process, or due to an unexpected interaction between the materials and the design developed by computational methods. The specific cause of the fault will determine whether the human developers, the human users, or the computational system itself is blameworthy for the fault. Each of these human parties is accountable for an explanation of why the fault occurred, and which party is potentially blameworthy depends on the type of fault. The developer is accountable for faults that are due to the product's design. The user is accountable for faults that result from the product's specification and design requirements (i.e., whether it is fit for purpose). Their accounts for why the product was flawed or caused harm determines whether they are blameworthy for that harm (i.e., if they were negligent in using, creating or testing the system) or if it was beyond their expected knowledge of how the system operates and the materials used to create the product. For example, the developer would not be blameworthy for a fault if their account explains how it is the result of unforeseen problems with how the physical world is represented in the simulation.

So here, we find a solution to the problem of assigning moral responsibility for products designed using computational design systems. Unless the fault is the result of an unforeseen interaction between the materials comprising the product and the designs created via computational design, the human users or developers are wholly accountable for faults in the design of the product and the product itself. Human users are blameworthy as product designers if their account reveals that they would be expected to know about the errors and potential risks in the parameters and fitness functions that they defined for the computational design system to use. If the developers' account of what caused the design flaws or product faults reveals negligence in designing, developing, or testing the system, then the developers are also blameworthy. While the developers are not blameworthy for flaws discovered by the computational design system, they have an obligation to correct their models to remove them once they have been revealed using the system.

6 Conclusion

Using computational design to create physical products raises two potential questions for responsibility: does employing computational design create a 'responsibility gap' for product design, and if there is no responsibility gap, who is morally responsible for the products created with the aid of these systems? Recognising the different forms of responsibility (particularly the distinction between causal responsibility and capacity responsibility) and building on the existing philosophical work on moral responsibility and AI systems provide us with the conceptual tools to address this problem.

Here, we have argued that there is no responsibility gap by showing that the human users of computational design systems and the human developers of these systems are

morally responsible for the designs these systems produce. This moral responsibility includes being accountable for design flaws and product faults, being blameworthy for negligence in developing the system that causes harmful design flaws and product faults and having an obligation to correct flaws in how the system produces designs that could not be discovered without being uncovered by faulty designs produced by the system. In doing so, we hope to demonstrate how the 'responsibility gap' for products designed and made by robots and AI systems may be addressed by considering how responsibility is attributed in specific applications of these technologies.

Acknowledgements This research was informed by a series of interviews with professionals involved in the computational design of products, as approved by CSIRO's Social and Interdisciplinary Science Human Research Ethics Committee in line with the guideline specified in the (Australian) National Statement on Ethical Conduct in Human Research. The authors thank those participants for generously sharing their insights and time to this research. We also thank the anonymous reviewers for their useful comments and suggestions.

Author contributions All authors contributed equally to developing the topic and argument of the paper. DMD wrote the text, with contributions by DH and JL.

Funding This research was funded by CSIRO's Responsible Innovation Future Science Platform.

Availability of data and materials Not applicable.

Compliance with ethical standards

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

Code availability Not applicable.

References

1. Allen, C., Wallach, W., Smit, I.: Why Machine Ethics? In: Anderson, M., Anderson, S.L. (eds.) Machine Ethics, pp. 51–61. Cambridge University Press, Cambridge (2011)
2. Anderson, M., Anderson, S.L. (eds.): Machine Ethics. Cambridge University Press, Cambridge (2011)
3. Bongard, J.C.: Evolutionary robotics. Commun. ACM **56**(8), 74–83 (2013). https://doi.org/10.1145/2493883
4. Caetano, I., Santos, L., Leitão, A.: Computational design in architecture: defining parametric, generative, and algorithmic design. Front. Archit. Res. **9**(2), 287–300 (2020). https://doi.org/10.1016/j.foar.2019.12.008
5. Coeckelbergh, M.: AI Ethics. The MIT Press, Cambridge (2020)
6. Davis, M.: "Ain't No One Here But Us Social Forces": constructing the professional responsibility of engineers. Sci. Eng. Ethics **18**(1), 13–34 (2012). https://doi.org/10.1007/s11948-010-9225-3
7. de Laat, P.B.: Algorithmic decision-making based on machine learning from Big Data: can transparency restore accountability? Philos. Technol. **31**(4), 525–541 (2018). https://doi.org/10.1007/s13347-017-0293-z
8. Domingos, P.: The Master Algorithm. Penguin, London (2015)
9. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 2nd edn. Springer, Berlin (2015)
10. Elish, M.C.: Moral crumple zones: cautionary tales in human-robot interaction. Engag. Sci. Technol. Soc. **5**, 40–60 (2019). https://doi.org/10.17351/ests2019.260
11. Epstein, Z., Levine, S., Rand, D.G., Rahwan, I.: Who gets credit for ai-generated art? iScience **23**(9), 101515 (2020). https://doi.org/10.1016/j.isci.2020.101515
12. Eshleman, A.: Moral Responsibility. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy (Winter 2016 Edition). https://plato.stanford.edu/archives/win2016/entries/moral-responsibility/ (2016). Accessed 12 Jan 2021
13. Gunkel, D.J.: Mind the gap: responsible robotics and the problem of responsibility. Ethics Inf. Technol. (2017). https://doi.org/10.1007/s10676-017-9428-2
14. Hanson, F.A.: Beyond the skin bag: on the moral responsibility of extended agencies. Ethics Inf. Technol. **11**(1), 91–99 (2009). https://doi.org/10.1007/s10676-009-9184-z
15. Hart, H.L.A.: Punishment and Responsibility: Essays in the Philosophy of Law, 2nd edn. Oxford University Press, Oxford (2008)
16. Heaven, D.: The designer changing the way aircraft are built. BBC future. https://www.bbc.com/future/article/20181129-the-ai-transforming-the-way-aircraft-are-built (2018). Accessed 8 June 2020
17. Jackson, C.: Generative design and engineering ethics: where's the intersection? Lifecycle insights. https://www.lifecycleinsights.com/generative-design-and-engineering-ethics-wheres-the-intersection/ (2019). Accessed 23 Nov 2020
18. Johnson, D.G.: Computer systems: moral entities but not moral agents. Ethics Inf. Technol. **8**(4), 195–204 (2006). https://doi.org/10.1007/s10676-006-9111-5
19. Kicinger, R., Arciszewski, T., de Jong, K.: evolutionary computation and structural design: a survey of the state-of-the-art. Comput. Struct. **83**, 1943–1978 (2005). https://doi.org/10.1016/j.compstruc.2005.03.002
20. Knight, T., Stiny, G.: Making grammars: from computing with shapes to computing with things. Des. Stud. **41**, 8–28 (2015). https://doi.org/10.1016/j.destud.2015.08.006
21. Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., et al.: The surprising creativity of digital evolution: a collection of anecdotes from the evolutionary computation and artificial life research communities. Artif. Life **26**, 274–306 (2020). https://doi.org/10.1162/artl_a_00319
22. Lohn, J.D., Hornby, G.S., Linden, D.S.: An evolved antenna for deployment on NASA's space technology 5 mission. In: O'Reilly, U.M., Yu, T., Riolo, R., Worzel, B. (eds.) Genetic Programming Theory and Practice II, pp. 301–315. Springer, Boston (2005)
23. Matthias, A.: The responsibility gap: ascribing responsibility for the actions of learning automata. Ethics Inf. Technol. **6**(3), 175–183 (2004). https://doi.org/10.1007/s10676-004-3422-1

24. McCormack, J., Gifford, T., Hutchings, P.: Autonomy, authenticity, authorship and intention in computer generated art. In: Ekárt, A., Liapis, A., Castro Pena, M.L. (eds.) Computational Intelligence in Music, Sound, Art and Design. EvoMUSART 2019, pp. 35–50. Springer, Cham (2019)
25. Moor, J.H.: The nature, importance, and difficulty of machine ethics. In: Anderson, M., Anderson, S.L. (eds.) Machine Ethics, pp. 13–20. Cambridge University Press, Cambridge (2011)
26. Nissenbaum, H.: Accountability in a Computerized Society. In: Friedman, B. (ed.) Human values and the design of computer technology, pp. 41–64. CSLI Publications and Cambridge University Press, New York (1997)
27. Noorman, M.: Computing and Moral Responsibility. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy (Spring 2018 Edition). https://plato.stanford.edu/archives/spr2018/entries/computing-responsibility/ (2018). Accessed 12 Jan 2021
28. Noorman, M., Johnson, D.G.: Negotiating Autonomy and Responsibility in Military Robots. Ethics Inf. Technol. **16**(1), 51–62 (2014). https://doi.org/10.1007/s10676-013-9335-0
29. Owen, R., Stilgoe, J., Macnaghten, P., Gorman, M., Fisher, E., Guston, D.: A framework for responsible innovation. In: Owen, R., Bessant, J., Heintz, M. (eds.) Responsible Innovation: Managing the Responsible Emergence of Science and Innovation in Society, pp. 27–50. Wiley, Chichester (2013)
30. Renner, G., Ekárt, A.: Genetic algorithms in computer aided design. Comput. Aided Des. **35**(8), 709–726 (2003). https://doi.org/10.1016/S0010-4485(03)00003-4
31. Samuelson, P.: AI authorship? Commun. ACM **63**(7), 20–22 (2020). https://doi.org/10.1145/3401718
32. Stahl, B.C., Coeckelbergh, M.: Ethics of healthcare robotics: towards responsible research and innovation. Robot. Auton. Syst. **86**, 152–161 (2016). https://doi.org/10.1016/j.robot.2016.08.018
33. Stanley, K.O.: Compositional pattern producing networks: a novel abstraction of development. Genet. Program Evolvable Mach. **8**(2), 131–162 (2007). https://doi.org/10.1007/s10710-007-9028-8
34. Trefzer, M.A., Tyrrell, A.M.: Evolvable Hardware: From Practice to Application. Springer, Berlin (2015)
35. Tutum, C.C., Vouga, E., Chockchowwat, S., Miikkulainen, R.: Functional generative design: an evolutionary approach to 3D-printing. Paper presented at the genetic and evolutionary computation conference (GECCO '18), Kyoto, Japan, 15–19 June 2018
36. van de Poel, I.: The Relation between forward-looking and backward-looking responsibility. In: Vincent, N.A., van de Poel, I., van den Hoven, J. (eds.) Moral Responsibility, pp. 37–52. Springer, Dordrecht (2011)