Challenges in Creating Effective Automated Design Environments: An experience report from the domain of generative manufacturing

Challenges in Creating Effective Automated Design Environments: An experience report from the domain of generative manufacturing

David Garlan
Bradley Schmerl
garlan@cs.cmu.edu
schmerl@cmu.edu
School of Computer Science,
Carnegie Mellon University
Pittsburgh, PA, USA

Rebekka Wohlrab
wohlrab@chalmers.se
Chalmers | University of Gothenburg
Gothenburg, Sweden

Javier Cámara
jcamara@uma.es
ITIS Software, Universidad de Málaga
Málaga, Spain

ABSTRACT

The emergence of powerful automated design tools in many domains is changing the nature of design, as human-intensive activities can be increasingly off-loaded to those tools. Rather than having a human consider only handful of options, as has been done historically, such tools now enable the generation of a large space of potential designs, exhibiting different tradeoffs among competing qualities of merit, and supporting systematic exploration of the design space. At the same time, this paradigm raises new challenges centered on enabling humans to effectively navigate that generated space in order to select a design that best meets their requirements. In this paper we describe our experience in the domain of generative manufacturing, in which we developed a novel design environment for airplane parts manufacturing that incorporates a number of sophisticated design tools and attempts to tackle the emergent problems of design space exploration that are faced by designers of those parts. We use this experience to highlight the challenges that we faced and reflect on their applicability more generally to tool-assisted software design environments.

ACM Reference Format:
David Garlan, Bradley Schmerl, Rebekka Wohlrab, and Javier Cámara. 2024. Challenges in Creating Effective Automated Design Environments: An experience report from the domain of generative manufacturing. In *2024 International Workshop on Designing Software (Designing '24), April 15–14, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3643660.3643949

1 INTRODUCTION

In the past, design has typically been a human-intensive activity: given a set of requirements, perhaps only partially specified, the designer would attempt to come up with a design that best satisfies the known requirements. Given limited human cognitive abilities, the process of considering possible designs for a given artifact has typically been limited to a small number of potential candidates from which the most promising is selected for implementation. This traditional process of designing relies heavily on the experience and skill of the designer to identify those design candidates, with the potential problem of having missed out on ones that might be better than those under consideration. Even when this process results in an adequate solution, it is not obvious that it will be sufficient if market conditions and requirements change. Because this process is so labor- and expertise-intensive, the cost of redesign is often so high that companies simply make do with sub-optimal solutions.

Today, in many domains this paradigm is changing. With the increasing availability of design tools that can automate, or partially automate, the generation of candidate designs based on a set of formalized requirements, a designer can now be presented with a very large number of candidates, each exhibiting different tradeoffs in a complex multi-dimensional space of possibilities. Examples include design tools for manufacturing [8], embedded systems [11], and software architecture [16]. In the arena of software systems design the increasing sophistication of generative AI tools is likely to only accelerate this trend.

However, the ability to automatically generate a large space of potential designs raises its own challenges. How can the designer understand the overall space in terms of the main tradeoffs? How can the designer focus on the regions of the space that are most relevant? What combinations of requirements are simply infeasible? What are key requirements thresholds that determine when one kind of solution is better than another?

Over the past two years, we and our colleagues have been involved in an exploratory research project, collaborating with an industrial partner, to develop design tools for generative manufacturing — and specifically for airplane parts manufacture. The goal was to develop a design environment to enable rapid and agile part design and manufacturing based on requirements by incorporating a suite of existing and novel design tools. Over the course of two years and two prototypes, we encountered numerous challenges in finding ways to provide designers with the ability to answer the kind of design questions noted above.

In this case study, we use our experience to examine the nature of those questions in the context of the manufacturing domain and the solutions that we came up with, reflecting on the applicability of those concerns and solutions more generally to tool-assisted design, and in particular to software systems design. We start by briefly surveying the background of design generation tools and mechanisms used in prior research to improve the understandability of complex design spaces. Then in Section 3 we describe the design environment that we came up with, focusing on the challenges we faced and lessons learned in developing two prototypes. Section 4 attempts to generalize from this experience, focusing on three key lessons that we believe would apply to design environments in other domains and specifically software system design. In the following section we discuss additional capabilities that we identified as of potential benefit, and conclude in Section 5.

2 BACKGROUND: MANUFACTURING AND BEYOND

The current state of the practice in manufacturing relies on the expertise of designers, who have to internalize many of the requirements and design choices with respect to materials, manufacturing techniques, tolerances, supply chains, etc. In practice this typically means that designers consider only a handful of options based on their past experience. Once selected, there is considerable cost to redesign, and consequently manufacturing plans are rarely changed. This situation leads to sub-optimal choices at the beginning of the process that become increasingly sub-optimal as requirements or aspects of the environment change (such as the cost of materials or the availability of suppliers for manufacturing the product).

Current manufacturing techniques involve the use of commercial computer-aided design (CAD) tools to design the parts to be manufactured, and computer-aided manufacturing (CAM) tools to develop instructions on how to build the designed part, for example using milling or printing machines. These tools typically require designers to manually make tradeoffs in the design space to develop a solution, which they then specify with these tools. The tools may give them some guidance, options, and analysis to help with the design, but fundamentally the designer already needs to have a design in mind, and how it is to be manufactured.

However, the design space for manufacturing is becoming increasingly complex, with more choices in materials, manufacturing methods, and alternative supply chains. Add to that a changing set of qualities and contexts, as well as changes to supply chains and manufacturers, and the design space that must be considered to produce an optimal part for a given context quickly becomes intractable for a designer to manage. Moreover, existing tools do not support reasoning about the tradeoffs that a designer has to make, such as how choice of material (aluminum, plastic, etc.) might affect cost and schedule.

This situation has led engineering researchers to investigate new approaches to CAD and CAM tools that use AI to generate and rank alternatives. For example, in [5] the authors use machine learning to optimize the topology of a designed piece given physical properties such as maximum mass and rigidity. The use of such tools shifts the designer's cognitive load from having to manually consider how to make tradeoffs in a complex design space, to having to examine and understand a large number of generated alternatives.

However, as noted above, the existence of many such possible designs introduces the new problem of navigating that space and understanding its key features. What is needed to support the process of automated design generation is tooling that helps designers understand the design space and the options that are being presented so that they can make informed choices about which of the generated designs best meets their needs.

Design space exploration has been broadly studied in many areas areas including product lines [15], model-based performance prediction [1] and formal verification [10]. However, *explanation of design spaces* has been largely unaddressed until recently, partly due to the considerable amount of data that it requires, which is often difficult to obtain, as well as for the challenge posed by the limited capacity of humans to effectively assimilate large quantities of information.

To address this challenge, our recent work [3, 4, 6, 17] has explored the use of dimensionality reduction techniques, traditionally used in areas such as biology and machine learning [13], to identify and facilitate the understanding to a human designer of the main design decisions and tradeoffs in a design space.

Principal Component Analysis (PCA) [9] is used in [3] by software architects to understand which design decisions contribute the most to the satisfaction of system qualities (e.g., performance, availability) by identifying the subset of design variables that explain the most variation across the design space. Additionally PCA indicates correlations and anti-correlations between these design variables (e.g., using component X in a software configuration is positively correlated with cost, and negatively with response time).

The range of tools employed in [3] is extended in [4, 17], which also incorporate: (i) Multiple Correspondence Analysis [12] (MCA – similar in functionality to PCA, but for categorical variables), (ii) Decision Tree Learning (DTL) [2], which is used to explain how concrete choices associated with specific design decisions influence the qualities across the design space, and (iii) clustering [14] to identify the main categories of solutions in the space. This prior work describes explores a combination of those tools in the contexts of software architecture [4] and planning for robotic autonomous systems [17]. These techniques are generalized in [6], which describes a design space explanation process and lessons learned from experience those domains, as well as the generative manufacturing case, described in this case study.

3 GENERATIVE MANUFACTURING TOOL SUPPORT

In this section we describe the design environment that we developed working with colleagues and industrial collaborators in the generative manufacturing domain. Over the course of two years we developed two major prototypes, incorporating lessons learned from the first into the second. After describing those lessons and how we addressed them, we consider in Section 4 how those lessons might generalize to other domains, and specifically how they might provide insight into tool-assisted software systems design.

Our initial generative manufacturing prototype used design generation tools to pre-compute a large space of designs that can then be explored by a designer. This is done in two phases. In the first phase a designer inputs a specification of the part's topological constraints (i.e., its rough shape), and dimensions of quality such as strength, tolerances, options of material (aluminum, steel, etc.) and manufacturing methods (additive, subtractive, etc.). The design tools then sample the entire design space, using CAD tools to generate a large number of potential designs (in the thousands). From this very large space, the tools identify the Pareto optimal ones, which represent the best designs for a given weighting of quality dimensions – mass, stiffness, and cost and time to manufacture the part. This is a reduced, but still very large, number of candidates.

In the second phase the designer can specify alternative attribute weights to understand the tradeoffs among qualities for the pre-computed Pareto optimal designs. By varying the weights on quality dimensions using the tool, the designer can explore the space, one design at a time, displaying the shape and properties of each, to find one that best satisfies their needs.

This first prototype was a vast improvement over the more-manual processes used by designers at our partner's company, since it allowed rapid exploration of many possible designs. However, it had a number of serious shortcomings.

First, it was difficult for the designer to get an overall sense of the design space landscape. Thus it was hard for them to answer questions like: What regions are feasible and which aren't? (For example, is it possible to use aluminum to make the part cheaply?). What are key thresholds of quality weights that lead to different outcomes? (For example, what is the maximum strength that we can get if we restrict ourselves to additive manufacturing?) How many other designs are close to a given one? (For example, are there many other designs that also use aluminum and additive manufacturing?)

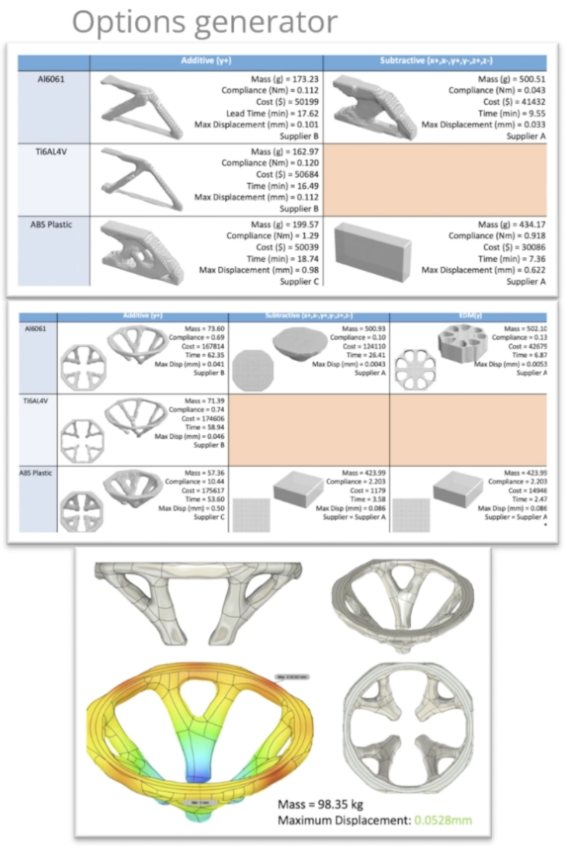
Second, some of the information required by the tools to pre-compute the design space was not readily available up front. In particular, getting time and cost values for manufacturing a specified quantity of parts, given a part's design, required reaching out to the various part manufacturers for "bids". While this might be feasible for small numbers of designs, it cannot be done for thousands.

Third, large parts of the pre-computed design space were of little interest to the designer. Often there were constraints on various design properties (such as maximum manufacturing cost) that immediately ruled out certain designs. Including these designs added unnecessary complexity to the set of design options provided by the tool, further complicating design space understanding.

Fourth, it was difficult for designers to intuitively connect utility weights with design outcomes. For example, how would increasing the significance of having low mass decrease its corresponding flexibility and cost of manufacturing?

Based on this experience, the second prototype moved to a very different process: instead of pre-generating a comprehensive and large space of possible designs, the tool uses an iterative process in which a designer incrementally explores the design space, adding new designs to the considered pool of possible solutions, by specifying constraints, and then using our tools to understand the pool of generated solutions that satisfies those constraints. This had a number of benefits.

First, by reducing the number of designs under consideration, it now became feasible to obtain supplier bids to use in comparing business-related properties from the part manufacturers. Although there was some residual uncertainty in these bids at an early stage of design, they provided a realistic basis for including key concerns of cost and schedule into the design process.



Options generator

	Additive (y+)	Subtractive (x+,x-,y+,y-,z+,z-)
Al6061	Mass (g) = 173.23 Compliance (Nm) = 0.112 Cost ($) = 50199 Lead Time (min) = 17.62 Max Displacement (mm) = 0.101 Supplier B	Mass (g) = 500.51 Compliance (Nm) = 0.043 Cost ($) = 41432 Time (min) = 9.55 Max Displacement (mm) = 0.033 Supplier A
Ti6AL4V	Mass (g) = 162.97 Compliance (Nm) = 0.120 Cost ($) = 50684 Time (min) = 16.49 Max Displacement (mm) = 0.112 Supplier B	🟧
ABS Plastic	Mass (g) = 199.57 Compliance (Nm) = 1.29 Cost ($) = 50039 Time (min) = 18.74 Max Displacement (mm) = 0.98 Supplier C	Mass (g) = 434.17 Compliance (Nm) = 0.918 Cost ($) = 30086 Time (min) = 7.36 Max Displacement (mm) = 0.622 Supplier A

	Additive (y+)	Subtractive (x+,x-,y+,y-,z+,z-)	EDM(y)

Al6061	Mass = 73.60
	Compliance = 0.69
	Cost = 167814
	Time = 62.35
	Max Disp (mm) = 0.041
	Supplier B
		Mass = 500.93
		Compliance = 0.10
		Cost = 124110
		Time = 26.41
		Max Disp (mm) = 0.0043
		Supplier A
			Mass = 502.10
			Compliance = 0.13
			Cost = 42679
			Time = 6.87
			Max Disp (mm) = 0.0053
			Supplier A

Ti6AL4V	Mass = 71.39
	Compliance = 0.74
	Cost = 174606
	Time = 58.94
	Max Disp (mm) = 0.046
	Supplier B
		🟧	🟧

ABS Plastic	Mass = 57.36
	Compliance = 10.44
	Cost = 175617
	Time = 53.60
	Max Disp (mm) = 0.50
	Supplier C
		Mass = 423.99
		Compliance = 2.203
		Cost = 1179
		Time = 3.58
		Max Disp (mm) = 0.086
		Supplier = Supplier A
			Mass = 423.99
			Compliance = 2.203
			Cost = 14946
			Time = 2.47
			Max Disp (mm) = 0.086
			Supplier = Supplier A

Figure 1: Option Generation.

Second, the new environment included design space explanation tools – specifically, ways to support comparison of systems across various dimensions and mechanisms to understand the key factors that influence design choices, as we describe in more detail below.

Third, this approach better corresponded to a designer's way of working, by taking advantage of their prior experience and knowledge to set initial bounds on the space of solutions through a set of iteratively refined constraints and preferences, but still effectively using the tools to interpret and enhance that space through design generation and explanation. Moreover, using constraints to bound the search space eliminated the generation of large numbers of irrelevant designs.

The second generation tool is illustrated in Figures 1—3. Figure 1 depicts designs for two parts that are generated by the CAD tool, which takes a set of requirements about the geometries and constraints on the solution (such as ranges of cost, mass, and time to deliver) and generates a set of designs that satisfy those constraints.

A key feature of this display is that regions of the design space that are *not* feasible are made explicit: in Figure 1 the cells shaded in orange indicate that there is no feasible design in this space for the given input constraints. The bottom of Figure 1 shows that the designer can view specific details about a particular design and the values of associated metrics.

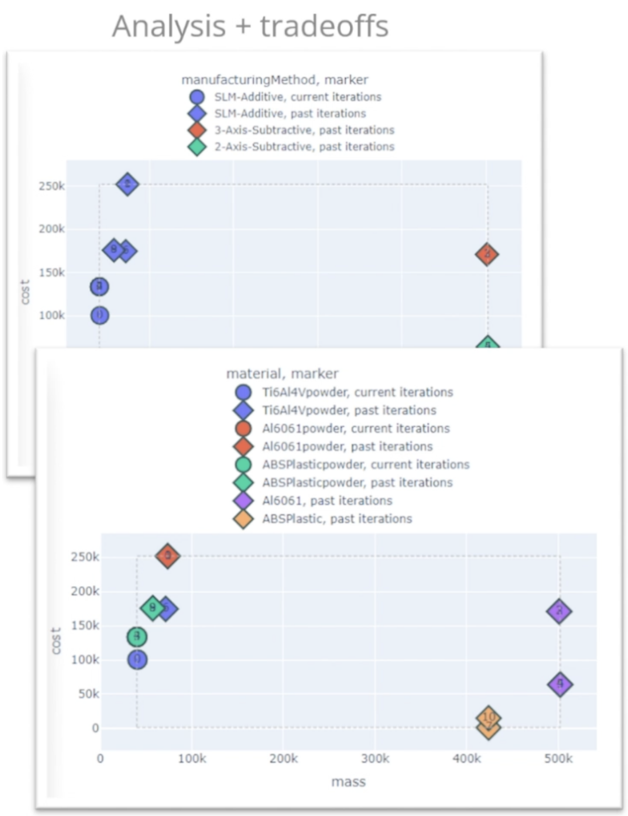


Figure 2: Pairwise Trade-off Analysis.

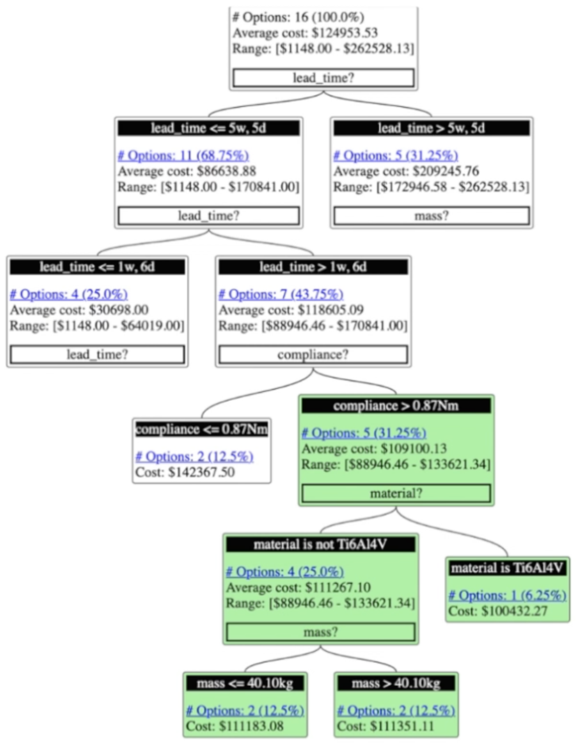


Figure 3: Decision Tree Analysis.

Figure 2 shows how the tool facilitates pairwise tradeoff analysis. For specified pairs of qualities, each of the currently generated designs is plotted along where it lies on each dimension. Such a plot gives the user an idea of how two quality attributes may be related, and would be used by a designer to (a) refine constraints further to restrict the set of options for the next iteration, or (b) identify parts of the design space that haven't been considered and try to come up with options in those unpopulated areas.

Figure 3 shows an abbreviated learned decision tree indicating how a particular quality is impacted by all of the other qualities. The decision tree depicted in the figure is showing to the designer how cost is impacted by decisions about lead time, compliance (flexibility) and choice of manufacturing material. As shown, the top-level decision points are based on lead time (i.e., the time to complete the manufacturing) – meaning that decisions about lead time are the most important with respect to differentiating cost, followed by compliance, then whether the material is titanium, followed by the mass. The decision tree also indicates how many of the designs make similar decisions, helping the designer understand clustering behavior.

In contrast to the pairwise comparison, the decision tree provides a combined view about how a particular quality is impacted by all the other concerns. This allows a designer to understand what parts of the design space might be missing from consideration to generate options in the next iteration, or to understand how many similar options may exist in a particular part of the design space, as well as indicating thresholds (for numerical variables) and decisions (for categorical variables) that influence the cost of design. Other decision trees can be created to "explain" other variables, such as the time required for manufacture of the part in the quantity needed.

4 BEYOND MANUFACTURING

Stepping back from the details of our tool and the particulars of the manufacturing domain, we can consider some general observations about tool-assisted design space exploration, and in particular, how these might apply to software system design.

Designs and requirements co-evolve: The availability of automated design tools might naively suggest that given a well-defined set of requirements, a designer simply pushes a button and the tool spits out the optimal design satisfying those requirements. In practice this doesn't work. First, the designer may not know what is feasible, let alone optimal, without exploring a bit of the design space. Second, it may not be clear what are the inherent tradeoffs associated with the design: by strengthening one requirement it is often unclear what other requirements or desired properties are also affected. Thus it essential for tools to provide a collection of designs and ways to compare them in order for the designer to understand which requirements and desired properties are feasible, and how those impact other requirements and properties.

At the same time, it must be possible for the designer to interactively control the number of designs that are under consideration.

While dimensionality reduction techniques, noted in Section 2, can reduce the complexity of design spaces, in practice it may work much better if the designer can directly affect which designs are available for comparison. This is particularly true when working with experienced designers, who can bring their understanding about what makes sense in their domain to help them focus the search and comparison process, while refining their more-detailed understanding of the possible requirements for a particular design task. Moreover, when the cost of elaborating the properties of a particular design (such as cost and manufacturing lead time for a set of manufactured parts) is non-trivial, it is important to bound the set of designs under consideration.

We argue that the need to support co-evolution of design and requirements in a tool-assisted context applies equally to software systems. A software system designer may know *what* quality attributes are important to their design, but they likely do not know precisely *how* their design choices affect quality outcomes or what tradeoffs need to be made to achieve desired levels of certain properties. Through design exploration and design space explanation of the sort we outlined earlier, a much clearer picture emerges of what can be achieved and at what cost.

In a similar way to the manufacturing context, software systems designers also can benefit from the ability to guide the search and generation of alternatives. For example, while a software architect might use an architectural design tool to generate a very large set of possible systems designs from scratch, in practice it is likely that such designs will be variations on some standard styles or frameworks that the architect already has in mind, or are already in place. Or, the design generation tools might be used to improve a "seed" design using a set of tactics, for example improving performance or availability. Further, there are likely to be certain properties of the designs – such as system-level performance – that require prototyping or simulation, and that therefore can feasibly be determined for only a relatively small number of candidates.

Design space "explanation" is a necessity: Regardless of the size of the generated space, beyond a very small number of candidates it becomes crucial to provide summaries and explanatory views of the current set of candidates chosen by the designer.

In our first prototype, we allowed a designer to explore the space of candidates, but did not provide ways to visualize the overall space itself or the tradeoffs among a set of alternatives. As a result, designers had to perform a lot of manual traversal, using a form of "generate-and-test" exploration.

In contrast, as described earlier, the second prototype provided (a) trade-off views and (b) decision tree views. With the former, designers could see all of the current candidates mapped onto 2-dimensional trade-off plots (cf. Figure 2). With the latter, designers could identify the key thresholds and decision points that led to alternative sets of designs with respect to a target property, such as which factors, and which thresholds of the design account for the cost of manufacturing (cf. Figure 3).

As we outlined earlier, there is a rich set of such visualizations and design space reduction mechanisms that can potentially be applied across many domains. However, we learned in working with manufacturing designers that not all of these are useful for all domains. For example, Principal Component Analysis can identify the strongly correlated or anti-correlated factors that account for the variability in a set of designs. However, this view turned out to be of little value for manufacturing, where such correlations are already well-understood by designers (e.g., that cost and schedule are anti-correlated).

This suggests, more generally, that care must be taken to tailor the potential space reduction techniques to the domain. For software systems design this is also likely to be true: in some domains it is clear to a seasoned designer what are the essential qualities of interest, and how are they correlated – at least at a high level. But at a more detailed level the particular questions they need to answer using the tools may depend substantially on the kind of system being designed [7].

Human expertise remains essential: While for some domains many of the properties that are important in developing a good design can be incorporated into generative design tools, there typically exist relevant properties that are not easily encoded or mechanized. These properties are often based on knowledge that the designer has outside the realm of automation and requirements specification.

In the manufacturing world, such properties included knowledge of the history of interactions with particular manufacturing subcontractors, hunches about the future price of materials, the reliability of certain manufacturing processes, etc. Recognizing this inherent inability of automated design tools to capture all dimensions of concern and knowledge, underscores the importance of creating automated design environments where such human knowledge and experience can complement the automated design process.

This observation is likely to be true for software systems design. Humans bring to the table much tacit knowledge, such as the capability of the development team to implement a particular design, strategic priorities of the company, the reliability of outsourcing partners, the expected lifetime of the system, the maturity of frameworks that might be used as system building blocks, technology trends, etc.

5 ADDITIONAL CAPABILITIES

Tools such as those we developed for assisting manufacturing design, and their likely analogs for software system design, represent an important step forward in providing automated assistance to designers. However, beyond the needs for design generation, distillation, and comparison which they address, there are several additional capabilities that we identified as being ripe for inclusion.

Change management and robustness. The domain of manufacturing undergoes frequent changes: new manufacturing methods and materials become available; the cost of materials fluctuates; the functional requirements of a part may change over time; the context in which the part is used may evolve. Given that there is a cost to changing a design, a useful analysis capability would be to evaluate how robust a design is to such changes. In other words, if the current assumptions about the manufacturing and deployment context change, will a given design continue to be adequate. Or, put another way, given some cost in making design changes, at what point would the design have to change to accommodate an evolving environment. These kinds of concerns suggest a new set of capabilities in automated design tools that would focus on evaluating and improving system design robustness to changes.

Such concerns also can directly affect software systems design. Software systems may make certain assumptions about the operating or development environments in order work correctly or effectively. However, it is important to know to what extent deviations from those expectations can be handled by the system without violating certain essential properties or making adequate design tradeoffs. This knowledge could be used to improve the robustness of the design: for example, adding redundancy to manage potential attacks, or self-adaptation mechanisms to recover from faults.

Requirements malleability and provenance. As we noted in the previous sections, requirements and designs naturally co-evolve. However, not all requirements are equally changeable. In practice some requirements may be non-negotiable, such as those that are mandated by a regulatory process. Others may have limits within which they must be satisfied, such as the cost of production.

Our current tool provides no way to indicate the degree to which a given requirement can be flexible. Thus we rely on the designer to keep in mind those restrictions and tolerances. Making them explicit would help bound the exploration, and help ensure that designs do not violate some externally mandated requirement.

To do this effectively requires automated support for requirements tracking [18]. Knowing, for example, that a particular physical property of a manufactured part is required for external certification is important in order to know how much flexibility one has in its design to make tradeoffs involving that property. Conversely, knowing that a certain target cost of production has some flexibility, provided that the part can be produced quickly, would allow alternative cost-time tradeoffs to be considered.

Similar capabilities are likely to also be essential for software systems design, where some requirements may derive from legal sources (for example privacy laws), while others are more "wish lists" where there is considerable flexibility in achieving them and making design tradeoffs.

6 CONCLUSIONS

In this paper we have outlined our experience in developing a design space exploration and understanding environment for generative manufacturing, highlighting some of the issues and challenges we faced along the way. We have argued that many of these issues generalize to other domains, such as software systems design, which are increasingly supported by design generation tools, and that have a rich set of requirements and quality dimensions that ultimately require the designer to identify a design that best balances the tradeoffs inherent in that domain.

Specifically, we identified three key observations that frame effective use of automated design generation tools: (a) the need to support co-evolution of requirements and design through designer interaction and iteration to bound and explore the space of designs, (b) the need for explanation and summarization tools that provide a designer with ways to understand the design space features and tradeoffs, and (c) the need to allow for human expertise about aspects of the design that cannot be easily encoded in design tools. We also identified two important areas for future investigation: (a) tools that can evaluate and improve design for design robustness in the face of environmental changes, and (b) tools that support requirements provenance and degree of malleability.

ACKNOWLEDGMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Spanish Government (FEDER, Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación) under projects TED2021-130523B-I00 and PID2021-125527NB-I00, and by Lockheed Martin.

In particular we would like to thank Levent Burak Kara, Greg Federer, Steve Smith, Zack Rubenstein, Hongrui Chen, Aditya Joglekar, and Ani Kimmel of Carnegie Mellon as colleagues who worked with us on the smart manufacturing project, and whose work is depicted in Figure 1.

REFERENCES

[1] Simonetta Balsamo, Antinisca Di Marco, Paola Inverardi, and Marta Simeoni. 2004. Model-Based Performance Prediction in Software Development: A Survey. *IEEE Trans. Software Eng.* 30, 5 (2004), 295–310.
[2] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 2017. *Classification and regression trees.* Routledge.
[3] Javier Cámara, Mariana Silva, David Garlan, and Bradley R. Schmerl. 2021. Explaining Architectural Design Tradeoff Spaces: A Machine Learning Approach. In *Software Architecture - 15th European Conference, ECSA 2021, Virtual Event, Sweden, September 13-17, 2021 (LNCS, Vol. 12857).* Springer, 49–65.
[4] Javier Cámara, Rebekka Wohlrab, David Garlan, and Bradley R. Schmerl. 2023. ExTrA: Explaining architectural design tradeoff spaces via dimensionality reduction. *J. Syst. Softw.* 198 (2023), 111578. https://doi.org/10.1016/j.jss.2022.111578
[5] Hongrui Chen, Aditya Joglekar, and Levent Burak Kara. 2023. Topology Optimization Using Neural Networks With Conditioning Field Initialization for Improved Efficiency *(International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. Volume 3A: 49th Design Automation Conference (DAC)).* https://doi.org/10.1115/DETC2023-116937
[6] Javier Cámara, Rebekka Wohlrab, David Garlan, and Bradley Schmerl. 2023. Focusing on What Matters: Explaining Quality Tradeoffs in Software-Intensive Systems via Dimensionality Reduction. *IEEE Software* (2023), 1–10. https://doi.org/10.1109/MS.2023.3320689
[7] J. Andres Diaz-Pace and David Garlan. 2024. The Architect in the Maze: On the Effective Usage of Automated Design Exploration. In *Proceedings of 1st International Workshop on Designing Software.*
[8] Mikell Groover and EWJR Zimmers. 1983. *CAD/CAM: computer-aided design and manufacturing.* Pearson Education.
[9] I. T. Jolliffe. 1986. *Principal Components in Regression Analysis.* Springer New York, New York, NY, 129–155.
[10] Eunsuk Kang, Ethan Jackson, and Wolfram Schulte. 2011. An Approach for Effective Design Space Exploration. In *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, Radu Calinescu and Ethan Jackson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 33–54.
[11] Simon Künzli, Lothar Thiele, and Eckart Zitzler. 2006. Multi-criteria decision making in embedded system design. *System on chip: next generation electronics* (2006), 3–28.
[12] Brigitte Le Roux and Henry Rouanet. 2009. *Multiple Correspondence Analysis.* SAGE Publications.
[13] Jake Lever, Martin Krzywinski, and Naomi Altman. 2017. Principal component analysis. *Nature Methods* 14, 7 (2017), 641–642.
[14] Angelos Markos, Alfonso Iodice D'Enza, and Michel van de Velden. 2019. Beyond tandem analysis: Joint dimension reduction and clustering in R. *Journal of Statistical Software* 91, 10 (2019).
[15] Alexandr Murashkin, Michał Antkiewicz, Derek Rayside, and Krzysztof Czarnecki. 2013. Visualization and exploration of optimal variants in product line engineering. In *Proc. of the 17th Intl. Software Product Line Conference.*
[16] Steffen Reussner, Ralf H.and Becker, Happe Jens, Robert Heinrich, and Anne Koziolek. 2016. *Modeling and Simulating Software Architectures: The Palladio Approach 1st Edition.* MIT Press.
[17] Rebekka Wohlrab, Javier Cámara, David Garlan, and Bradley R. Schmerl. 2023. Explaining quality attribute tradeoffs in automated planning for self-adaptive systems. *J. Syst. Softw.* 198 (2023).
[18] Rebekka Wohlrab and David Garlan. 2022. A Negotiation Support System for Defining Utility Functions for Multi-Stakeholder Self-Adaptive Systems. *Requirements Engineering* (2022). https://doi.org/10.1007/s00766-021-00368-y.