Collaborative Design Informatics: Leveraging Data to Make Design Teams Better

by

Mark Darryl Fuge

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alice Agogino, Chair
Professor Sara Beckman
Professor Björn Hartmann
Professor Paul Wright

Fall 2014

Collaborative Design Informatics: Leveraging Data to Make Design Teams Better

by

Mark Darryl Fuge

Abstract

Collaborative Design Informatics: Leveraging Data to Make Design Teams Better

by

Mark Darryl Fuge

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Professor Alice Agogino, Chair

Both inside corporations and in self-organized online communities, globally distributed groups of thousands of people now collaborate together on design projects over the Internet. This changes the nature of product design, creating potential for new levels of innovation and product development speed—for example, developing vehicle designs in less than four months, or implementing new business models for urban revitalization in less than a year. However, the plethora of information created by these communities comes with a price: individuals cannot process all of it in a reasonable time frame. Without a means of harnessing their collective efforts, collaborative design communities can never reach their full potential as engines of design innovation and development. To address this problem, this dissertation applies techniques from data science and machine learning to answer to the following central question:

> How can online design communities effectively use the design data they generate to help manage their operations and improve their designs?

Specifically, it presents examples around particular design communities (OpenIDEO and HCD Connect), and some of the challenges they face: How do you maintain a sustainable and creative design community without centralized command? How do designers locate the most relevant or creative inspirations out of thousands of ideas? How do novice designers use the community to learn what design methods are appropriate for a given problem? How can you scaffold novice designers within a community so that they can meaningfully contribute without requiring full expert knowledge? By framing these real-world problems through the lens of Network Analysis, the Maximum Coverage Problem, and Recommender Systems, this dissertation demonstrates how modern machine learning techniques can ameliorate the issues community members face in practice.

From an computational perspective, it finds that the complexity of solving many distributed design tasks necessitates not only looking at a design itself, but also how it is situated in a human community; human relationships play as big a part in predicting a design's success as the content of the design itself. From a human perspective, data-assisted techniques can adapt to human behavior in ways that improve the collaborative structure of large teams, the relevance of methods used for design problems, and the number and variety of ideas that need to be explored by a designer.

The dissertation's findings imply that customizing data science techniques to take advantage of the socially embedded nature of design benefits designers and scientists alike, not only by making design teams more effective but also by providing deeper insight into how humans design the way they do. They point to a future where data-driven design tools are not just a means to an end, but a critical part of how we understand our own needs and creations; where science can be applied, not just to the creation, but to the process of creation.

Acknowledgements

```
With thanks to those who blew the wind,
And those who sailed the ship,
We sailed it tight against the tide,
And I shall be forever in your debt.
```

- Kirk Jones

This poem reminds me of the varied roles people have played in bringing this thesis to the light of day:

```
With thanks to those who blew the wind,
```

I would never have set sail, nor come as far as I have, without the unwavering support of those who have pushed me along. My advisor, Alice, as brilliant a tactician as ever I have met, taught me how to look beyond my research, to seize the trade winds when luck turned my way and make the most of them. You were just the kind of mentor that I needed. I am also grateful to my committee members: Sara, whose unparalleled clarity of mind was the best compass for navigating the mists of the dissertation that I could have hoped for; Björn, whose tenacity for exploration into uncharted, intersecting fields inspires me to stay at the wheel; and Paul, whose willingness to support all my last-minute requests was an absolute godsend. As my guide into the waters of industrial research, Tolga Kurtoglu showed me how Design Theory folks can make a place for ourselves in the applied world and opened my eyes to whole fields of research. I am indebted to the Department of Defense's National Defense Science and Engineering Graduate (NDSEG) Fellowship program, which gave me the financial resources necessary to complete my Ph.D., and also to the National Science Foundation (through grant IIS-0856098) which supported me during my first two years at Berkeley.

Much of what I know about effective research, I owe to Burak Kara, without whose training I would not have succeeded at Berkeley. You have an uncanny way of raising my spirits from unfathomable depths to soaring heights, and were it not for your many kind words and seemingly endless faith in me, I would have turned back long ago. Likewise with Susan Finger, who called me to the harbor in the first place: your vision over the horizon is either lucky or grounded in wisdom I don't yet understand (thus far all signs point to the latter). Most of all, to my parents, Dennis and Alyson Fuge, who have propelled me as far and as steadfastly as they were able to. If I ever aspire to do anything great in the world, it is in effort that I may give as great as I have received.

```
And those who sailed the ship,
```

All the wind in the world would mean little without the brave and merry band of heroes that have stood by me all these years. First of all, to the many undergrad researchers who have taken to the oars and sails in pursuit of my many research directions: without you it would have been a slow moving journey indeed, and working with you has been the highlight of my graduate career. In particular, I want to thank Josh Stroud, Bud Peters, and Kevin Tee, whose blood, sweat, and tears lies embedded in many of the coming pages.

And beneath deck, I was grateful to have had the company of unforgettable men and women to share stories and commiserate with during stormy times: the members of the Berkeley Institute of Design; the BEST Lab; Edwin, Boris, and the rest of the ChemEs; Orianna and the EECS crew; and Jen Lawrence. Thank you for making the journey better than the destination. To Mike C., Shy, and the rest of the crew at Kitchen On Fire, thank you for showing me the light of a great hobby and keeping me sane over the past few years—of all the skills I have learned during my Ph.D., I have a sneaking suspicion that the ones you showed me will ultimately be the most useful.

A large part of chapter 3 came out of collaborations with Nathan Maton at IDEO. I am grateful to him for sharing his time and efforts to make my research as relevant to real-world problems as possible. In addition, thanks go to the many OpenIDEO community members whose actions laid the groundwork for that chapter, as well as the HCD Connect user community who I study in chapter 4.

Lastly, thanks go to Celeste, who was both the wind and the sail. At times, a muse, guiding the bow of the ship. At others, a well-timed gust, carrying me out of the doldrums. Even sometimes a ruthless storm, jettisoning the flotsam and jetsam that weighed down my thoughts and writing so that I might emerge lighter, faster, and with a clearer sense of direction. You have made me a better person.

I wouldn't have reached this distant shore without the tireless help of all these wonderful people, and the remain lines sum up what my own words cannot:

```
We sailed it tight against the tide,
And I shall be forever in your debt.
```

For my parents,
Dennis and Alyson Fuge,
in memory of Harold Fuge,
the original Fuge engineer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

On January 14th, 2013, the U.S. Department of Defense offered up an unlikely challenge to the American public: design the next generation Amphibious Combat Vehicle—entirely over the internet. Over 200 teams and 1,000 participants, many of whom had never met, signed up and competed together for a chance at a $1,000,000 prize. When the competition closed, three months later on April 22nd, there were several complete chassis designs that fully satisfied all of the Marine Corps' stringent requirements.

Now, to be fair, these teams had a bit of help: DARPA provided advanced, web-based Computer-Aided Design (CAD) tools [28] and a multi-university-developed modeling language called META [75, 66] to help teams test different designs and collaborate in real time. However, the designs themselves were never the end goal; DARPA wanted to prove a point that freelance crowds, given the right tools, could produce viable designs at a fraction of the time and cost that it typically takes a defense contractor.

1.1 The Rise of Online Design Communities

DARPA is not the only organization to invest in these online communities. Others (including those in table 1.1) have used online communities to design solutions that address large-scale, pandemic problems, such as climate change, poverty, and public health—complex problems that benefit from a design process that can scale to that scope. Starting around 2002, ThinkCycle, one of the earliest product-focused, crowd-design platforms by Sawhney *et al.* [114] at MIT, appropriated web technologies such as messages boards to bring together designers interested in creating sustainable technologies. In the decade that followed, thanks to enhanced software, availability of rapid manufacturing, and the burgeoning "Maker Movement," society has seen

Name	*Goal*	*Local/Global*	*Novice/Expert*
Design For America	Social Impact	Local	Novice
OpenIDEO	Social Impact	Global	Mixed
VehicleForge	Defense	Global	Expert
Instructables	Education	Global	Novice
99Designs	Graphic Design	Global	Expert
Local Motors	Automotive	Mixed	Expert
Challenge.gov	Policy	National	Mixed
Marblar	Technology	Global	Mixed

Table 1.1: Present day online design communities operate across a broad range of spectrums: from hyper-local to completely global; from physical products to policy; from novices to experts

an explosion of online design communities in all shapes and sizes (*e.g.*, Table 1.1). They now range across a spectrum of industries, expanding the breadth of existing companies as well as defining new non-profit services for social impact.

On one end of the spectrum, Marblar first partners with existing patent-holders, such as NASA and the University of Pennsylvania. They then encourage their online design community to create new product ideas using the provided patents. Once Marblar selects the most promising ideas, they pass the ideas over to Samsung, who gives members of the online design community a portion of the profits once Samsung takes the product to market. In this case, both patent holders and product manufacturers can use online design communities to expand existing offerings or capabilities.

On the other end of the spectrum, OpenIDEO partners with sponsors (typically non-profits, such as USAID or Amnesty International) to post challenges around social problems (*e.g.*, reducing malnutrition or improving maternal health). Participants in their online design community then share collected user stories, prior work, and other inspirations, to come up with thousands of solutions that address the challenge. These solutions range from products to services to business models, and even to government-level policy agendas. In many cases, OpenIDEO participants, with or without the help of the sponsors, have successfully taken these solutions into the real-world, creating new businesses, mobile applications, or community programs.

So why would these companies and organizations chose to invest in online design communities? Proponents cite several reasons:

Cost: hosting a temporary online design challenge with a thousand participants costs significantly less than paying for the same number of man-hours within a company.

Time: thousands of people can parallel process ideas, reducing lead-time compared

to smaller teams.

Diversity: the internet can reach a more diverse set of people across a variety of dimensions—cultures, locations, ages, and industries—widening the potential variety of solutions produced.

Scale: with appropriate software infrastructure, companies can tackle projects much larger in scope by using online communities that scale to sizes unheard of in traditional corporate design teams.

Online design communities are not always the answer for certain types of industries. For example, those with tightly controlled intellectual property or regulatory environments (*e.g.*, medical devices), or those where highly specialized engineering and R&D departments need to be efficiently integrated (*e.g.*, aircraft jet engine development). However, when appropriate, these communities can provide a wealth of useful designs beyond what any individual or company could muster on their own. Will these online design communities eventually become product development behemoths, supplanting corporations like GE or Apple? Probably not. However, they are changing how companies view the role of crowds in product development: companies like Proctor and Gamble, General Electric, IDEO, and Samsung have partnered with online design communities to maximize the value of new ideas, (*e.g.*, Innocentive or P&G Connect+Develop) as well as companies' existing products and services.

1.2 A Wealth of Design Data: Promise or Peril?

An online design community's strength lies in its numbers: thousands of participants can together create thousands of ideas that build upon one another. As with Innovation Tournaments [129], these communities might not generate better ideas than traditional design teams, on average, but the higher number and variance allows members to occasionally generate truly exceptional ideas.

However, their strength is also their Achilles' heel: a single member no longer has the bandwidth to collaborate with or build upon the thousands of ideas that the community generates. Moreover, without efficient ways to search through or filter the community, members cannot even make smart choices about what subset of ideas they should focus on. Essentially, they are searching for a needle in a haystack, slowing the community's potential to develop more innovative designs. This leads to the key question addressed in this dissertation:

> How can online design communities effectively use the design data they generate to help manage their operations and improve their designs?

Figure 1.1: The structure of this dissertation breaks down the role of design community data into nested parts: at the broadest level, it studies design communities, how they form, and what makes them effective; within communities, it studies how individuals and teams use and learn design processes; within processes, it studies how particular ideas are formed and evaluated.

1.3 Thesis Overview

To address the question of how to support individuals in locating appropriate design data, the dissertation focuses on two particular real-world communities, OpenIDEO and HCD Connect, and breaks down the issues they face into three units of analysis, moving from largest to smallest in scale (Fig. 1.1): Community-level, Process-level, and Idea-level. For each of those units, the chapters identify the problems that OpenIDEO and HCD Connect face when the volume of available design data goes beyond what participants can handle themselves. They then present algorithms designed to overcome those problems, and provide some insight into how to use that volume of data to benefit the community. Moreover, they demonstrate how different data analysis techniques can be taken beyond analyzing existing designs, to helping individuals generate new ones. Specifically, each chapter covers the following:

Chapter 2: Background reviews prior literature from a variety of relevant areas, including: Crowdsourcing, Machine Learning, Information Retrieval, Network Analysis, Design Theory, and Creativity. It places the subsequent chapters in the context of previous research and demonstrates the gaps in knowledge that this dissertation fills.

Chapter 3: Design Communities studies the structure and operation of design communities themselves, specifically: what keeps members engaged in a community, how do these communities grow over time, and what incentives or interventions enable better collaboration for idea generation? It covers OpenIDEO, an online design community that addresses large social issues such as malnutrition or maternal health. Specifically, it applies tools from Network Analysis and Graph Theory to study OpenIDEO's evolution over time and how that affected attributes such as their idea generation potential or user engagement.

Chapter 4: Design Processes studies the design processes that individuals use as they participate within a community, specifically: how does one identify the general design process used by community members, without requiring them to explicitly encode their design process? It shifts focus to a different design community, HCD Connect, which operates a repository of design case studies for design process communication and education. The problem facing this community is that the number and complexity of the stored design cases exceeds the amount that an individual member can reasonably read, limiting its usefulness at sharing knowledge. The chapter demonstrates how techniques such as Spectral Clustering, False Discovery Rate Control algorithms, and Collaborative Filtering can alleviate many of those concerns.

Chapter 5: Design Ideas studies how to evaluate the ideas that design communities produce, specifically: how does one evaluate the creativity in a way that scales to tens of thousands of ideas? The chapter casts this problem as a type of non-linear function approximation task using a connection to a class of mathematical objects called *submodular functions*.

1.4 How to Use This Dissertation

This dissertation is structured as a story around design communities and the problems that they face. If reading the entire dissertation, one should read each chapter sequentially. However, certain subsets of the dissertation may interest particular audiences:

Computer Scientists looking to apply their algorithms to new domains. Sections 2.1 and 2.2 present relevant background information on applying crowdsourcing and machine learning to design domains, respectively. Chapters 3 and 4 provide actual design datasets for Network Analysis and Recommender Systems, respectively, which could serve as possible benchmarks. Researchers working in Active Learning would be particularly interested in chapter 5, as it deals with a situation where human-labeled

data is expensive and highlights many future research avenues between the Design and Active Learning communities.

Those looking to understand how to manage or influence design communities. Chapter 3 deals specifically with managing online communities, particularly from the viewpoint of Social Network Analysis. The material in 3.1 and 2.1 covers prior work on Network Analysis and Crowdsourcing, respectively—essential areas when managing large online design communities.

Design Educators looking to incorporate data-driven strategies in their teaching. Chapter 4 discusses how to extract design patterns from human behavior for educational purposes. This includes a description of a design method case study repository, along with a link[1] to download any code you would need to visualize how different design methods from IDEO's HCD Toolkit relate to one another. For evaluating design creativity in a classroom setting with many submissions, you would also find chapter 5 and the accompanying code useful, as it specifically addresses how to scale up creativity evaluation with limited expert input.

Designers looking to increase their toolkit and creativity by understanding crowdsourcing and algorithms that might help them. Chapter 5 addresses tools for evaluating new ideas. Sections 2.1 and 5.1.1 provide relevant background material on crowdsourcing and creativity, respectively, which both play a role in idea generation within online design communities.

Researchers looking to reproduce or build off the results of the dissertation. Each chapter has links to download any datasets or experiment code, or you can go to the dissertation companion site to download the set.[2] Everything necessary to reproduce the experimental results is open source or otherwise freely available.

[1] http://www.markfuge.com/hcdconnect
[2] http://www.markfuge.com/dissertation

Chapter 2

Background

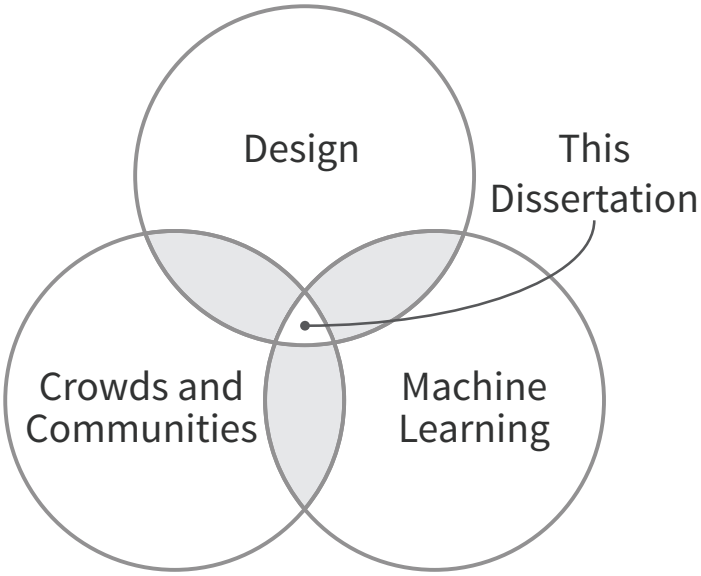


Figure 2.1: This chapter reviews background work in three areas: Design, Crowds and Communities, and Machine Learning. The dissertation fills a gap of knowledge at the intersection of these areas.

Before this dissertation can dive into how to help online design communities deal with the data they generate, it first needs to delineate how it fits into what past research has accomplished. Each of the remaining chapters covers more specific related work relevant to itself (such as Network Analysis, Recommender Systems, or Creativity), but this chapter shows which major research areas this dissertation sits between, to provide some context and demonstrate related fields that might benefit from the remainder of the dissertation. This chapter breaks relevant background

down into three broad categories: Design, Crowdsourcing and Communities, and Machine Learning. Since each of those areas is massive, in and of itself, this chapter will cover their intersecting pieces (see Fig. 2.1), and show how this dissertation fills a gap of knowledge at the intersection of all three areas.

2.1 Design, Crowdsourcing, and Online Communities

While this dissertation covers online design communities focused on product and service design, entire fields of research have evolved around utilizing crowds for other forms of productive and creative work (*e.g.*, the Human Computation Conference), as well as in methods by which computers can support them (*e.g.*, the Computer Supported Collaborative Work conference). Quinn and Bederson [102] provide a taxonomy (adapted in Fig. 2.2) that categorizes some of the different ways that groups of people can collectively create value. They dissect crowd-based research into several overlapping areas, aggregating several definitions from a variety of past literature:

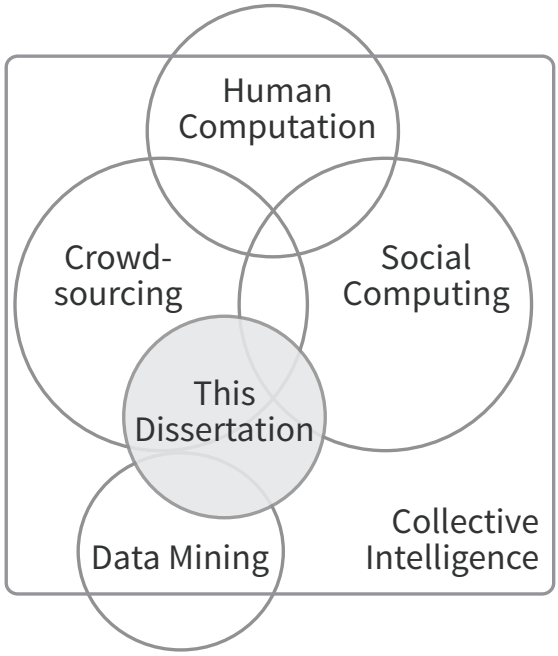


Figure 2.2: Within the taxonomy of Quinn and Bederson [102], this dissertation applies research from across data mining, crowdsourcing, and social computing, depending on the specific purpose of a design community.

Collective Intelligence: "groups of individuals doing things collectively that seem intelligent."

Human Computation: "a paradigm for utilizing human processing power to solve problems that computers cannot yet solve."

Crowdsourcing: "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call."

Social Computing: "applications and services that facilitate collective action and social interaction online with rich exchange of multimedia information and evolution of aggregate knowledge."

Data Mining: "the application of specific algorithms for extracting patterns from data."

One interesting thing to note about Fig. 2.2 is that Quinn and Bederson posit no overlap between Data Mining and Crowdsourcing, Social Computing, or Human Computation. They view the purpose of data mining as philosophically separate since data mining does not refer to the collection of data, only the subsequent processing [102, pg. 3]. Within their taxonomy, this dissertation bridges across data mining, crowdsourcing, and social computing, depending on the specific design community and the purpose of a particular algorithm. In chapter 3, it focuses on the intersection of social computing and crowdsourcing within design communities, while chapter 4 applies concepts primarily from data mining to extract patterns in design method use, and chapter 5 uses both crowdsourcing and data mining toward creativity evaluation.

The rest of this section reviews prior research in two sub-areas particularly relevant to this dissertation: crowds for specific creative tasks and crowds as members of a community.

2.1.1 Crowds for Specific Creative Tasks

Many of the typical examples of crowdsourcing systems focus on fairly analytical (though computationally complex) tasks: labeling images, translating text from one language to another, extracting information from scanned forms. In contrast, design is a highly creative and generative activity. This section will review past crowdsourcing efforts focused on other generative tasks such as writing and creating visualizations that are more informative.

Writing is a skill that requires both deep analysis as well as delicate synthesis. Bernstein *et al.* created a crowdsourcing system called Soylent, which helps crowd

workers take a piece of writing and edit or re-write the piece to a desired length [10]. They scaffold workers by splitting a complex task (writing correctly and concisely) into a series of smaller tasks (finding passages, fixing those passages, and verifying that the edits are correct) that are distributed across many people. In doing so, they are able to avoid many pitfalls of using crowd workers, such as lazy or inaccurate workers and high variance in work quality. In the context of composing customer reviews, Dow *et al.* [31] demonstrate that the quality of crowd work can be improved by a "sheparding" process, wherein the original worker (self-assessment) or other members of the community (external assessment) could provide feedback, with the possibility of reviewing and editing the review afterwards. They find that in both the self-assessment and external assessment cases, the quality of work improves, with external assessments leading to a greater amount of revision. In this dissertation, chapter 3 describes how OpenIDEO also breaks their design process into stages to segment an otherwise difficult design activity, and uses a formal feedback process to help community members improve their designs.

Like with writing, composing accurate visualizations is a difficult task. Heer and Bostock [56] use crowd workers to evaluate graphical perception. They replicate a series of seminal experiments in graphical perception by using members of Amazon's Mechanical Turk platform to rate various diagrams. For example, they compare bubble charts with tree maps by asking workers to estimate relative areas using each diagram and record which diagram leads to more accurate estimates. Their work is similar to that of Horton *et al.* [62], who show that Amazon's platform can reliably replicate several economic and psychological experiments that were previously observed during in-person lab studies. In the area of visual design, Xu *et al.* [133] demonstrate that crowds can generate more structured crowd feedback around a design (*e.g.*, specific feedback on visual elements, or asking which parts of a design one notices first) that is more useful than simply whether they "like" or "dislike" a design. Chapter 5 describes how these kinds of approaches allow distributed experts to be combined to assess design creativity.

2.1.2 Crowds as Members of a Community

In many of the examples above, crowd members worked largely independently; when they did receive external feedback, it was anonymous and without a means of further collaboration. However, design is inherently a team sport. Most crowd-based design takes place in a more social and collaborative setting, where one is not only concerned about the work, but also the social interactions among individuals.

Dow and Settles [32] study music collaborations at an event called "February Album Writing Month" (FAWM) . In FAWM, songwriters from around the world can come together in the month of February to produce "14 songs in 28 days." Song-

writers can post song snippets, comment on each others' music, and even set up live collaborations over the internet with each other to try out different ideas and co-compose songs. They also [116] create a path-based logistic regression model that can predict which particular artists might collaborate together, using factors such as how well-connected an artist is, their message history, and the music genre. They found that communication pathways within the community and complementary skill sets, such as different instruments, were important predictors of future collaboration.

Designing a product or service often requires a special set of skills, and thus the people who participate in the community can be particularly important. In certain instances, the desired participants may be spatially localized: for locating people with computer science expertise, Heirmerl *et al.* [57] retrofitted a vending machine with a computer interface for grading computer science exams, and placed it in the computer science building at U.C. Berkeley. By placing the interface where content experts were more populous, they avoided problems associated with online platforms, such as Mechanical Turk, where the vast number of members might not be qualified for the task.

Once an operating community is established, the mechanism for transferring knowledge between members must still be worked out. Within the crowdfunding domain, Hui *et al.* [63] study how more experienced community members provide mentorship to less experienced members to help them successfully run Kickstarter campaigns. They find that community members assist at various points through the creation and execution of the fund-raising campaign: from providing example campaign materials to publicizing the campaign within their social network to providing manufacturing support or expertise. A vast majority of this communication is done through separate channels (*e.g.*, Skype, Email), and Hui *et al.* call for better integrated support systems for promoting this kind of mentorship.

In this dissertation, chapter 3 tackles how OpenIDEO establishes their online community, while chapter 4 introduces data-driven mechanisms for supporting knowledge sharing among community members.

2.2 Design and Machine Learning

Design, as a field, stretches across a wide variety of domains, and Finger and Dixon provide an early review of the beginnings of the field with respect to Mechanical Engineering [36, 37]. Machine Learning is likewise diverse, and those looking for a thorough introduction should see the comprehensive reviews in textbooks by Murphy [86] and Bishop [11]. This section focuses on their intersection: machine learning approaches to the design of new products and services, since that is the intersection that relates to this dissertation. To review the appropriate literature, it will first

review some basic concepts from machine learning (such as classification, regression, clustering, *etc.*) to provide some common context and background before diving into the major machine learning approaches from two fields relevant to this dissertation: Mechanical Engineering Design and Computer Graphics.

2.2.1 Overview of Machine Learning Strategies and How They Apply to Design Problems

In general, most researchers divide machine learning problems into one of two types: supervised or unsupervised. **Supervised learning** refers to a problem where there are pre-known labels for the data and the algorithm's task is to correctly predict the label for a new data point. It is typically broken down into two types: classification and regression. Classification would include problems with a discrete number of options, like email spam filtering where an incoming email is either spam (*e.g.*, an advertisement) or ham (*e.g.*, an email from your friend). Regression would include problems with a continuous scale like temperature prediction where the goal is to predict what the temperature will be next Tuesday. In either classification or regression, there is a correct answer provided to the algorithm, and its goal is to maximize its correctness on future data. Applied to design problems, supervised learning can be used for scalable design creativity evaluation (Chapter 5) and recommending design methods (Section 4.5).

In contrast, in **unsupervised learning** the goal is not to predict a particular label, but rather to group or cluster the data into categories. The purpose of these groups is to discover new knowledge by locating categorizations within the data that might have been unknown previously. For example, Walmart might use data about which items were purchased together to segment its customers or find complementary product lines. Applied to design problems, section 4.5 demonstrates how unsupervised learning can be used to group related design methods (Section 4.4).

Given a type of problem (*i.e.*, supervised vs unsupervised), there are two broad classes of approaches that can solve that problem: generative or discriminative models. Typically, generative models are used for unsupervised problems, whereas either generative or discriminative models would be used for supervised problems. **Generative models** describe the joint probability between the object of interest (y) and the data (x) (*i.e.*, $p(y, x) = p(y|x) \cdot p(x)$), while **discriminative models** only describe the conditional probability between the object and the data (*i.e.*, $p(y|x)$).

In this dissertation, chapter 4 uses unsupervised generative models for clustering design methods and supervised generative and discriminative models for method recommendation. Chapter 5 demonstrates how supervised generative models can approximate design creativity evaluation.

2.2.2 Machine Learning within Mechanical Engineering Design

Within Mechanical Engineering Design, algorithms that help generate new products and services typically fall under the umbrella of "Computational Design Synthesis." They decompose into three main approaches: agent-based systems, grammar-based systems, and evolutionary systems. Representative work from **agent-based design** systems is the work of Campbell *et al.* [19] which gives hundreds of different software agents different tasks and constraints, and then proposes a two-stage iteration procedure whereby some of the agents create a design and then passes it off to a different set of agents that modify the design and passes that design on, and so forth. For example, they present an electro-mechanical design problem in which a set of agents need to design different configurations of a bathroom scale. One type of agent is responsible for interpreting the user requirements and selecting a possible set of components. These components are then passed to a different type of agent who selects and sizes components, based on costs or other constraints. Simulating this back and forth between many agents results in a set of usable designs.

In contrast to constantly simulating and tweaking designs using agents, grammar-based approaches attempt to define a **design grammar**, similarly to the way the English language uses a grammar. For example, English is made up of certain phrases, such as noun phrases (*e.g.*, "The cat"), and verb phrases (*e.g.*, "slept on the bed"), and if you know that noun phrases always come before verb phrases, then you can correctly generate an English language sentence by randomly selecting phrases and putting them in the correct position (*e.g.*, "The cat slept on the bed"). This idea extends to design languages as well: if you can define a single set of relationships and phrases that describes a design, you can generate as many valid designs as you want by simply selecting phrases from that language. For example, McCormack *et al.* devised a shape grammar (relations between geometric objects) that emulated the style of the Chevy Buick automobile [85]. With this grammar, they could simulate different kinds of cars that had the same design style as the Chevy Buick: small SUVs, sports cars, *etc.* Many of these grammar systems derive from earlier work in pattern languages from architecture [3], which assumes that a design is structured in terms of formal patterns that one can derive and then use to reproduce new designs. These grammars can be deterministic (certain rules are always followed), or stochastic (rules follow others with certain probabilities), such as in Campbell *et al.* [20] where it is necessary to traverse a tree of possible design rules and learn which rules are more likely to follow each other.

Lastly, **evolutionary algorithms** have been adapted for design, where initial designs are mutated and bred together in order to eventually produce better designs. The way evolutionary algorithms work, is by first defining the "DNA" of a design (*e.g.*, a vector of numbers that controls the shape of a car profile), and then that

DNA is changed slightly (*e.g.*, adding a few inches to the roof of the car), and bred with a different design, producing many new designs that share different copies of their parent's DNA. New designs are then evaluated by some measure of fitness (*e.g.*, their aerodynamic drag, or how many people said they liked it), and only the top performing designs are allowed to reproduce in the next round. Evolutionary algorithms are a popular choice when a human has to make a decision regarding which design to choose. In these cases, a human selects the designs that get bred, and slowly, over time, the designs shift toward what the human likes. This approach is called Interactive Genetic Algorithms (IGA), and you can see examples of it in the work of Cho [25] and Poirson [100]. Gu *et al.* instead train an Artificial Neural Network using some human feedback, and then use that network as a means to evaluate possible designs [55].

2.2.3 Machine Learning within Computer Graphics

Within Computer Science, computer graphics and interface design are the two relevant areas that use machine learning to aid in design tasks most familiar to Mechanical Product Design—chiefly computational geometry. The ability to quickly model physical objects using computational geometry has become increasingly important as direct digital manufacturing equipment (3D printers) has reached mass-market appeal. As in Mechanical Engineering, shape grammars are one popular solution for geometries that can be procedurally generated. For example, Talton *et al.* [124] define a grammar-based modeling technique based on a reversible Metropolis-Hastings sampling technique, essentially allowing them to sample different designs from normally incompatible grammars. Teboul *et al.* [128] demonstrate how shape grammars can be inferred by example using Reinforcement Learning, while Talton *et al.* [125] demonstrate grammar induction using Markov Chain Monte Carlo (MCMC). They show examples of generating building facade and architecture, respectively. In the domain of webpage design, Kumar *et al.* [72] demonstrate that by inferring design grammars over web page designs, you can determine common patterns that novice designers can use to create new webpages or search for creative inspiration.

Aside from grammars, researchers have also used properties of the geometry itself to help explore and generate new designs. Chaudhuri *et al.* [24] compute shape histograms and density functions for different shapes and compare them to one another. By calculating which portions of the shape are similar and different, they can segment different parts of a shape and mix-and-match parts across objects. For example, given different airplane designs, they are able to take the wings of a fighter jet, and place it on the body of a jumbo jet, without having the user manually edit the geometry. Similarly, Orbay *et al.* [94] use a spectral decomposition of a geometry to determine "high-frequency," or rapidly changing geometry, from "low-frequency" geometry. This

allows them to isolate the basic shape of an object from its detailed features, permitting recombination and brand identification; for example, they demonstrate that users identify a Ford mustang through its front bumper, rather than other features.

The above applications from both Mechanical Design and Computer Graphics are similar to this dissertation in that all of them use large collections of user-submitted data as a basis to ultimately help novice designers create better designs. Specifically, some of the spectral decomposition techniques described above will be revisited in chapter 4 for design method clustering.

2.3 Crowds and Machine Learning

Researchers have explored various ways of using both crowds of humans and machines to complement each others' strengths, reducing human effort and cost as well as improving algorithm performance. This dissertation covers that same fundamental goal when applying this intersection to design communities.

Quinn *et al.* [103] proposed the CrowdFlow computation model, wherein a user can choose with what speed and level of accuracy they want a task performed. If a user wants high accuracy, then their system relies more on additional human verification, but if they want fast speed, then it lets automated machine learning algorithms do the bulk of the work (even if their accuracy is less). This reduces cost and leverages the comparative strengths of the human and computer.

In addition to using humans alongside of a machine learning algorithm, researchers have used humans *inside* learning algorithms. In Sorokin *et al.* [121] machines can elect to ask a human to label a certain picture for them, thus receiving a new piece of labeled data. This gives the algorithm the option of asking for help on difficult cases. Embedding humans even further, Gomes *et al.* [50] uses the crowd to classify images as similar or dissimilar to one another (*e.g.*, these images are both bags, or these images are dissimilar because this was taken in a forest and this other one inside a building). They then compare the human evaluations with a generative statistical model to predict clusters of objects, and are able to find hierarchical classifications within the images. Likewise, Tamuz *et al.* [127] uses pairwise similarity triplets (*e.g.*, is object A more similar to B or C?) to construct a similarity matrix, or "Crowd Kernel" that can then be used by any number of machine learning classifiers, such as a Support Vector Machine.

In this dissertation, chapter 5 returns to the trade-offs between human and machine computation when translating preferences into design creativity ratings.

The following chapter starts at the community scale, with an extensive study of OpenIDEO. It uncovers what a design community is, how it forms, and what its

structure means for generating ideas.

Chapter 3

Managing Online Design Communities

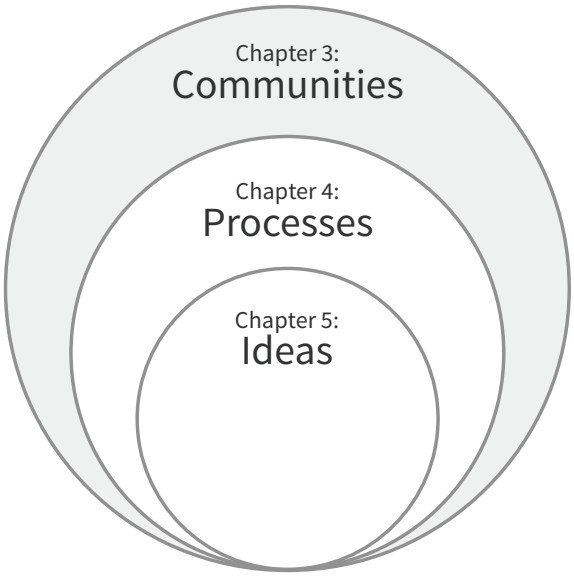


Figure 3.1: This chapter addresses the role of data at the design communities level: how communities form, and what makes them effective.

This chapter studies the structure and operation of design communities themselves, specifically: what keeps members engaged in a community, how do these communities grow over time, and what incentives or interventions enable better col-

Portions of this chapter appear previously in [46, 42]

laboration for idea generation? As a particular example, it focuses on OpenIDEO, an online design community that addresses large social issues such as malnutrition or maternal health by using collaborative design and user research. Specifically, it demonstrates how to apply tools from the field of Network Analysis and Graph Theory to study the origins of OpenIDEO as well as attributes such as its idea generation potential or user engagement.

In relation to other collaborative design networks, OpenIDEO differs in the following ways: it focuses on applying human-centered design to both product and service design; the users come from a variety of design backgrounds, including industrial design, engineering, business, and arts; and the challenges focus on large-scale social problems, rather than specific technical challenges seen in engineering-centric competitions (*i.e.*, DARPA's FANG challenges). This chapter uses OpenIDEO from among other possible online design collaboration platforms (*e.g.*, Napkin Labs, frogMob, VehicleForge, *etc.*) due to the breadth of project types, the large user community, and availability of collaboration meta-data (such as explicit links between concept ideas). The review by Lakhani *et al.* [74] contains additional information about OpenIDEO's governance, such as how it runs the challenges, the type of design challenges it hosts, how it operates within IDEO, and the type of participants using the platform.

The first half of the chapter conducts the largest network analysis of an online design community in the literature to date, and several findings emerge. First, OpenIDEO's social structure differs markedly from other online social communities in a key factor: it has negative assortativity (*i.e.*, the tendency for the most collaborative people to work with the least collaborative people, called *disassortativity*). This unexpected behavior leads to properties that benefit idea generation, such as high network efficiency and low clustering. The chapter identifies possible interventions that might be at cause for this, and makes some recommendations for design managers.

The second half of the chapter views OpenIDEO as a dynamic network, and focuses on how its community structure and communication within the larger network evolve over time. This provides insight into how users participate, and what role their actions play in developing OpenIDEO's network properties. The chapter finds that a persistent core of users in OpenIDEO has developed over time, and that their disassortative connections to transient members within the network seem to be driving the observed network. The vast majority of users actively participate after joining the site, but their participation drops significantly on subsequent challenges, particularly when those challenges do not overlap in time (*i.e.*, when there are gaps between challenges). An analysis of usage patterns across users shows that commenting behavior differentiates the high-activity users from the low-activity users; this suggests implementing commenting incentive strategies to increase engagement.

3.1 Background on Network Analysis

Researchers from many disciplines, from sociology to computer science, to business, have studied online communities, using both qualitative and quantitative research techniques. They apply qualitative techniques when the specific behaviors by a small number of particular individuals are the topic of the research; for example, trying to understand the collaborative strategies or background of users who win the OpenIDEO challenges. To determine community-wide behavior in groups such as OpenIDEO, researchers prefer quantitative techniques because they can be cost-effectively used on larger networks and provide a complementary picture of the community structure compared to that gained by qualitative studies. These quantitative methods are typically based on graph theory and are referred to as **Network Analysis**. This section first reviews the basics of Network Analysis, and then breaks down prior studies into two types: Theoretical (*i.e.*, mathematical) and Empirical. These aspects provide the background necessary to discuss the main results of this chapter, which center around the network properties of OpenIDEO and its effects on ideation.

3.1.1 Basics of Network Analysis

Network Analysis is a class of mathematical techniques that studies particular types of complex phenomena. Its primary assumption is that a phenomenon can be reasonably modeled as a mathematical graph consisting of nodes (or *vertices*) connected to each other by links (or *edges*). For example, in a social network such as Facebook, a node might be a person, a link might be the strength of a relationship, and the phenomenon of interest might be how a viral video propagates among people over time. By representing phenomena as graphs, network analysis can adapt measures from graph theory in order to explain or predict certain behaviors, ranging from disease transmission to scientific co-authorship to protein interactions. Researchers have used Network Analysis for decades, and the textbooks by Newman [90] and Scott [115] provide an exhaustive review.

The most critical assumption in any network analysis study comes from how the network nodes and links are defined. However, once the nodes and links have been defined, one can compare several graph properties, both on a global (whole network) level, and at a local (node-centric) level that provide insight into the behavior of the network. For example, graph properties can be used to predict qualities like the social power of individuals, weak-points in information flow within networks, or the likelihood of co-authorship between researchers.

The below alphabetical list defines some commonly used terms from Network Analysis that appear in later explanations:

Assortativity (also called *assortative mixing* or *homophily*) is the propensity for

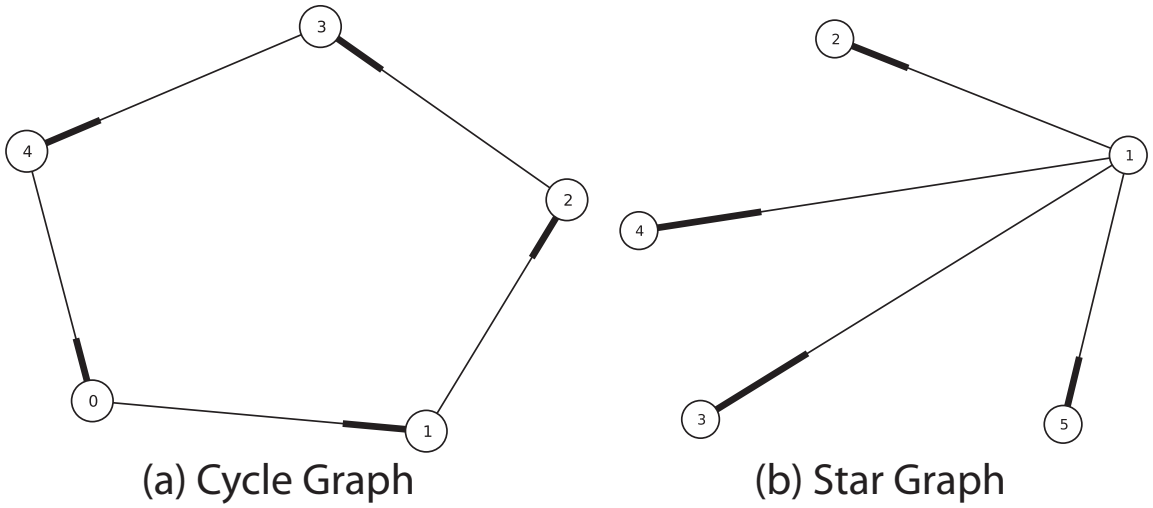


Figure 3.2: Directed links are represented by a thicker segment indicating the direction (*e.g.*, in (a) 2 points to 3, and 3 points to 4).

nodes in a network to create links with similar nodes, and to avoid creating links with dissimilar nodes. For example, engineers might be more likely to be friends with other engineers than with dentists, and vice versa.

Degree assortativity, means that nodes with high degree (those who communicate with many people) are more likely to communicate with other nodes with high degree, instead of nodes with low degree (those who communicate infrequently). Social networks are known for being positively degree assortative [90, Sec. 7.13].

Centralization refers to how well the graph is centered around a single focal point on a scale from zero to one. High centralization would imply a deeply hierarchical structure, such as a star graph (*e.g.*, Fig 3.2b), while low centralization would imply that all nodes are equally central, such as a cycle graph (*e.g.*, Fig 3.2a).

Clustering Coefficient is a measure of how tightly connected nodes are in a graph, specifically measuring the ratio of number of triangles between a node and any two neighbors and the number of possible triangles (*e.g.*, how many of your friends are also friends with each other) [90]. Node-wise clustering coefficients can be averaged to characterize how clustered a graph is as a whole.

Connected Component is a subset of the nodes in a graph that can be reached by following links between them. For example, if two nodes are connected to each other, but not to any other nodes, then they form their own connected compo-

nent. Real-world networks typically have one large connected component which contains most of the nodes (*e.g.*, the center component of Fig. 3.3b contains around 95% of the nodes), followed by some smaller components with only a few nodes each (*e.g.*, the single nodes on the outside of Fig. 3.3b) [90].

Degree of a node measures the number of incoming and outgoing links to that node. For example, in Fig. 3.2(b), node one has a degree of four because it connects to four other nodes.

Degree Distribution refers to the fact that different nodes have different degrees. The distribution of these degrees follows different patterns depending on the type of network structure. In many real-world networks, this distribution is **power-law distributed** (or *scale-free*), which means that it exhibits a relatively linear plot when plotted on a log-log scale.This corresponds to many nodes having a few links, and only a few nodes having many links.

Density is the ratio between the number of links that exist between nodes and the maximum number of possible links that could exist (*i.e.*, a complete graph). In large real-world networks, the density is typically low [90].

Diameter is the length of the shortest path between the two farthest nodes in the graph. It provides a sense of how spread out the graph is and provides one measure of the resistance to the flow of information.

Efficiency measures how easily and quickly information is transferred across a network. It is inversely related to the average shortest path length required to go between all pairs of nodes on the graph; if efficiency is high, all nodes are within a few links of one another, and if efficiency is zero then no node can communicate with any other node (*i.e.*, there are no links between nodes).

k-Clique is a set of k nodes that are all connected to one another (*i.e.*, they form a complete sub-graph). For example, if A knows B and C, and B also knows C, then A, B, and C are a 3-clique.

Link (also called an edge) is a connection between two nodes on the graph. It can have a direction as well as a weight (*e.g.*, if person A sent person B ten emails, that would correspond to a link directed from A to B with weight ten).

Size is the total number of nodes in a graph.

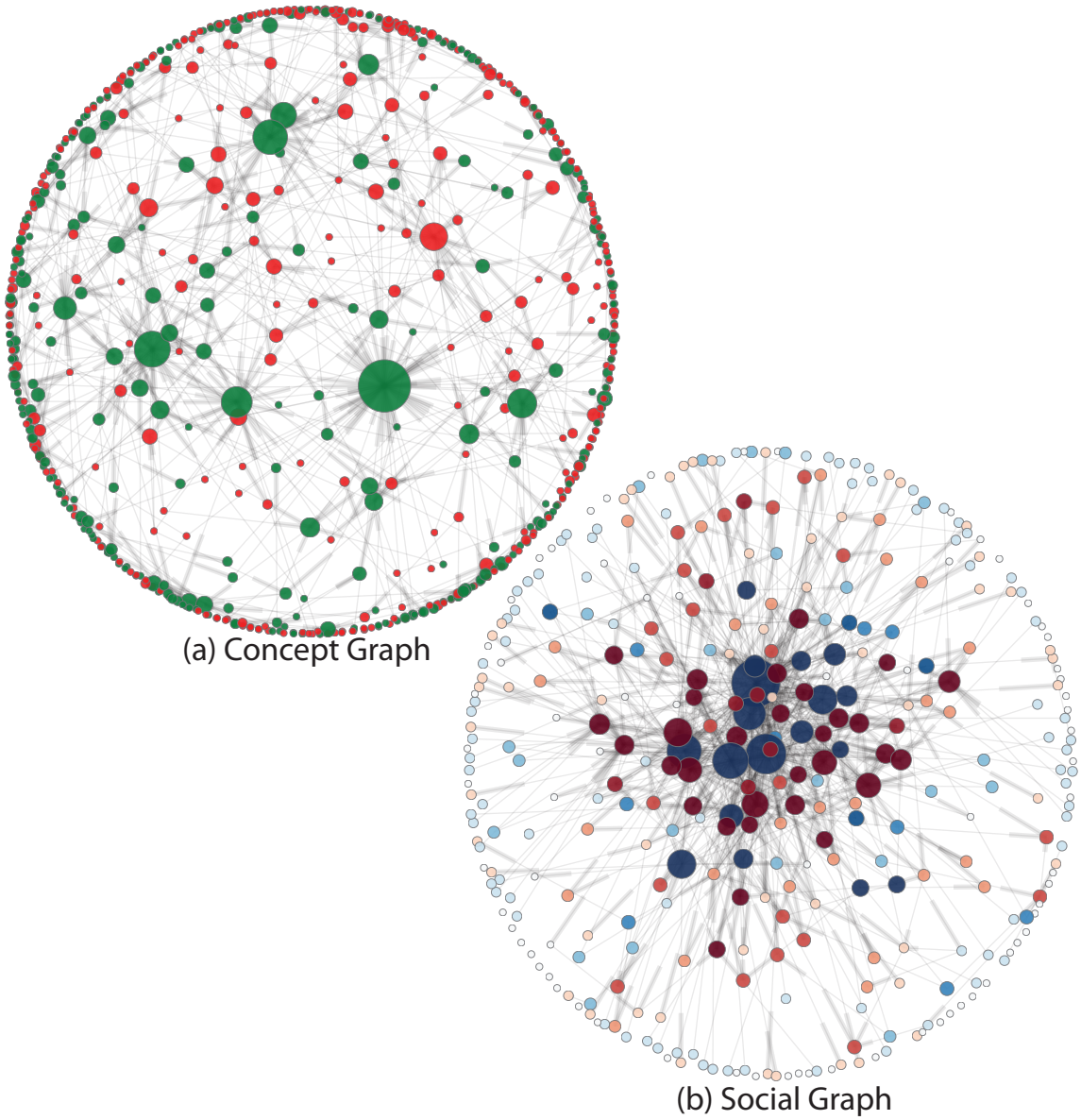


Figure 3.3: (a) is a concept graph, where red nodes represent inspirations and the green nodes represent concepts. (b) is a social graph, where redder nodes indicate more comments are received than given, whereas bluer indicates the opposite. In both cases, the size of the nodes represent the degree (number of incoming and outgoing links) of the node.

3.1.2 Theoretical Network Models

While there has been limited empirical work on actual design networks, prior research has proposed different *theoretical* models for how design networks might operate. The vast majority of available theoretical models for collaborative design networks are either: 1) simulation studies using agents with predefined collaboration rules, or 2) studies that compare common mathematical models, such as Preferential Attachment [90, Ch. 14.1] or Design Structure Matrices [35] to real-world data.

Agent-based simulations typically define a computational model of a product, and then create a series of software agents who can choose what portion of the product to work on. For example, Panchal [97] simulates a product architecture (he does not mention an actual product) that has 9 different modules, each of which depends on a subset of the other modules. This simulation then tracks the product and communication between the agents, building a simulated collaboration network that can then be analyzed for structural properties [136, 97, 76]. The typical applications for this line of work are in identifying potential strategies for managing complex system design, under the assumption that the agents behave similarly to real people.

In contrast, other studies attempt to take real network data and fit mathematical network models to that data [117]. The key assumption behind that line of work is that if actual networks obey certain properties, such as power-law distributions, one should be able to match a power-law distributed mathematical model to that data. Upon doing so, insights are often gained about why the network does or does not conform to theoretical expectations. For theory behind time-varying theoretical models, Csermely *et al.* [27] and Castellano *et al.* [23] provide good reviews.

3.1.3 Empirical Network Studies

Within collaborative design networks, prior research by Le *et al.* [77] studies the network structure of both Open Source Software and Hardware networks. In particular, they model the RepRap Fused Deposition Modeling (FDM) project as a network of hardware components. They differ from this chapter in that they only address technological networks, while this chapter also address the social evolution of the design community. Panchal considers social factors [97, 98], but bases the findings primarily on computer-simulation studies addressed above in section 3.1.2. In contrast, this dissertation provides empirical evidence from a large, real-world design community. As a complementary qualitative study for OpenIDEO, Gordon *et al.* [53] demonstrates how ideas that leverage Human-Centered Design techniques have a higher chance of winning the design challenges.

In terms of **how a network's structure affects idea generation**, both Mason *et al.* [84] and Stephen *et al.* [123] independently found that higher local clustering

around a node (*i.e.*, when all your neighbors are also neighbors with each other) reduces their idea generation ability, due to "complex contagion"—the tendency to copy your neighbors when they all say similar things. Essentially, they demonstrate via human experiments that if your immediate collaborators are also connected to each other (high clustering), bad ideas can fixate the entire group on a poor solution. In contrast, networks with high efficiency (low average distance between all nodes) but low clustering do not suffer from this "group think," while still being able to spread good ideas rapidly throughout the network.

Outside of design networks, three types of previously studied empirical networks are similar to those considered in this dissertation: social networks, open source software development networks, and scientific co-authorship networks. They each have elements one would expect to find in a collaborative design network, and therefore serve as a meaningful basis for comparison.

Social networks, such as social media or email networks, are similar to OpenIDEO in that a traceable process of social communication occurs between participants through actions such as providing feedback on ideas. Social networks tend to be highly positively assortative with multiple smaller communities of people interacting together [90, 112, 130], and Kossinets and Watts [69] observe that this behavior grows over time. Social networks tend to possess power-law distributed degree distributions, though there are notable exceptions (*e.g.*, Facebook [130]).

Open Source Software (OSS) is similar to open design networks in that the members are typically decentralized, can choose which projects they want to work on, and are creating some artifact that will be used by people; it is different in both the kind of project as well as the specific mechanisms of collaboration. In most studies of OSS, the node unit of analysis is a particular developer and a link exists between developers if they have worked on the same project together [80]. These networks display high assortativity and often generate many smaller communities, particularly around programming languages. They also possess standard power-law distributed degree distributions that are typical of many social networks [134]. Looking at OSS development over time, Saraf *et al.* [110] demonstrate how assortative mixing—where developers form new links with those most similar to them—increases over time.

Research co-authorship is another type of network where there is formal interaction and the goal is to generate new ideas in collaboration with others. It differs from OpenIDEO in that the barriers to collaboration in OpenIDEO's case are smaller than for research co-authorship, and the online social interactions in OpenIDEO are traceable in a way that is not feasible in research networks. Like OSS networks, co-authorship networks are also positively assortative, tend to form communities within the larger network, and have a low average clustering coefficient [90]. Barabasi *et al.* [7] present a representative empirical analysis of how scientific collaborations evolve over time; they note how collaborations follow a familiar power-

law distribution and that the clustering of the community tends to decrease over time.

3.2 Analysis of OpenIDEO's Design Network

To understand both the general properties of the OpenIDEO network as well as the properties of any sub-communities, this section divides its analysis into three parts: 1) Structural measures that address information flow within the network; 2) Community measures that address the network's robustness and community structure; and 3) Effects of certain members that address the specific role of possible OpenIDEO interventions.

3.2.1 Data Collection and Pre-processing

The data consist of the 22 OpenIDEO challenges that were completed by November, 2013.[1] Most design challenges start with a question, for example "How might we restore vibrancy in cities and regions facing economic decline?" Each challenge consists of a series of sequential phases: Inspiration, Concepting, Applause, Evaluation, Selection of Winners, and Realization. During Inspiration, contributers can submit "insights, examples, stories, or comments"[2] designed to provoke possible solutions from the community. During Concepting, contributors submit concepts designed to solve the challenge. Inspirations and concepts are typically a few paragraphs long with accompanying figures.

In either stage, contributors can link to other people's inspirations or concepts by clicking a "build off of this idea" button in the web interface (similar to the "forking" behavior on software repositories such as GitHub [78]). This creates an explicit link used to model interrelations between the submissions by constructing a graph where nodes represent a submission in the particular design challenge, and a link exists from node A to B if concept B builds upon concept A (*e.g.*, Fig. 3.3). This section refer to this as the *concept graph*, with separate concept graphs for each of the 22 challenges.

During all stages of the challenge, people may post comments on other people's concepts as well as reply to comments on their own concepts. To model this social interaction, a separate graph (the social graph) represents an OpenIDEO user as a node, and adds a weighted directed link from user A to user B if user A comments on user B's concept for that particular design challenge or if user A replies to a comment given by user B. The content of these comments can be positive, negative, or neutral in tone, and does not have to relate strictly to submitted designs, since the goal here

[1] http://www.openideo.com/open

[2] http://www.openideo.com/how-it-works/full.html

Quantity	*Min.*	*Mean*	*Max.*
Avg. # of participants per challenge	119	372	820
Avg. # concepts per challenge	189	506	1220
Avg. # of actions per user	1.41	4.09	7.24
Avg. # of concepts per user	1.00	1.41	2.29

Table 3.1: A summary of the OpenIDEO corpus statistics averaged across each of the 22 challenges.

is to model the social transactions between individuals (though a more qualitative study of the comment content would be an interesting extension of this work). Every additional interaction from A to B adds an additional unit of weight to the link between A and B. There are separate social graphs for each of the 22 challenges.

Table 3.1 lists some basic statistics averaged across the 22 challenges in corpus. The average number of users and concepts are both of an order between 100 and 1000. The average number of actions per user and the average number of concepts submitted per user are both of an order between 1 and 10. Later in the chapter, section 3.3.3 provides more detail on the specific actions of these users.

There are particular users, who this section refers to as *OpenIDEO community managers*, that have specific roles on the platform: they help facilitate the challenge by reaching out to many concepts and commenting on them to promote interaction. There are usually two of these managers per challenge, one who is a member of IDEO staff and another who is an active member of the larger community.

3.2.2 Structural Measures

The measures in this section are designed primarily to address the ease with which information flows through each network. Greater size and diameter imply that information has farther to travel, while greater clustering, centralization, efficiency, and density imply greater ease of information transfer.

Comparing the concept and social graphs across challenges reveals certain key structural similarities and differences: Despite similar network sizes, the two networks have drastically different link structures, diameters, densities, and average clustering. Figures 3.3a and b illustrate representative concept and social graphs, respectively, positioned using a Fruchterman–Reingold force directed layout algorithm; immediate inspection reveals the tightly clustered core-periphery structure of the social graph (Fig. 3.3b) as well as the sparser, more community clustered concept graph (Fig. 3.3a). To make the differences between these structures clearer, Fig. 3.4 highlights several key similarities and differences:

Size (Figure 3.4a): Within each challenge, both the concept and social graphs have approximately between 200-800 nodes, with two concept graphs reaching into the low 1,000s—these sizes are fairly small compared to social internet communication networks, but large compared to the size of typical collaborative design groups. Particularly for the concept graphs, this size indicates that it would be time-prohibitive for any individual to actually read through all available ideas in a challenge.

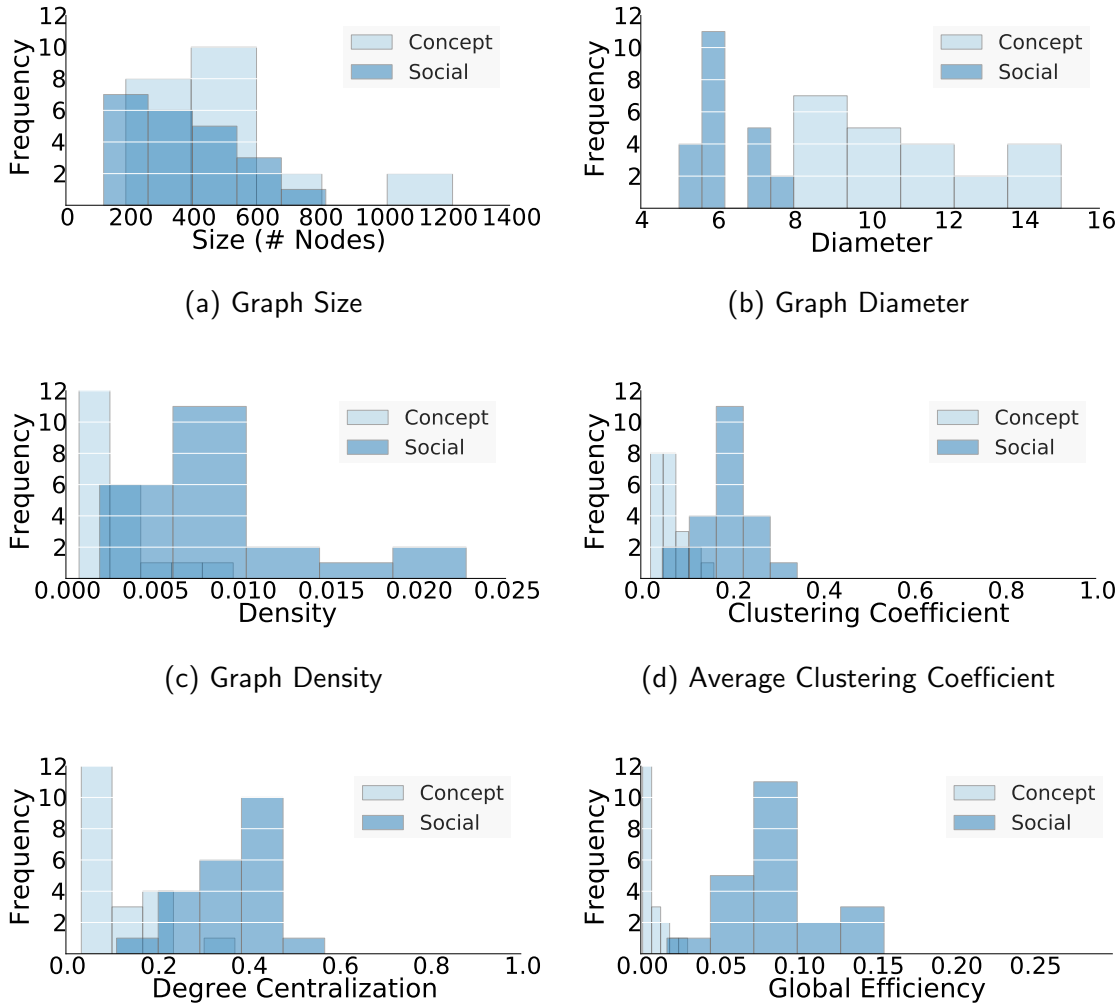
Diameter (Figure 3.4b): Since both types of networks in OpenIDEO have disconnected components (thus infinite graph diameters), it is more reasonable to measure the diameter of the largest connected component of the graph. For that case, OpenIDEO's social graph has a significantly smaller component diameter than that of the concept graph, despite their being of roughly equal size. Part of the reason for this smaller diameter is the fairly efficient center of the social graph (Fig. 3.3b) which bridges many nodes, decreasing the distance information needs to travel and making communication and feedback easier to transmit.

It is notable, though not unexpected, that both the concept and social graph have disconnected components. This indicates that there are concepts that are never being built off of and users who are not participating in the social community, both of which are losses of potential information.

Density (Figure 3.4c): The social graph is approximately four times as dense as the concept graph. This means that there are a higher proportion of connections (and thus less disconnections) between most users, whereas, on average, 42% of the inspirations or concepts that get submitted are never built off of (or at least tagged as such on the website). The concept graph's low density is possibly due to the number of available concepts and the effort required to build off of an idea.

Average Clustering Coefficient (Figure 3.4d): As expected, the sparsely connected and spread out concept graph has low average clustering, while the social graph has higher clustering, roughly comparable to other social networks. In the concept graph, higher clustering would mean several ideas are similarly related, whereas in the social graph higher clustering would mean groups of similar social connections and communication.

Centralization (Figure 3.4e): Figure 3.4e demonstrates that both the concept graphs and social graphs are decentralized, with the concept graphs having significantly less centralization. Both of these results match what one would expect from an open innovation platform: many users should have access to different parts of the graph in order to have exposure to diverse groups of ideas and people. Part



(a) Graph Size

(b) Graph Diameter

(c) Graph Density

(d) Average Clustering Coefficient

(e) Degree Centralization

(f) Global Efficiency

Figure 3.4: The concept graphs have higher diameter(3.4b) and lower density (3.4c) than the social graphs, despite roughly equivalent network sizes(3.4a). This is possible due to small levels of clustering within the concept graph, and the fact that the social graph has certain mechanisms built in that reduce the graph diameter. The concept graph exhibits low centralization(3.4e) and low global efficiency(3.4f), while the social graph exhibits medium centralization and low efficiency. In both cases, higher efficiency would be more advantageous in order to ease transfer of ideas and feedback, respectively. Figure 3.3 provides some visual intuition behind these results.

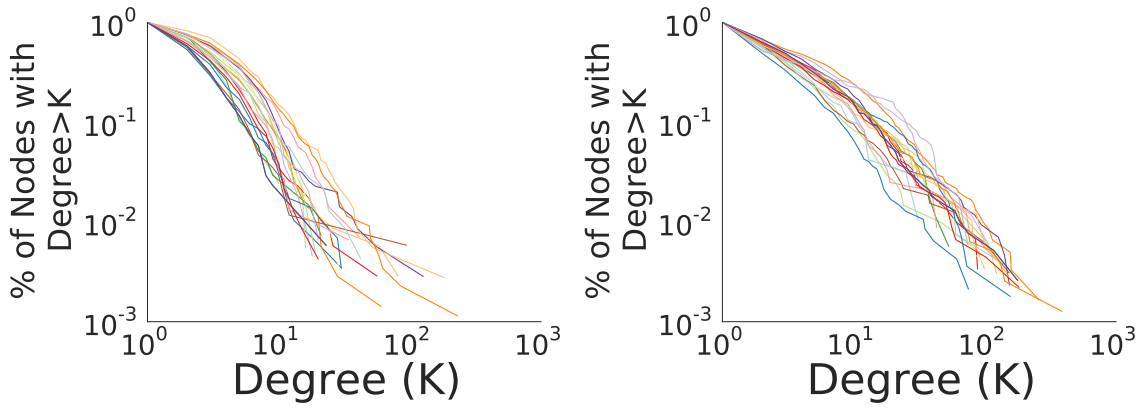
of the increased centralization in the social graph comes from the presence of the OpenIDEO community managers who are well connected to many members of the social community—a point section 3.2.4 explores in depth.

Efficiency (Figure 3.4f): Figure 3.4f illustrates the global network efficiency of both the concept and social graph. The low efficiencies in the graphs come with both advantages and disadvantages: on the one hand, lower efficiencies mean higher network redundancy and robustness at a given density—the concept graph has both low density and low efficiency, so it doesn't gain the redundancy benefit, while the social graph's central core structure does. However, at a given density, higher efficiencies create better information transfer across the network and also correlate to lower clustering—this is a useful structure when groups of people have to collaboratively solve uncertain problems together without getting stuck [84]. Section 3.4 returns to these ramifications after the next section discusses the role that community structure plays in these two types of networks.

3.2.3 Community Measures

To understand the type of community structures inherent in the OpenIDEO network, this section conducts three types of analysis: (1) degree distribution, (2) assortativity, and (3) community detection using the k-clique percolation method [96]. The results were unexpectedly different than other networks of their type: the social graph is highly disassortative with only a single, large core structure, while the concept graph has many small communities. For the social graph, this unique structure gives it higher robustness under node removal than standard social networks, and its disassortativity likely helps it maintain that structure. Both of these are advantageous properties for an open innovation network where participation is voluntary because one would like the design collaboration network to continue to transfer information when someone stops participating (node removal), and to maintain connections with the larger number of users who are on the periphery and only participate occasionally (disassortativity).

Degree Distribution (Figure 3.5): Both the concept and social graphs appear power-law distributed, due to the linear nature of the degree distributions in Fig. 3.5 (power-law distributions appear linear when plotted on a log-log scale, as in Figure 3.5). In terms of robustness, power-law distributed networks are robust (*i.e.*, do not change much) under random node removal (*i.e.*, random people leaving the network), but are particularly susceptible to targeted node removal (*i.e.*, removing the highest degree or most important individuals) [90]. However, as demonstrated below,



(a) Concept Graphs (b) Social Graphs

Figure 3.5: Degree complementary cumulative distribution functions (CCDF) for the largest connected component of different types of Open IDEO networks. Each line corresponds to a different challenge. Both types of networks are generally power-law distributed.

OpenIDEO possesses a core-periphery structure that mitigates this robustness concern [27, 106]; even removing several of OpenIDEO's highest degree members (the OpenIDEO community managers) does not significantly alter the network properties.

Assortativity (Figure 3.6): One possible reason for the social graph's robust core-periphery structure lies in the network's lack of assortativity. Figure 3.6 compares the assortativity of the OpenIDEO concept and social graphs, where assortativity ranges from 1 (completely assortative) to -1 (completely disassortative).

Unlike other social networks (see Table 3.2), the OpenIDEO social graph is actually *disassortative*, meaning that those members who communicate frequently are actually communicating more often with infrequent members of the group rather than frequent members, and vice versa. Indeed the directed links in Fig. 3.3b display a balance between outsiders commenting on concepts generated by members within the core, as well as core members reaching out to those on the periphery. This balance is one hypothesis for the disassortative, core-periphery structure seen in the social graph. Other possible reasons include: OpenIDEO's reputation system, which awards "collaboration points" for commenting with other people's concepts; community managers who reach out to less active users; specific stages of the design process for commenting, viewing, and evaluating the work of others; and soft incentives from IDEO that reward active users through possible job opportunities within the larger company.

While these features undoubtedly improve participation, collaboration, and disassortativity, they may not be present in other design networks. These results suggest

Type of Network	*Assortativity*
OpenIDEO	−0.413
Student Relationships	−0.029
Email address books	0.092
Mathematics Co-authorship	0.120
Biology Co-authorship	0.127
Film Actor Collaborations	0.208
Company Directors	0.276
Physics Co-authorship	0.363

Table 3.2: A comparison of Assortativity between OpenIDEO and other typical networks. All numbers for non-OpenIDEO social networks are reproduced from [89, Table II]. Assortativity is measured on a [-1,1] scale. The only other disassortative network is in student dating relationships (where students reported dating members of the opposite gender more than their own gender).

encouraging this disassortative behavior as a means to increase network efficiency, decrease clustering, and improve idea generation—a recommendation that section 3.4 returns to in greater depth.

K-Clique Percolation (Figures 3.7 & 3.8): To uncover any possible community structures, this section uses the Clique Percolation Method [38] to detect communities of different sizes and overlap. It is a widely-used community detection method that can identify an unspecified number of communities where k specifies how interconnected the community should be—higher k values mean smaller, more densely connected communities, while lower k values would create larger more loosely connected communities. It works by constructing k-cliques and then merges k-cliques together if they share k-1 nodes in common, identifying larger communities. For example, a 2-clique would be any 2 connected node pair, and a 2-clique community would merge any pairs which shared at least 1 node in common—this special case would be the same as finding the connected components of the graph. By increasing k, one can uncover increasingly connected communities within the graph.

Figure 3.7 compares the number of k-clique communities for each type of graph as k is increased. The concept graph contains many 3 and 4-clique communities, but none larger than 5. The social graph contains, on average, 1-2 communities, but becomes a well-connected central community as k increases.

To characterize what these communities look like, Fig. 3.8 plots a representative example from challenge 10 that compares the identified k-clique communities as k is increased. In the concept graph, as k increases, several mostly non-overlapping communities form throughout different parts of the graph—this demonstrates patches

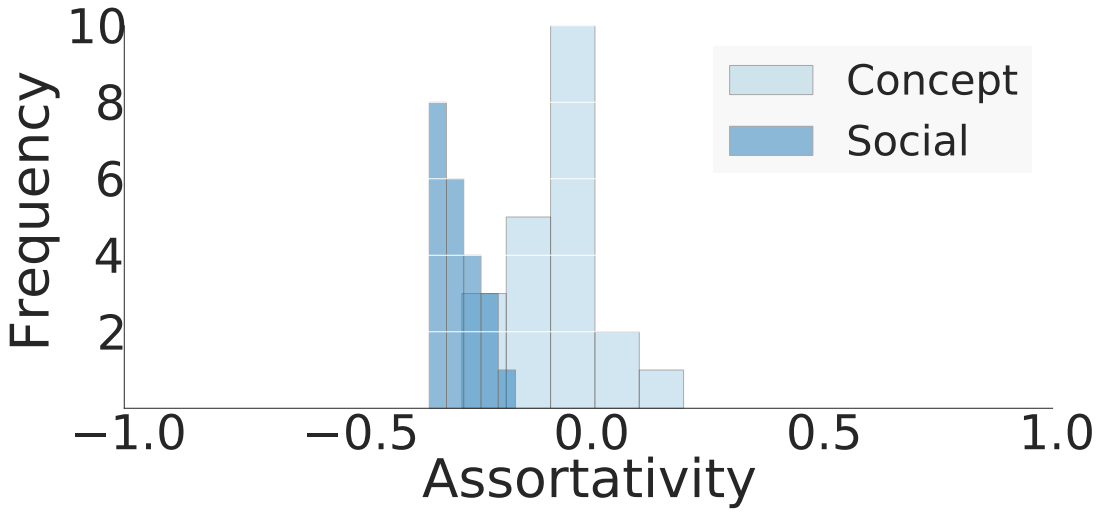
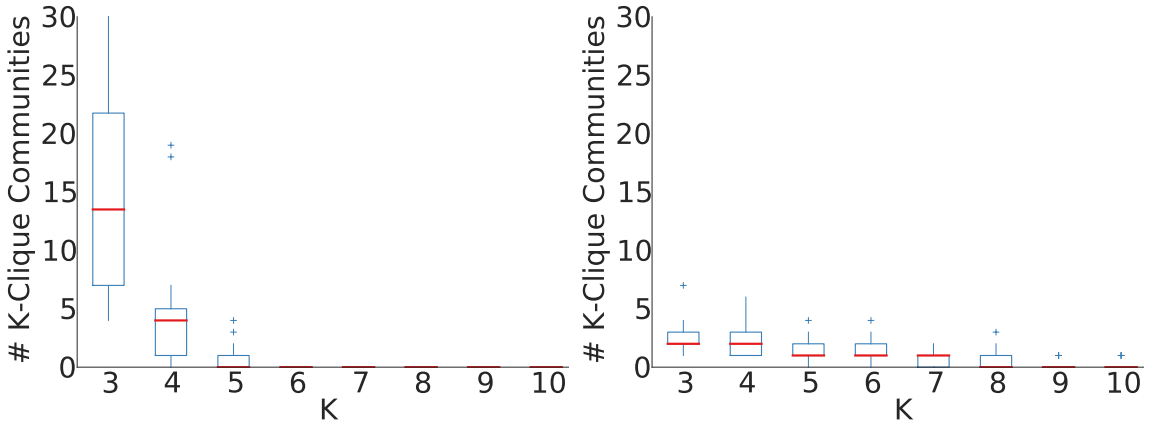


Figure 3.6: Unlike most social networks, the OpenIDEO social graph appears negatively assortative (disassortative) by degree, rather than positively assortative. This means that members with high degree (lots of communication) talk more with those with low degree, rather than with others of high degree. This style of communication is highly atypical of most social networks. It reduces the diameter of the network and increases the fraction of the members in the largest graph component. The concept graph appears neither assortative nor disassortative.



(a) Concept Graphs

(b) Social Graphs

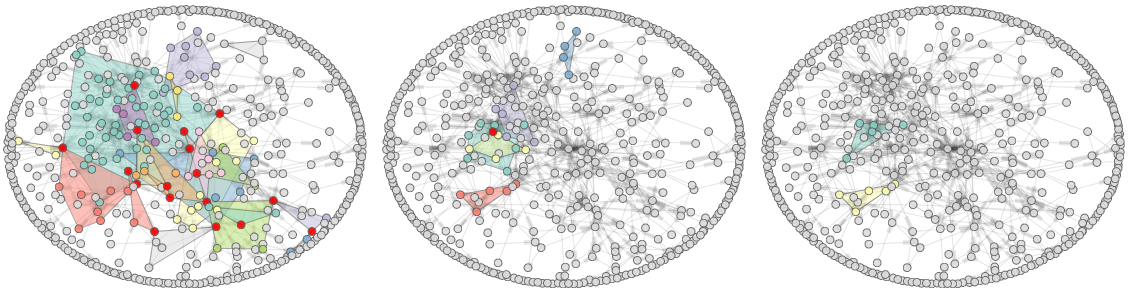
Figure 3.7: Boxplots of the number of communities detected using the k-Clique Percolation Method, for different values of k in both the concept(3.7a) and social graphs(3.7b)[96]. The concept graphs have a high number of small communities, while the social graphs have only a few communities that are significantly more connected. This reinforces the visual data in Fig. 3.8.

of interrelation between small collections of different concepts. In contrast, the social graph starts with a large, central community incorporating most of the network core. As k increases, the core remains, decreasing somewhat in size. Any new communities that form have substantial overlap with the existing central core, rather than forming on a different portion of the graph—this is again consistent with the notion of the social graph maintaining a core-periphery structure.

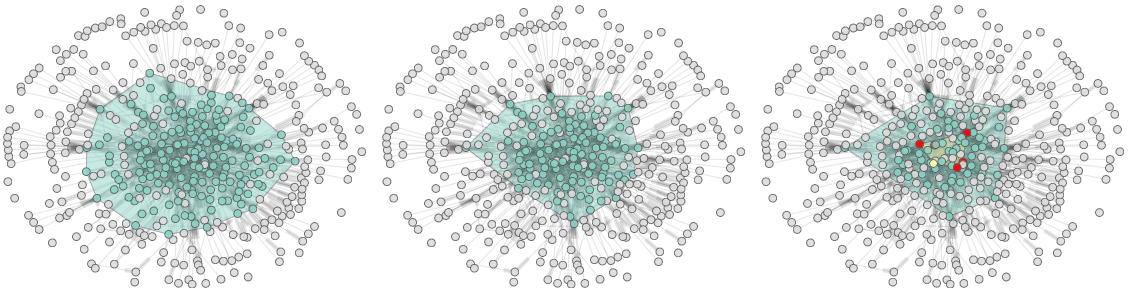
3.2.4 Effect of OpenIDEO Community Managers

One hypothesis for some of the observed behavior is that the OpenIDEO community managers could be purposefully acting within the network to produce these structures, and that removing them from the graph would better resemble a standard social network model. Removing those users, and any of their links, from the social graphs across all challenges and repeating all of the above analyses does not change the most of the results.

Figure 3.9 compares the two most substantive changes: (3.9a) demonstrates that removing the OpenIDEO community managers increases the assortativity of the social graph, though it still remains significantly disassortative; (3.9b) demonstrates that the centralization of the network decreases substantially. Given that the role of the OpenIDEO community managers is to reach out and involve different members, it is



(a) Concept 3-Cliques (b) Concept 4-Cliques (c) Concept 5-Cliques



(d) Social 3-Cliques (e) Social 4-Cliques (f) Social 5-Cliques

Figure 3.8: Visualizing the communities created using the k-Clique Percolation Method, for different values of k in both the social and concept graphs [96]. This uses the networks from challenge 10 as a representative example. The colored sub-graphs represent nodes within a given community, and red nodes represent nodes in multiple communities. As k -increases (*i.e.*, right to left above), the "tightness" or cohesiveness of the community increases. For the concept graphs (3.8a-3.8c), multiple, non-overlapping communities are present at different community scales (k=[3,5]). This signifies different "clusters" of related ideas that are mostly separate from one another (*i.e.*, only a small number of the nodes in a-c are red). In contrast, the social graphs have a single core community that is tightly connected (*i.e.*, still present at higher k-values). Any additional social communities tend to be heavily overlapping (*i.e.*, many red nodes in 3.8f relative to the number of separate communities).

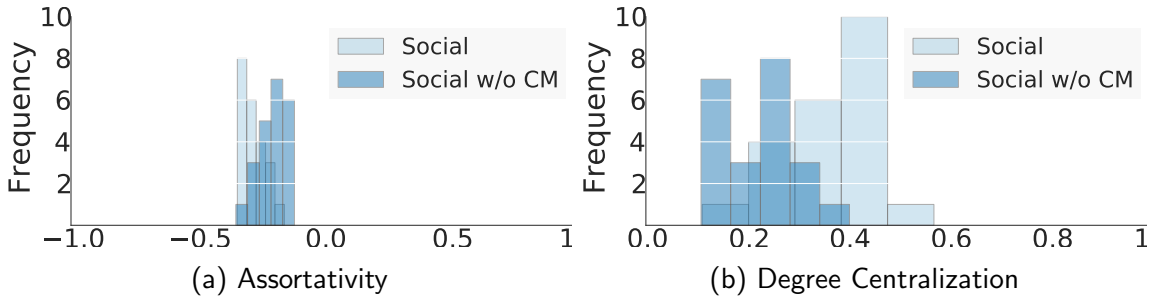


Figure 3.9: Removing OpenIDEO community managers from the social graph ("Social w/o CM"), there are some noticeable, but small changes: the centralization of the network decreases and the assortativity increases. The general behaviors described above are unlikely to be caused exclusively by existing OpenIDEO community managers.

not surprising that their actions change both assortativity and centralization. What *is* surprising is that, even devoid of the community managers' comments, OpenIDEO's social graph remains disassortative and still somewhat centralized. Section 3.4 returns to the implications of this behavior after looking at how OpenIDEO's network as evolved over time.

One caveat with the results in Fig. 3.9 is that it does not eliminate the possible indirect effects of community managers. For example, even if we remove all direct communication links by community managers, they may still have instigated communications between two other users. Even though Fig. 3.9 cannot remove all effects of the community managers, the main takeaway remains clear: the dissassortative behavior in the network links is not solely a function of the community managers' direct actions.

3.3 The Growth and Evolution of OpenIDEO

With an understanding of what the OpenIDEO network looks like and how it operates, this half of the chapter studies how it evolved from its inception to present day. This provides insights into the user behaviors and events that shaped the structure of the community. Analyzing OpenIDEO's evolution requires one addition methodological choice beyond those in section 3.2: how does one model time-dependent quantities, like link strength, using only discrete time events?

To handle temporal events in the OpenIDEO network, this section uses the approach of Palla *et al.* [96], who model each edge weight as having a decay factor:

$$w_{A,B}(t) = \sum_i w_i \exp\left(-\lambda |t - t_i| / w_i\right) \tag{3.1}$$

where i indexes an event between user A and user B. Essentially, this treats the edge weight as a sum of exponentially decaying functions—if the users interact regularly, they will have high edge weight, but if they go months without interacting, their edge weight approaches zero. This section treats all events as equally important by setting $w_i = 1$, and sets the decay rate λ such that w_i is 1% of its original strength after 100 days of inactivity.

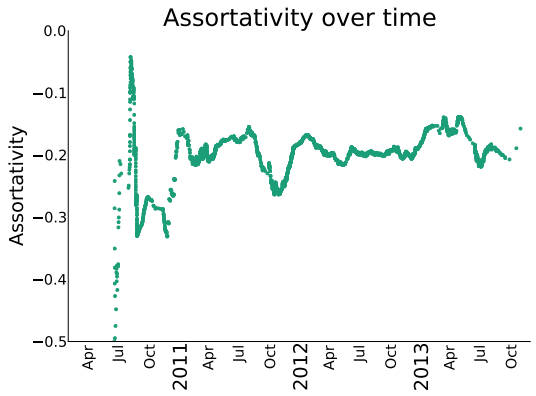
With this added temporal model, one can represent the OpenIDEO network at any given timepoint by summing up all prior user events, and using Eqn. (3.1) to appropriately decay the edge weights. The same k-Clique Percolation Method [38] used above in section 3.2.3 can be used to detect communities within the temporal network.

To set k and the edge weight cutoff in the k-Clique Percolation Method, this section takes the approach described in Palla *et al.* [96] of finding the highest k such that large communities are still able to form, and then reducing the cutoff threshold until it is barely above the value needed to preserve community structure. In physical terms, this cutoff procedure defines a minimum communication frequency that two individuals must overcome to still be considered "connected" to each other. Much like in other social communities, if two individuals stop collaborating the strength of their connection decays over time and an individual would leave a former community once his or her existing ties to that community atrophy. This minimum level of atrophy and the tightness of the resulting community are what the cutoff and k values represent, respectively.

With a temporal version of the OpenIDEO collaboration network, the next section can now present the results, starting at the whole-network level, moving down to community level, and finally to the level of individual users. The results highlight when the network-level disassortative mixing, efficiency, and clustering behavior observed in section 3.2 above started to occur; how community structure develops over time; how users enter and participate in the system; and what particular actions users take.

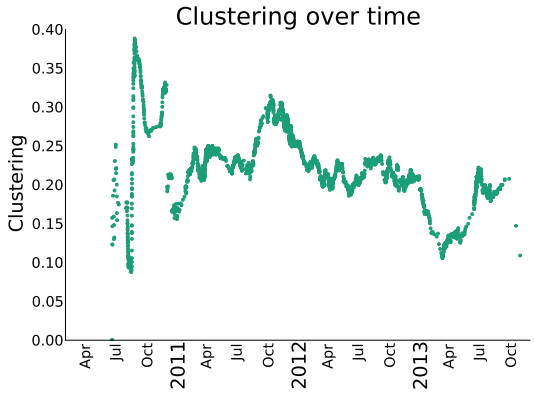
3.3.1 How the Entire Network Evolves

Figure 3.10 plots the assortativity (3.10a), efficiency (3.10b), and clustering (3.10c) of the OpenIDEO network over time. All three properties varied over the entire lifetime of the network, with the biggest variations occurring during the first nine months of OpenIDEO's growth. The network starts and remains disassortative over its lifetime, which means that frequent users collaborate more often with infrequent users than with each other.



(a) The network remains disassortative (negative assortativity) over its lifetime, unlike most other social networks.

(b) Efficiency fluctuates over time.



(c) Average clustering fluctuates over time, with a minor decrease as the network grows older.

Figure 3.10: It took about about 9 months to a year for the system properties to equilibrate.

3.3.2 How the Community Structure Changes Over Time

To visualize how OpenIDEO's community structure has changed over time, Fig. 3.11 summarizes the size and number of communities, and Fig. 3.12 provides visual snapshots of the community structures at various timepoints.

Figure 3.11 plots a point for each community on the x-axis with its corresponding size on the y-axis. For example, early in OpenIDEO's development, only a few small communities existed (Fig. 3.12a: Dec. 10th, 2010). However, around the start of Challenge 3, the user base grew, expanding the communities (Fig. 3.12b: Mar. 23rd, 2011) until they eventually merged into a singe, core community (Fig. 3.12c: Feb. 7th, 2012). After a year, the size of the largest community began to decrease and split into several smaller communities who share some common nodes (the red nodes in Fig. 3.12). As Fig. 3.12g demonstrates, the decrease in community size is not due to lack of membership or partcipation—the core is still actively participating, but has started developing smaller communities within the core, as evidenced by Fig. 3.12h (Jul. 2nd, 2013).

3.3.3 The Lifetime of Community Members

Having looked at the network and community levels, the frame of reference now focuses on the individual users: How long do they stay with the site? How do they spend their time when participating? What makes long-term, multi-challenge participants different than single-challenge participants?

To answer the first question about user lifetimes in OpenIDEO, Fig. 3.13a records the difference between the date a user joins the site and their last date of activity on the site (*i.e.*, when they last submitted a concept or left a comment). Aggregating the data for all 5753 users, it shows a log-scaled histogram of the number of days between joining and last activity; it demonstrates the long-tail of participation, with only a small number of users remaining through several challenges. Figure 3.13a does not account for the fact that certain users joined later than other users, biasing the histogram towards lower participation times. To address this, Fig. 3.13b divides the number of days a user has been active by the total number they could have been active, based on their join date. The story remains the same: the vast majority of participants are transitory visitors, with a central core of committed members.

To further explore how users participate in the design community, Fig. 3.14 shows the three different possible user actions (joining, submitting a concept, or giving a comment to someone) for each user as a function of time. The y-axis represents a particular user id, where the users have been sorted by the date they joined OpenIDEO. It demonstrates not only the user growth pattern over time, but also the general behavior of most users: after joining, the users partake in a frenzy of activity that includes

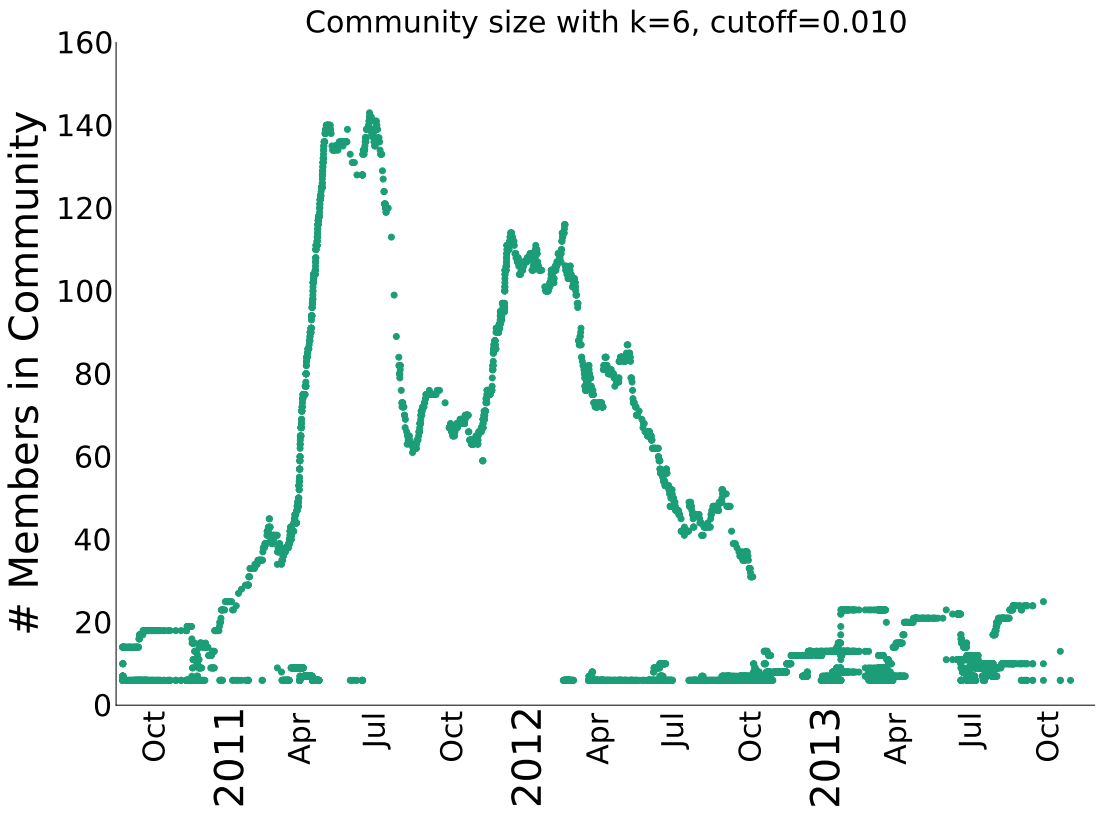
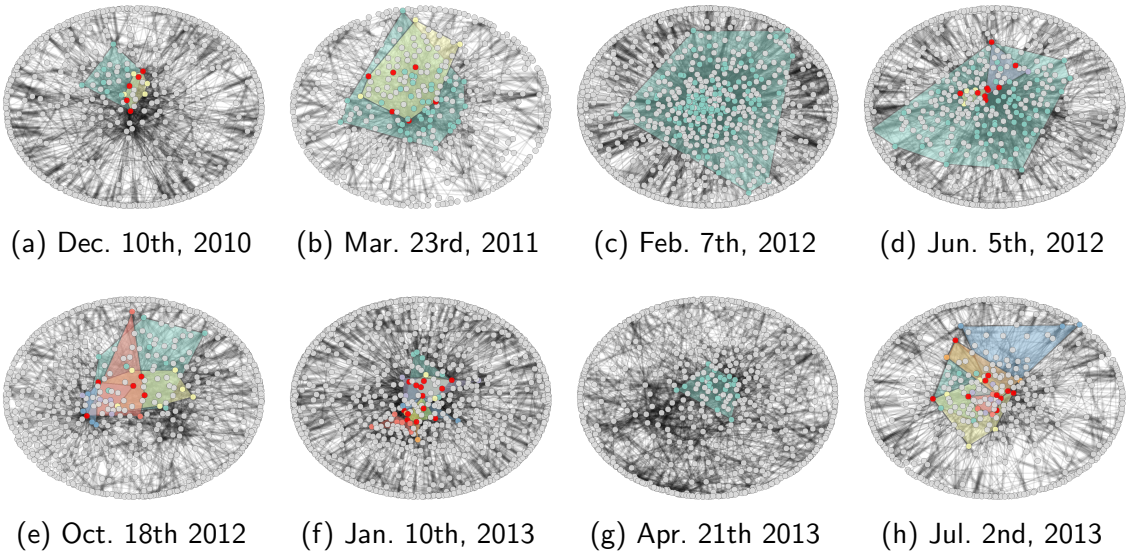


Figure 3.11: OpenIDEO's community structure changes over time, with a single large community emerging from 2011-2013, eventually splitting into several smaller communities all clustered around the central core (Figs. 3.12e-3.12h).



(a) Dec. 10th, 2010 (b) Mar. 23rd, 2011 (c) Feb. 7th, 2012 (d) Jun. 5th, 2012

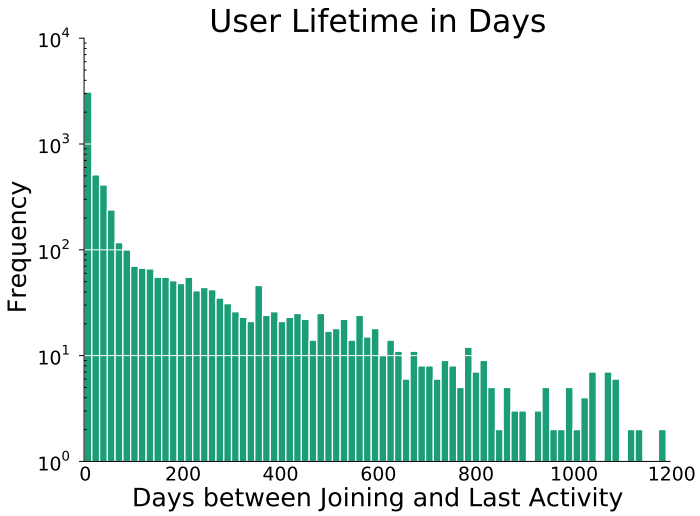
(e) Oct. 18th 2012 (f) Jan. 10th, 2013 (g) Apr. 21th 2013 (h) Jul. 2nd, 2013

Figure 3.12: A series of community snapshots of the OpenIDEO collaboration network over time. The colored polygons and nodes represent different communities, with the bright red nodes representing nodes that straddle different communities. Grey nodes are not part of any community.

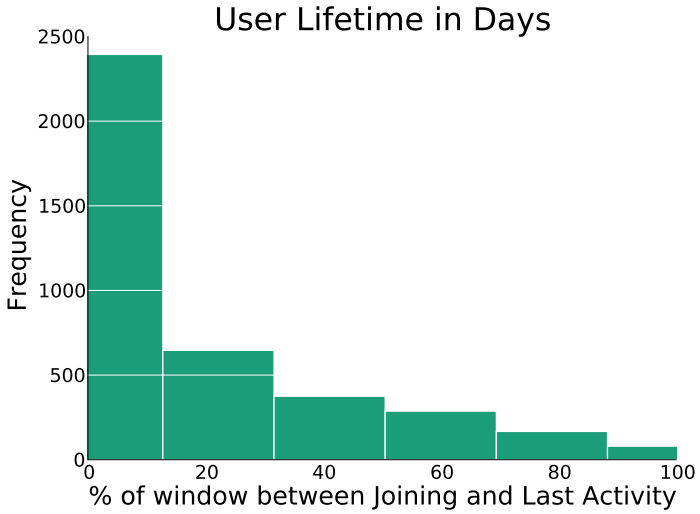
both concept submissions and feedback. Once the challenge ends, user participation drops dramatically, with reduced participation in subsequent challenges. Challenges that begin on or before a previous challenge ends correlate with higher user retention in the subsequent challenge; cases where there is a significant gap between challenges (*e.g.*, mid-May to July, 2011) did not see much return participation.

Figure 3.15 normalizes Fig. 3.14, by shifting everyone by their join date, making the x-axis equivalent to the number of days a user has been on the site. This figure presents a clearer picture of the exact user progression, showing the concentration of activity around the initial challenge followed by a sharp drop-off in participation for most users. Since the y-axis is ordered in time, it also shows that users have behaved similarly since around June, 2011 to present—there is little difference in activity level or distribution of activities after around user 750 and later.

Figure 3.15 shows an identical plot to that of Fig. 3.14, except that all the users have been sorted on the y-axis by the total number of actions they have performed on the site (essentially ranking them by activity level). This provides a sense of relative size and activity level: there are many more single-challenge participants compared to multi-challenge participants, and only a small fraction of the participants heavily contribute. This long-tailed activity distribution is common across a wide variety of social systems [90].



(a) A large number of users are transient—their activity on the site does not extend past a few days. 1794 users out of 5753 users ($\approx 31\%$) join, but do not interact with the site further.



(b) After normalizing the lifetimes by the amount of time since the user joined, the user lifetime steadily decrease, as expected.

Figure 3.13: User lifetimes show a highly transient user population with a common core of long-term participants.

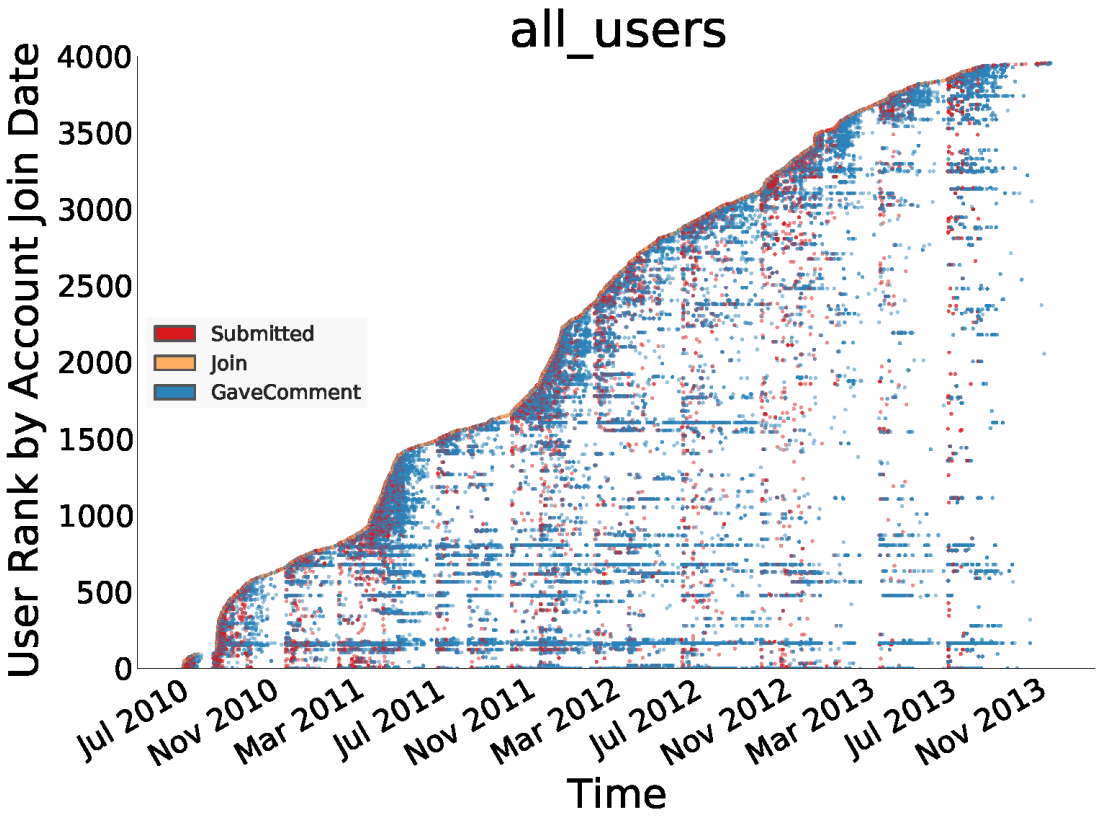


Figure 3.14: This figure captures user actions over time, including when they joined, when they submitted concepts, and when they commented on the concepts of others. Two things are evident: 1) Unsurprisingly, most activity takes place during challenges: user joins, submissions, and commenting activity all increase during challenges; and 2) user retention and participation across multiple challenges was higher when consecutive or simultaneous challenges were available

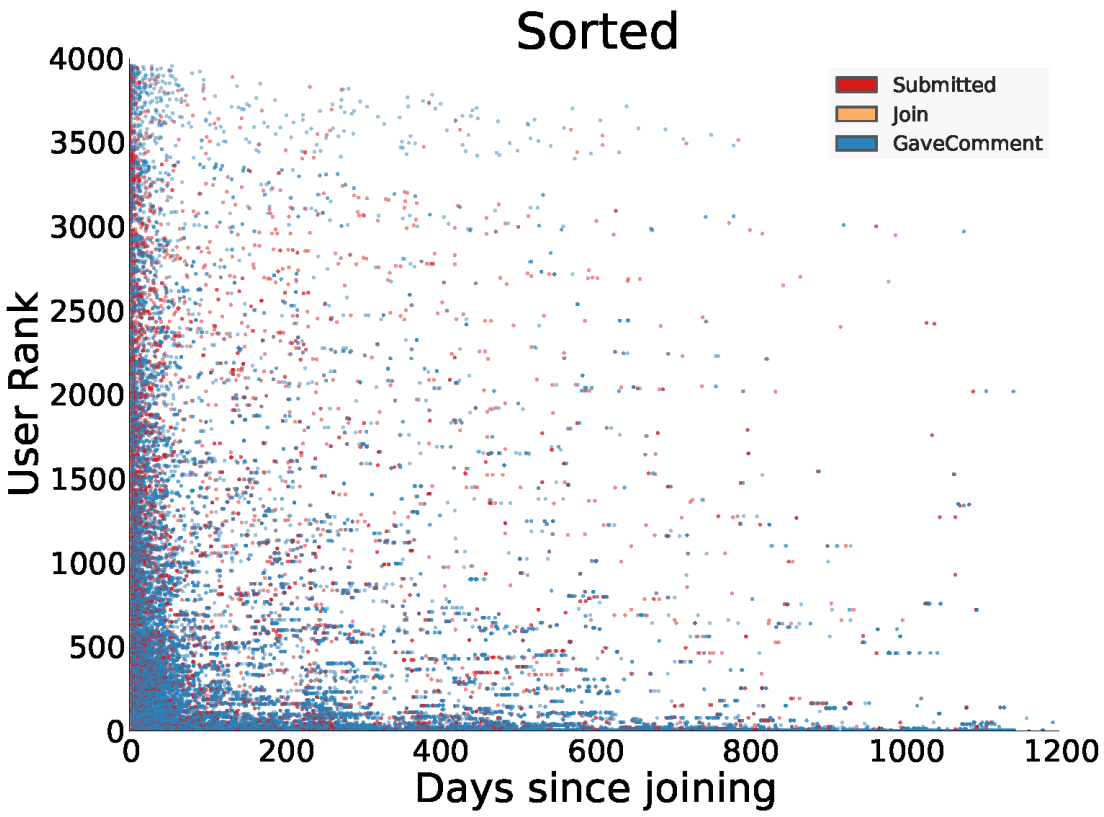


Figure 3.15: This figure captures user actions over days since joining the community, when they submitted concepts, and when they commented on the concepts of others. Users are sorted by their total number of actions on the site, where the most frequent users are towards the bottom. Several things are evident: 1) Only a small portion of users participate regularly across many challenges—most users only temporarily participate. 2) Users initially start out with a flurry of activity that tempers in later challenges. 3) Among more frequent members, giving comments is more popular than submitting new concepts.

The differences in user activities become more pronounced when grouping users by those who have partcipated in only one challenge (2897 users = 73% of active users) and those who have participated in multiple challenges (1062 users = 27% of active users). Figure 3.16 records what happens next to the users after they perform a particular activity—essentially it is a "transition matrix" between activities. Comparing Fig. 3.16a and 3.16b, those participants who participated in multiple challenges spent more time giving comments to others than they did submitting concepts or getting comments from others. In both cases, commenting formed a reinforcing cycle of giving and getting comments.

3.4 Implications for Design Communities

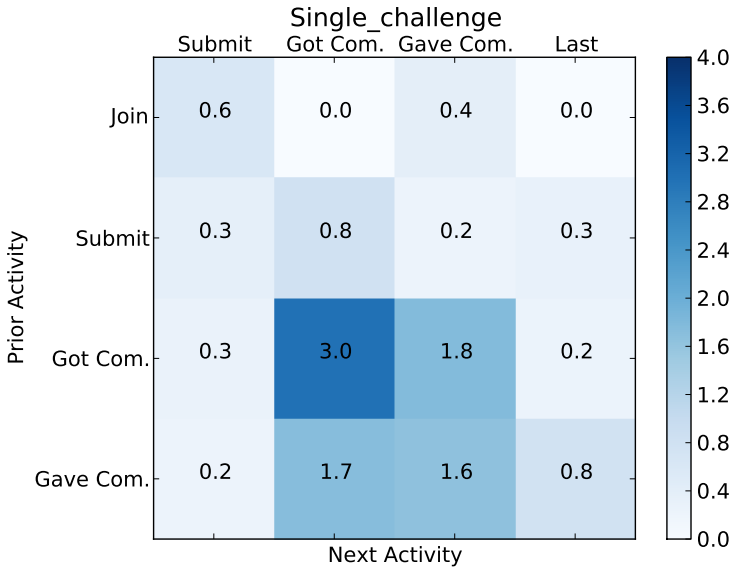
The results from the static analysis in section 3.2 and the dynamic analysis in section 3.3 lead to implications for both managing online design communities and for modeling design communities computationally,

3.4.1 Implications for Managing Online Design Communities

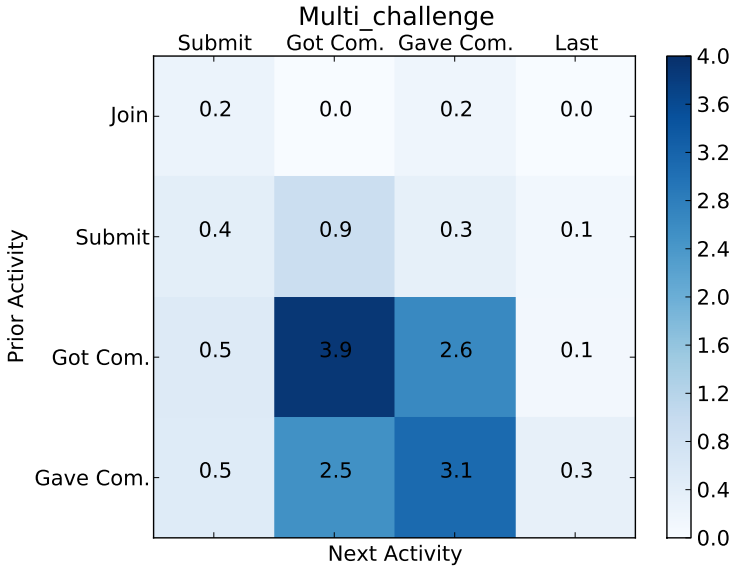
Low efficiency and high diameter reduce information flow in the concept graphs. Since concept nodes represent ideas and links represent information flow in the form of building off of ideas, the way concept graphs are evolving into distributed, low efficiency networks leads to a couple of possible conjectures: (1) the vast majority of concepts lack useful information, and thus are not worth building off of; (2) it is difficult to find and connect disparate concepts, leading to only minor local clustering and limited global structure; or (3) the time frame or format of concept submission is such that it does not provide sufficient time to review, connect, and cycle through iterations of concepts on the network.

Addressing (1) is outside of the data's scope, but (2) and (3) could be addressed by employing many of the techniques used in the above network analysis: locating ideas from distant parts of the graph to present to participants as possible idea "mash-ups" or using community detection techniques to identify or create common idea groupings. This would increase efficiency within the concept graph and has the potential to combine distinct features from different parts or "idea communities."

Encourage core users to collaborate with periphery users to increase network robustness, centralization, and efficiency. While the core-periphery social structure was different than expected, it carries with it several advantages and trade-offs that help make the design network more robust and stable:



(a) Users who only participated in a single challenge submitted around 1-1.5 concepts on average, and gave about 4 comments. They received around twice as many comments as they gave to others.



(b) Users who participated in multiple challenges submitted between 5-6 concepts on average over their lifetime, but gave a substantially higher percentage of comments to others.

Figure 3.16: A comparison of the transition states between single- and multi-challenge users. The numbers in the boxes represent the average number of times a user went from one activity to the next activity. Those users who participate in multiple challenges put more emphasis on giving comments.

1. Core-periphery networks are more robust to random or targeted node loss than other power-law distributed network types of similar efficiency [27, 106, 90]. This is good since open innovation networks are reliant on voluntary participation by individual nodes, any of which could stop participating at any moment.

2. The core-periphery structure is conducive to high centralization and network efficiency, which helps transfer information among collaborators.

3. The disassortative mixing creates an inclusive environment for periphery users to get involved and move towards the core.

As a proactive strategy for strengthening design networks, one could incentivize disassortative behavior by asking high-degree core members to comment or collaborate with periphery members more regularly (a practice currently employed by the OpenIDEO community managers).

However, a highly clustered central core may harm ideation potential. The primary concern with highly clustered core networks is that, when used to communicate ideas or concepts, it may impede idea generation. Highly clustered, inefficient networks facilitate forms of complex contagion, or multiple repeated exposure, that can cause people to prematurely cease exploring ideas [84]. Essentially, if all your neighbors are exploring similar ideas, you are more likely to produce something similar to that idea—fixating on it in place of exploring other options. In a highly clustered network this effect feeds on itself since many people have common neighbors, creating false confidence about the strength of an idea and premature fixation on a portion of the design space. Two means to counteracting high clustering are to: 1) expand the diversity of the contributors, as this improves the overall variety ideas being discussed; and 2) encourage the idea generation practice of first doing individual idea generation before viewing the ideas of others. (This limits initial exposure to potentially fixating ideas, after which members can take better advantage of the core-periphery network regardless of its efficiency or clustering.)

Promote continuous challenge involvement and commenting to increase use engagement. The transient nature of the user population is both a blessing and curse. On the one hand, having new members constantly joining increases idea diversity (and possibly novelty), but on the other hand having users leave after one challenge does not foster a strong sense of community feedback or knowledge retention across challenges. One can employ two complementary strategies here: increase user retention and make better use of the transient population. For the first, one can space challenges so that they are consecutive—the continuity of involvement appeared to correlate with participation in the subsequent challenges. While this link

is not necessarily causal, social momentum and continuous engagement in the site should positively affect retention. For the second, OpenIDEO's two strategies of using community managers to "cross-pollinate" between new users' ideas, along with providing incentives for giving comments, facilitates the disassortative, efficient, and lightly clustered network structure that Mason *et al.* [84] and Stephen *et al.* [123] recommend for product ideation. Further incentives for building off of existing ideas, rather than just commenting, would increase this benefit.

For establishing community, Fig. 3.11 and 3.12 both demonstrate the dynamic nature of how design communities evolve. Research has not yet investigated whether having a single larger community or several smaller connected communities provides a more conducive idea generation environment, so it remains to be seen what specific level of community a design network should strive towards. Despite the changes in community structure, the general structure of OpenIDEO remains similar over time: a central core of users collaborates heavily with temporary periphery members. This overarching structure is the more likely cause of the beneficial disassortativity, efficiency, and clustering seen in Fig. 3.10.

3.4.2 Implications for Modeling Design Communities

Consider explicitly modeling disassortativity in collaboration networks. The disassortative nature of the collaboration social graph is non-standard in current social collaboration models, and does not appear in datasets from nearby domains like Open Source Software. Those working on theoretical or simulation models of design team collaboration should consider including disassortativity characteristics as part of their modeling strategy.

More research is needed to understand and model of core-periphery structures. Research in core-periphery structures is still an active area of research [27, 106]—there is much to be gained by collaborating with other researchers working in network analysis. As an example, section 3.2.4 presented an initial exploration of the role of the community managers—much more work could be done to explore the possibilities for network interventions in design collaborations. A natural extension of this work would be exploring the structural effect of pairing new periphery members with existing core members or recommending concepts from different parts of the concept graph.

3.5 Summary

This chapter presented an empirical network analysis of OpenIDEO, a real-world online design community, and tracked its evolution over time. OpenIDEO's social graph

is disassortative (*i.e.*, Table 3.2) and lacks the multiple community structure found in typical social networks (*e.g.*, citation co-authorship [51] and social communication networks [130] of similar size frequently display sub-communities of 4 and higher). Moreover, it took several design challenges to stabilize, with its community structure becoming a mildly clustered core-periphery network. This could be caused by multiple factors, including: size, presence of community member leads, and collaboration incentives, though further study would be necessary to determine causal relationships. Community members that participated in multiple challenges were more likely to give design feedback to others, with consecutive challenges engendering higher retention.

While the efficiency and robustness benefits of the social graphs' structure are advantageous, there is the possibility for design fixation through complex contagion if the core network becomes too clustered. Section 3.4 discussed possible counter-strategies including increasing community involvement with periphery nodes, increasing participant diversity, spacing challenges sequentially or with overlap to promote continuous involvement, and using incentive structures to encourage giving feedback within communities. It also addressed how these structures might impact theoretical models of design networks, specifically the need to model disassortative collaboration behavior and core-periphery structures. For researchers, it provides a benchmark with which to compare the growth of other design communities.

Several new questions arise for future quantitative or qualitative investigation: At what point do transitions occur between single- and multi-community network structures? How does one balance network efficiency with the desire to help members exploit the good ideas of others? What are the range of factors that convince users to regularly participate? How could interventions, such as targeted collaboration reminders, alter the network's evolution over time to promote better ideation? What causes an individual to continue participating in the network when a challenge ends? What are appropriate computational methods for modeling this social interaction (*e.g.*, Markov Reward Networks, as in Fig. 3.16)? Answering many of these questions requires a more controlled environment than the present observational dataset allows, and would be a fruitful area of future research. Ultimately, understanding how these design communities grow, evolve, and (eventually) die out allows managers of online design communities to foster environments that better support distributed idea generation.

Thus far, the dissertation has looked at overall design communities and how they form. The next chapter introspects within these communities, looking at specific design processes used by HCD Connect, another online design community, and how user behaviors can be used to extract design processes from data.

Chapter 4

Extracting Design Processes

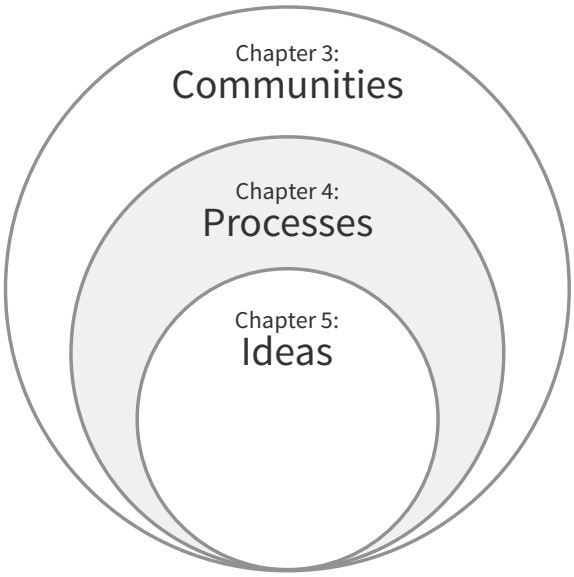


Figure 4.1: This chapter addresses the role of data at the design process level: how individuals and teams learn and use design methods within a design process.

Once one has set up a design community, the next task is to structure their interactions through some kind of *design process*, so that they can produce new designs. This process typically involves the application of a variety of *design methods*, each of which is an activity that furthers the completion of a design. These methods can be applied across different stages of a design process. For example, when starting out

Portions of this chapter appear previously in [43, 44]

designers would leverage user research methods, such as interviews or observations to define what a product or service should accomplish. Once this is done, a designer might shift to methods that help them generate and then evaluate ideas, such as TRIZ [4] or prototyping. The methods a designer uses, and the order in which she uses them, would be her particular design process. This process might vary across teams or individuals, as well as across the specific type of problem being solved. Selecting an appropriate design process is one hallmark of an experienced designer. The largest current design method database lists over 300 different methods [108]—a conservative estimate that easily exceeds any designer's ability to learn.

This chapter studies how online design communities learn about and act upon design processes, specifically: How do the actions of community members shed light on design methods and how they can be used? How do you identify the design process used by community members, without requiring them to explicitly encode it? How do you use the data created by community members to help them improve their design process in the future? To answer these questions, the chapter shifts focus to a different design community, HCD Connect, which focuses on design process communication and education.

HCD Connect stores design case studies using IDEO's Human-Centered Design (HCD) Toolkit, a collection of 39 user research methods intended to be used at the beginning portion of a design process. It uses these case studies as a community-created education resource; the idea being that new members can read through the case studies, learn about different design methods, and start to understand which methods work well together in practice. The main problem facing this community is that the number and complexity of the stored design cases exceeds the amount that an individual member can reasonably read, limiting its usefulness at sharing knowledge; for example, as of this writing, HCD Connect lists over 1500 case studies, each of which contains pages of text and multi-media content.

To address this problem, the chapter casts HCD Connect in the context of Recommender Systems, a sub-field of Information Retrieval. By doing so, it demonstrates how information about design methods can be automatically reused to aid in search and knowledge discovery, while limiting the number of cases a designer needs to manually search through. For example, it will show how using cases to calculate the covariance between different design methods leads to three useful outcomes:

1. False Discovery Rate Control algorithms [9] from large scale statistical hypothesis testing can illuminate which particular methods are appropriate for different kinds of design problems, such as Agriculture or Healthcare.

2. Using Spectral Clustering [91] on the covariance between design methods (method covariance) allows you capture which methods group together. These automatic groupings agree with human expert groupings to 92% accuracy.

3. Collaborative Filtering techniques [68] can combine method covariance with attributes of a given problem to help designers select better methods than they would be able to using method popularity alone.

In each of these areas, the chapter demonstrates how the wealth of data that previously impeded the analysis of design methods instead now acts as an asset, providing knowledge regarding how methods are used and improving the quality of design processes over time. It ends with discussing implications for studying design methods, recommending design methods, and how the results of this chapter generalize beyond HCD Connect.

4.1 Background on Design Methods and Recommender Systems

This chapter builds off of two primary research areas: categorizations of user research methods and recommender systems.

4.1.1 Categorizations of User Research and Design Methods

Researchers have been developing and discussing appropriate user research methods for decades, with yearly conferences devoted to the topic (*e.g.*, EPIC[1]). Many authors have written books cataloging or otherwise classifying design and user research methods [73, 14, 15, 67, 83]. Recently, seperate websites[58, 107] Coming from the field of architecture, Geoffrey Broadbent's work [14, 15] seeks to understand design methods through the lens of how the designed artifact interacts with various stakeholders, such as the humans who use the design or the environment the design will be situated in. Others view design as a temporal process, and organize design methods according to which stage of a design process a method is most appropriate. For example, Christopher Jones [67] divides the design process into three sequential stages (Divergence, Transformation, and Convergence), and allocates methods according to each stage. IDEO's HCD Toolkit is most similar to Jones' organization, in that its Hear, Connect, and Deliver stages follow each other in time.

Design and user research methods vary along many factors, and their widespread proliferation and expansion has been recently addressed by websites that collect and categorize methods along multiple dimensions. For example, the Helen Hamlyn Centre for Design at the Royal College of Art operates "Designing with People" [58], a collection of user research and design methods that categorizes research methods by their inputs and outputs, the stage of the design process, the relationship of the

[1] http://epiconference.com/

method to the people who will use the design, and the type of interaction afforded by the method. Roschuni *et al.* [108] use ontologies to not only categorize method dimensions, but to understand how those dimensions interact with one another.

The work in this dissertation contrasts with prior research in several ways. First, almost no prior research on categorizing design methods validates their categorizations in any formal or quantifiable way, such as using inter-rater reliability across multiple raters. In contrast, this chapter demonstrates how tools from Spectral Clustering can use eigen-decompositions of method similarities to quantify the differences between groups of methods, matching expert-given groupings with high accuracy. Second, this chapter provides quantifiable means for differentiating how methods differ from each other along different problem-specific dimensions, such as the type of design problem (*e.g.*, Healthcare vs Agriculture problems). It does this by leveraging large-scale statistical hypothesis testing techniques, such as False Discovery Rate Control algorithms. This computational viewpoint proposed in this paper can inform current research in categorizing user research methods, by providing a set of quantitative techniques that complements existing qualitative approaches.

4.1.2 Recommender systems

Recommender systems refer to a class of algorithms that recommend content to a user. Some popular applications include Netflix, which uses a person's movie watching habits to recommend new movies, or Google, which uses keywords as well as past browsing behavior to recommend web-pages. There is a vast amount of research on this topic, including yearly conferences such as ACM's RecSys[2], and two recent review papers by Resnick and Varian [105] and Adomavicius and Tuzhilin [1] provide a more complete overview. For the purposes of this chapter, related efforts can be broken down into three camps depending on the type of data they use to produce their recommendations: Content-based Filtering, Collaborative Filtering, and Hybrid Filtering.

Content-based Filtering bases its recommendation solely on the content of the item itself. For example, if a user says they like comedic movies, Netflix might recommend movies tagged with "comedy" more frequently than those tagged with "drama." This was one of the earliest approaches to recommending content, with its roots in the Information Retrieval community [82]. Popular examples include Google's PageRank algorithm [95] as well as text-modeling approaches such as Latent Semantic Analysis [29] and Latent Dirichlet Allocation [12], which build content features by summarizing text content. In the context of design methods, these content features might include the method's textual description or the time required to execute the

[2] http://recsys.acm.org/

method.

A related field of research which is gaining popularity is the field of "Learning to Rank," which formulates item ranking as a statistical learning problem and uses classification and regression techniques from machine learning to solve ordinal ranking problems. For example, RankNet [17] and ListNet [21] both utilize Artificial Neural Network architectures to determine ranking functions over content features. For a comprehensive overview of Learning to Rank methods, Liu [79] provides an excellent review. Over the past decade, solely content-based approaches have fallen out of favor for either Collaborative Filtering models or hybrid models that combine both approaches.

In contrast to Content-based Filtering, **Collaborative Filtering** bases its recommendation solely on the covariance between users and items. For example, if user A likes the movies "Titantic" and "Caddyshack", and user B likes "Titantic," then the algorithm might conclude that user A and user B are similar, and thus user B might also like "Caddyshack," regardless of the content of the movie itself. For design methods, this might be which design cases use which methods—if case study A uses methods 5 and 17, then the algorithm learns something about the relationship between 5 and 17 that it can leverage for future predictions, despite not knowing anything in particular about method 5 or 17. The earliest collaborative filtering methods were Neighborhood methods, such as that of Herlocker *et al.* [59], which used weighted averages of scores from similar users to estimate a new item score.

Neighborhood techniques have been largely replaced by matrix factorization approaches, which are generative unsupervised models that uncover a latent set of user and item features, representing the score as a cross-product between the two. Their wide-spread usage and popularity is due in part to their independence from content features and in part to the "Netflix Prize" competition, which spurred research from academia and industry alike. Notable examples that emerged from that area include the Bell-Kor system [8], which won the Netflix Prize, as well as techniques such as Bayesian Probabilistic Matrix Factorization [109], variants of which are currently under active research.

Hybrid Filtering mixes the above two models by using both content and collaborative features to inform the recommendation, often at the cost of additional computation and complexity. For example, if user A likes "Titantic", "Caddyshack", and "The Shawshank Redemption"; user B likes "Titantic"; and "Titantic" is considered a drama, then a Hybrid Filtering algorithm might conclude that user A and user B are similar and enjoy dramas, and thus user B might prefer "The Shawshank Redemption" over "Caddyshack" since it is both similar to what user A selected, but also within the "drama" category. This hybrid approach ameliorates some of the disadvantages of the above two models: for new items which do not have collaborative features (referred to as the "cold-start" problem), hybrid models can use content in-

formation to improve recommendations; likewise, hybrid models can use collaborative information when item content is not available or informative. Most modern, successful recommender systems use some form of Hybrid Filtering [105, 1, 8]. For example, Badaro *et al.* [6] utilize weighted combinations of Content- and Collaborative-Filtering approaches, while Ghazanfar and Prugel-Bennett [49] use neighborhood-based content and collaborative features that are combined using Boosting [40]. Hybrid approaches are not without their own problems, however; Yujie and Licai [135] highlight the fact that the increased number of parameters and data sparsity among those parameters can make it difficult to accurately train hybrid methods without sufficient data.

4.2 Overview of HCD Connect

This chapter analyzes the design methods and processes in the Human-Centered Design (HCD) Toolkit developed by IDEO, an award-winning global design firm. In particular, it looks at HCD Connect, an online platform run by IDEO.org, IDEO's non-profit branch that deals with design for development projects. HCD Connect distributes a user research method toolkit and provides a forum where designers can post case studies of different developing world problems. These cases describe the user research methods a designer used to address a particular design problem [64, 61]. Specifically, the user research methods the designers discuss come from the 39 methods included in the HCD Toolkit. HCD Connect categorizes these user research methods into three different design stages:

"**Hear:** Determine who to talk to, how to gather stories, and how to document your observations.

Create: Generate opportunities and solutions that are applicable to the whole community.

Deliver: Take your top solutions, make them better, and move them toward implementation." [3]

A summary of each of the 39 methods can be found on IDEO's online version of the HCD Toolkit.

The dataset used in this chapter consists of 809 case studies posted to HCD Connect between June 2nd, 2011 to September 13th, 2013. Figure 4.2 shows an example of what a case study contains: (a) text and pictures describing the problem, (b) information regarding the user who submitted the case, (c) a list of development "focus areas" which categorize what type of problem the case was solving, and (d) a list of the HCD Toolkit methods that the case used to address the problem.

[3] http://www.hcdconnect.org/toolkit/en

Figure 4.2: An example of an HCD case. Some common elements include: (a) A title and description discussing the problem and methods used, (b) information about the user submitting the case study, (c) a list of focus areas applicable to the case, and (d) a list of HCD Toolkit methods that the case used.

# Cases	*% Cases*	*Focus Area*
506	62.5	Community Development
480	59.3	Agriculture
317	39.2	Education
281	34.7	Environment
225	27.8	Health
140	17.3	Water
124	15.3	Gender Equity
97	12.0	Energy
92	11.4	Financial Services

Table 4.1: Breakdown of the 809 cases by Focus Area. A case could have multiple focus areas.

For the information regarding the user who submitted the case(Fig. 4.2b), the organizational affiliation of the person who submitted the case is classified as a member of "IDEO" if their organizational affiliation contained the string "IDEO" and classified as "non-IDEO" otherwise. IDEO members are typically industrial designers within IDEO, organizers within IDEO.org (IDEO's non-profit arm that operates HCD Connect), or IDEO.org fellows (who are designers that specifically work with IDEO.org). Non-IDEO members come from almost every continent and have occupations that range from directors and managers at non-profit organizations to freelance designers to design graduate students to Entrepreneurs/CEOs to development consultants. The common factor across most members is that their work focuses on development or social programs.

For the list of development "focus areas" (Fig. 4.2c), Table 4.1 lists all the nine possible focus areas, along with how frequently each area occurs in the cases. Focus areas are not mutually exclusive; a case study can include multiple focus areas.

The list of HCD Toolkit methods that the case used (Fig. 4.2d), is encoded in a 809×39 binary matrix, where each row is a case, each column is method, and a cell is one if that method was used in that case study and zero otherwise. As is evident in Fig. 4.2, a case study in HCD Connect can utilize multiple methods. As the next section demonstrates, this is because different methods complement each other; for example, a project evaluating mobile phone applications for healthcare might use interviewing methods to gather user feedback on a prototype, while a storyboarding method could evaluate a user's workflow. In such situations, methods would be positively correlated with one another. In other situations, one might expect methods to substitute for one another; for example, if someone has already conducted an individual interview then they might be less likely to perform other types of interviews. In that case, methods would be negatively correlated with one another. For

Example Problem Statements:	**HCD Connect Methods Used in Case Study:**
As we worked side by side with small-holder farmers in Peru to harvest coffee, we learned there were many things we could improve to make our device easier to use. **Focus Area:** Agriculture.	Individual Interview, In-Context Immersion, Community-Driven Discovery, Capabilities Quick Sheet, Participatory Co-Design.
Butcher block furniture is popular in the United States. However, in India there is a whole market for recycling waste wood. This recycling can be better if the wooden pieces are adhered together and then made into furniture. As in butcher block furniture, here also pieces of wood are put together to for a plank for furniture. **Focus Areas:** Environment; Community Development.	Storytelling With Purpose, Try Out A Model, Individual Interview, Inspiration In New Places, Innovation 2x2.
In collaboration with the American Refugee Committee (ARC) and IDEO.org design team, IDEO.org co-lead, Jocelyn Wyatt, shares her experiences facilitating co-design sessions with women in the Democratic Republic of Congo in order to gain insights on bringing health, water, and nutrition solutions to the community. **Focus Areas:** Water; Community Development; Health.	Storyboards, Role-Play, Track Indicators, Evaluate Outcomes.

Table 4.2: Examples of the 886 design method case studies from HCD Connect. They contain problem descriptions, as well as the human-selected methods used to solve that problem and the tagged "Focus Area" of the problem.

the 39 methods in IDEO's HCD Toolkit, there was almost no incidence of methods being negatively correlated with one another, meaning that the methods in the HCD Connect Toolkit did not frequently substitute for one another. Table 4.2 lists some respresentative examples of the kind of case studies contained in the dataset.

4.3 Using Data to Reveal Design Processes

This section demonstrates how to use the above case studies to answer several questions about design methods that would be difficult for a novice designer to answer by just reading a few case studies:

1. How does method usage vary across the entire case study corpus?

2. Which methods complement one another?

3. Which methods are more or less useful for different kinds of problems?

4. How does method usage compare between professional designers at IDEO and the rest of the HCD Connect community?

This chapter users one major assumption throughout all sections: that the self-reported methods used were both accurate and appropriate. (An assumption that Section 6.2 will address with respect to all chapters of the dissertation.)

Using statistical techniques such as the Bootstrap [33] and False Discovery Rate Control algorithms [9], the analysis finds that: methods from earlier in the design process are more frequently used; that certain methods correlate well with others, primarily within design stages, and to a lesser extent across design stages; that a select few methods are significantly more common for certain types of development problems than they are in general; and that IDEO designers use fewer methods overall than non-IDEO counterparts and tend to focus on earlier design stages. In general, the algorithms find method pairs that are expected, even though those algorithms have no knowledge of the content of the design method itself.

For the first question, "How does method usage vary across the entire case study corpus?", Figure 4.3 demonstrates the percent of cases that contain a particular method. From this, one can immediately discern the popularity of methods in the initial phase of the HCD toolkit (Hear): members use many of these methods in up to one quarter to one third of all cases. As one moves later in the design process, method usage decreases.

4.3.1 Finding Complementary Methods

For the second question, "Which methods complement one another?", Figure 4.4 visualizes the Pearson product-moment correlation coefficients between each pair of methods across all cases; this correlation ranges between 1 (always used together) and -1 (never used together). Notably, there are no cases of strong negative correlation; methods were either positively correlated or uncorrelated. The figure groups the rows and columns such that each design stage remains together, with the green, orange, and purple labels corresponding to the Hear, Create, and Deliver stages, respectively.

To dig into these correlations further, this subsection will consider two sets of data. First, it looks at correlations across all 809 case studies, regardless of which methods they use. This provides an overall picture of the full corpus and assumes all case studies are equally valuable. Second, it restricts the corpus to only those case studies that use methods from across all three phases ("Hear", "Create", and

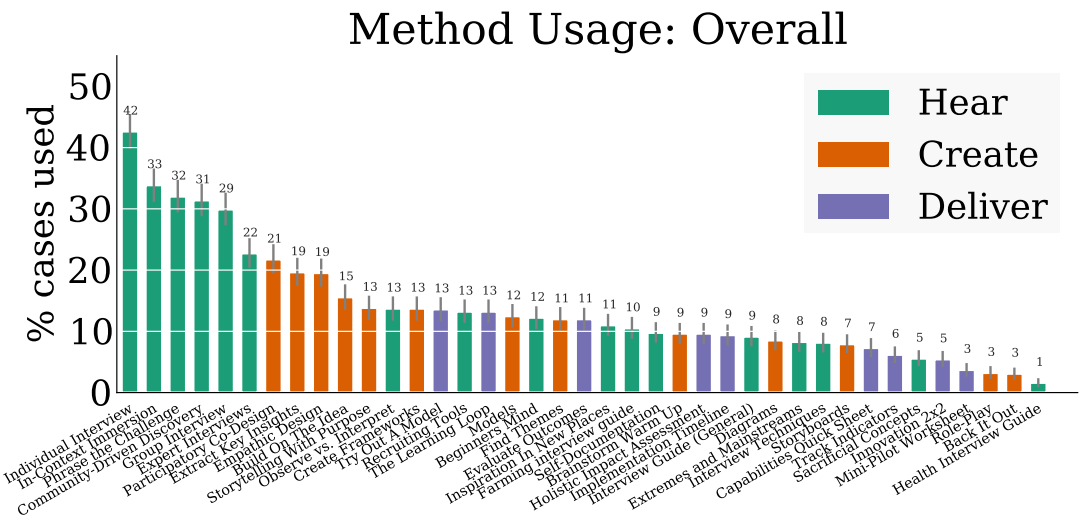


Figure 4.3: Percent method usage by case. Overall, users use methods from earlier design stages more frequently.

"Deliver"). This restricted corpus provides a different interpretation of how methods are related by studying only case studies that covered the entire process.

4.3.1.1 Method comparisons across entire corpus

To highlight which methods are most complementary to one another across the entire corpus, Table 4.3 rank orders the top 20 method pairs by correlation coefficient—*i.e.*, they are the 20 methods most likely to co-occur together. (A full ranked list of all correlations can be downloaded from the dissertation's companion website.) This approach locates many pairs of methods one would expect to be complementary. For example, the methods Individual Interview, Group Interview, Expert Interview, Interview Guide, and Interview Techniques all highly correlate with one another—they all leverage a type of interviewing. Highly visual methods that involve drawing abstractions or clustering also highly correlate with each other: Create Framework, Diagrams, Storyboards, Find Themes, and Extract Key Insights. Methods concerned with assessing the end result of the process correlated together: Evaluate Outcomes, Track Indicators, Implementation Timeline, and The Learning Loop. Community-centered methods, such as Build on the Idea and Participatory Co-Design, correlate with one another. The vast majority of the top-ranked correlations have methods from the same design stage; this is expected, since methods from the same stage would have a higher likelihood of complementing one another, as well as being more

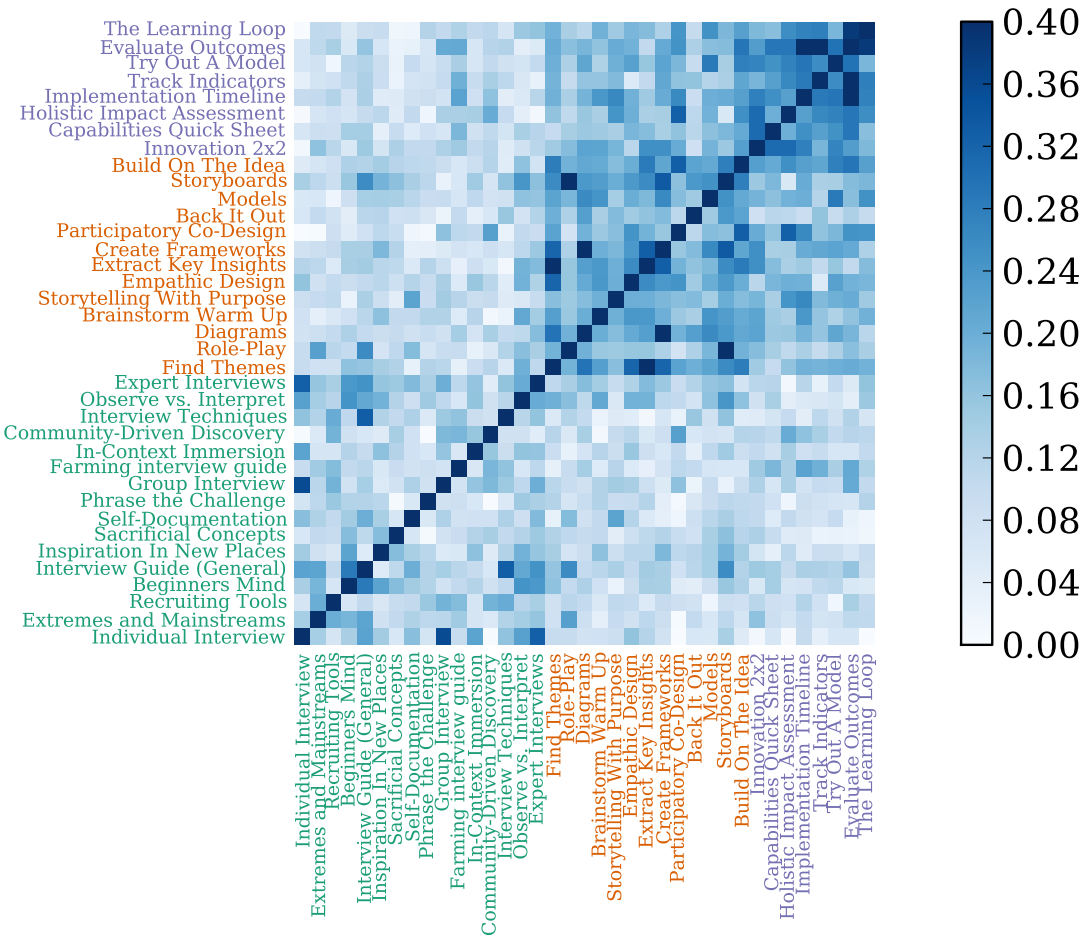


Figure 4.4: Certain methods more positively correlate with other methods, however there is almost no negative correlation between methods. The shaded boxes indicate the correlation coefficient between methods—darker indicates increasing positive correlation. The diagonal is thresholded to 0.4 for clarity of presentation, since it always has correlation of one. Methods from later stages (Create and Deliver) have higher correlation within each category, as well as across categories. "Hear," "Create," and "Deliver" methods are labeled using green, orange, and purple, respectively.

Corr.	*Method 1*	*Method 2*
0.46	(D) Evaluate Outcomes	(D) Track Indicators
0.42	(C) Find Themes	(C) Extract Key Insights
0.41	(C) Storyboards	(C) Role-Play
0.41	(C) Create Frameworks	(C) Diagrams
0.40	(D) Evaluate Outcomes	(D) Implementation Timeline
0.38	(D) The Learning Loop	(D) Evaluate Outcomes
0.36	(H) Individual Interview	(H) Group Interview
0.34	(C) Create Frameworks	(C) Storyboards
0.33	(H) Interview Techniques	(H) Interview Guide (General)
0.33	(C) Create Frameworks	(C) Extract Key Insights
0.33	(C) Build On The Idea	(C) Participatory Co-Design
0.33	(H) Individual Interview	(H) Expert Interviews
0.33	(C) Participatory Co-Design	(D) Holistic Impact Assessment
0.32	(C) Find Themes	(C) Create Frameworks
0.32	(C) Find Themes	(C) Empathic Design
0.31	(D) Capabilities Quick Sheet	(D) Innovation 2x2
0.31	(D) Innovation 2x2	(D) Holistic Impact Assessment
0.30	(D) Try Out A Model	(D) Evaluate Outcomes
0.30	(C) Find Themes	(C) Diagrams
0.29	(C) Build On The Idea	(D) Evaluate Outcomes

Table 4.3: The 20 highest correlated methods from Fig. 4.4; these methods likely complement each other. The method's design stage within the HCD Connect toolkit is shown in parentheses ('H,' 'C,' or 'D' for "Hear," "Create," and "Deliver," respectively.

similar to each other in goal (thus having multiple activities, like interviewing, constitute several possible methods).

One possible caveat to the above results is that certain cases may only focus on certain stages, and thus the correlations could be biased toward correlations within each stage. For example, if a certain project only covered the beginning of the design process (*e.g.*, the "Hear" stage), then certain complementary methods in later stages may not correlate as frequently as they would in case studies that cover all design stages. The next section addresses this caveat by restricting the corpus so that it only contains cases that used at least one method from each of the three design stages.

4.3.1.2 Method comparisons across cases that use all stages

Restricting the corpus to only those cases that use methods in all three stages ($\approx 27\%$ of the 809 cases), Table 4.4 rank orders the top 20 method pairs by correlation

Corr.	*Method 1*	*Method 2*
0.45	(H) Interview Guide (General)	(C) Role-Play
0.43	(C) Storyboards	(H) Interview Guide (General)
0.43	(C) Storyboards	(C) Role-Play
0.37	(C) Create Frameworks	(C) Diagrams
0.37	(H) Interview Techniques	(H) Interview Guide (General)
0.37	(H) Extremes and Mainstreams	(C) Role-Play
0.33	(C) Models	(H) Expert Interviews
0.32	(D) Evaluate Outcomes	(D) Track Indicators
0.32	(D) Innovation 2x2	(H) Extremes and Mainstreams
0.32	(H) Group Interview	(D) Evaluate Outcomes
0.31	(H) Individual Interview	(H) Expert Interviews
0.31	(H) Interview Guide (General)	(H) Extremes and Mainstreams
0.31	(C) Create Frameworks	(D) Innovation 2x2
0.30	(H) Community-Driven Discovery	(C) Participatory Co-Design
0.30	(C) Extract Key Insights	(D) Innovation 2x2
0.30	(H) Interview Techniques	(C) Storyboards
0.30	(H) Individual Interview	(H) Group Interview
0.29	(D) Capabilities Quick Sheet	(H) Beginners Mind
0.29	(H) Individual Interview	(C) Empathic Design
0.29	(C) Models	(D) Try Out A Model

Table 4.4: The 20 highest correlated methods from Fig. 4.4, when filtered by cases that use methods from across all phases. The method's design stage within the HCD Connect toolkit is shown in parentheses ('H,' 'C,' or 'D' for "Hear," "Create," and "Deliver," respectively.

coefficient, similarly to Table 4.3. The two tables share many similarities, but also important differences. In terms of similarities, they both continue to highlight strong correlations for certain within-stage methods. For example, the previous clusters of Interviewing methods (*e.g.*, Individual Interview, Group Interview, *etc.*) and Visual methods (*e.g.*, Frameworks, Diagrams, *etc.*) remain.

In terms of differences, methods now correlate more by how the method is used than by the stage it is used in. In Table 4.3, many visual methods from the "Create" stage correlated together, whereas in Table 4.4 they also correlate with visual methods from different stages. For example, Frameworks ("Create" stage) and Innovation 2x2s ("Deliver" stage) are highly correlated. Likewise, Community-Driven Discovery ("Hear" stage) and Participatory Co-Design ("Create" stage) both heavily involve community participation; they occur as highly correlated in Table 4.4 but not in Table 4.3.

The comparison between Tables 4.3 and 4.4 highlights an important assumption about the above correlation analysis: segmenting corpora will affect the kind of correlations one can expect to find. In Table 4.3 and Fig. 4.4, the clusters and correlations uncovered temporal variation, despite the algorithms having no knowledge of the design stages. When the corpus is segmented to remove this temporal variation, factors relating to the context of the method (*e.g.*, Visual methods) emerge instead. When applying this kind of technique to new domains, the purpose of the desired correlations and clusters should drive the choice of corpus segmentation. In essence, the kind of problem one wishes to solve (*e.g.*, dividing methods by time, or how they are used, or by user group, *etc.*) needs to affect how one collects and segments the data.

4.3.2 Differences in Method Usage Across Focus Areas

To answer the third research question, "Which methods are more or less useful for different kinds of development engineering problems?", this section partitions the case studies by focus area (Table 4.1). It then computes independent sample t-statistics for each method's usage frequency in a focus area, compared with its usage frequency across all other focus areas. Testing all these combinations results in 351 different statistical comparisons, and Fig. 4.5 plots these t-statistics as a probability plot. As expected, most of the comparisons result in no appreciable difference (the straight line), however, on the right and left sides, a few comparisons stand out as unexpected—these are the methods that are particularly suited for a given focus area.

To quantify exactly which pairs substantially differ from zero, the **Benjamini-Hochberg (BH) procedure** [9] can adjust the observed t-statistics. The BH procedure is a Bonferroni-like post-hoc correction to the results of multiple statistical tests (*i.e.*, it can correct the 351 t-statistics to account for the increased probability of false positives). Its principle advantage is that it allows one to directly control the False Discovery Rate—essentially Type-I error, but across multiple tests. With this, one can filter down the comparisons in Fig. 4.5 to the reduced list in Table 4.5. This table contains only method pairs that pass an adjusted 5% Type-I error probability threshold, assuming independent tests. It orders each method and focus area by the probability of the observed t-statistic, while also providing the percentage difference in frequency (%Δ—essentially the percentage effect size).

The results indicate that several methods had sizable differences in percent usage depending on the focus area: In Agriculture—Farming Interview Guide (+16%) and Try Out A Model (+11%); in Community Development—Participatory Co-Design (+15%) and Community-Driven Discovery (14%); and in Gender Equity—Group Interview (+17%). Many of the selected pairs are expected; for example, the algorithm correctly identifies that the Farming Interview Guide is appropriate for Agriculture problems, even though the algorithm did not have prior knowledge about what agri-

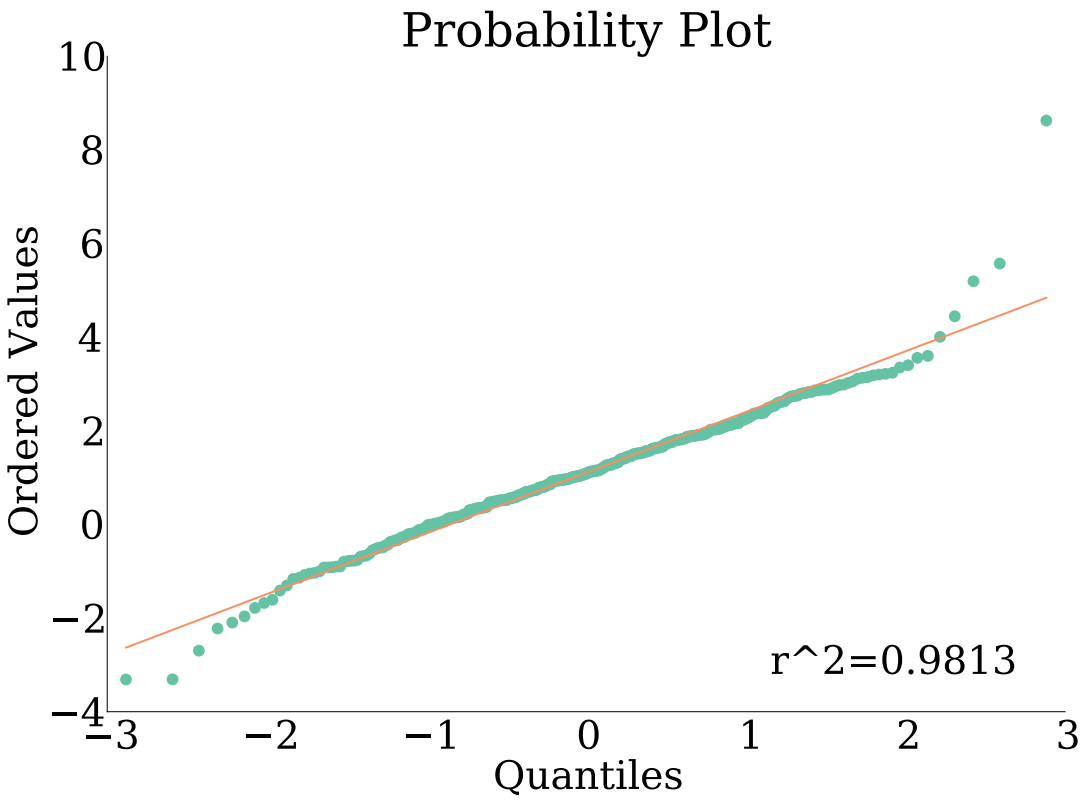


Figure 4.5: A Normal Probability plot for focus area method t-statistics. Most methods in each focus area are not appreciably difference from their usage overall; however, for select methods on the left and right hand side, their usage patterns differ from other focus areas. Table 4.5 lists the methods whose usage differs across particular focus areas.

Prob.	*% Δ*	*Method*	*Focus Area*
5.8e-17	15.7	Farming Interview Guide	Agriculture
3.4e-08	15.3	Participatory Co-Design	Community Development
2.6e-07	11.6	Try Out A Model	Agriculture
1.0e-05	14.3	Community-Driven Discovery	Community Development
6.8e-05	4.7	Mini-Pilot Worksheet	Agriculture
3.6e-04	8.7	Holistic Impact Assessment	Environment
4.9e-04	17.1	Group Interview	Gender Equity
7.2e-04	8.9	Storytelling With Purpose	Education
8.4e-04	5.3	Track Indicators	Agriculture
1.1e-03	-11.1	Expert Interviews	Water
1.1e-03	-14.4	Individual Interview	Water
1.3e-03	8.0	Build On The Idea	Community Development
1.5e-03	7.9	Farming Interview Guide	Environment
1.6e-03	13.2	Storytelling With Purpose	Gender Equity
1.7e-03	15.3	Community-Driven Discovery	Gender Equity
1.8e-03	7.4	Storytelling With Purpose	Community Development
2.1e-03	4.5	Health Interview Guide	Health
2.1e-03	17.3	Community-Driven Discovery	Financial Services
2.3e-03	4.6	Innovation 2X2	Agriculture
2.6e-03	7.8	Evaluate Outcomes	Environment
2.9e-03	5.9	Holistic Impact Assessment	Community Development

Table 4.5: Methods whose usage in a given Focus Area is significantly different from all other Focus Areas. The first column lists the probability of the observed t-statistic, the second lists the difference between the usage percentage of that method in that focus area with respect to other focus areas, the third column lists the method, and the forth lists the particular focus area in which method usage was different. These methods were selected from those in Fig. 4.5 using the Benjamini-Hochberg procedure at a False Discovery Rate (FDR) of 5% assuming independent or positively correlated tests.

culture means. This provides a unique view of methods that uses the corpus of data to illuminate meaningful differences with respect to a problem's focus area.

4.3.3 Differences Between IDEO and non-IDEO users

For the last question, "How does method usage compare between professional designers at IDEO and the rest of the HCD Connect community?", this section compares the method usage behavior between IDEO and non-IDEO affiliated users. This affiliation is a proxy for a particular design culture, since there was no straightforward way to separate out professional designers and non-professional designers from the non-IDEO user pool.

Figure 4.6 demonstrates the differences in how IDEO and non-IDEO members report methods. In the IDEO case, the designers place heavier emphasis on earlier stage (Hear) methods, with method usage dropping off rapidly in later stages. Moreover, those designers did not report many case studies where they used methods from multiple stages (*e.g.*, Hear+Connect). This is in part due to the low percentages of Create or Deliver methods in general, but also could be due to different reporting styles—IDEO designers could systematically split their cases into multiple case studies over different stages, rather than a single case, or they could only be hired for projects in the "Hear" stage of development. Another possible explanation could be that IDEO's culture or the particular structure of their toolkit creates an unstated preference or emphasis on earlier stage methods, or possibly that members selectively report cases they believe would fit that culture.

Comparing individual methods, Fig. 4.7 confirms Fig. 4.6: IDEO users use fewer methods overall, but have a much higher percentage usage in the initial Hear stage, rather than in the Create or Deliver stage. In addition, Fig. 4.7 demonstrates that IDEO designers prefer certain types of methods for each phase, compared to non-IDEO designers who use more of a mix—for example, IDEO designers appear to prefer methods that involve data interpretation, such as extracting insights and themes, building frameworks and models, *etc.* (many of those methods complement each other as per Table 4.3). Since this data involves only self-reported method usage from after a completed design process, there is potential for self-selection: observed differences between groups might be caused not only by differences in behavior, but also by differences in what methods or projects an individual chooses to report. Also, IDEO could be hired to perform more projects that use methods from the "Hear" stage, leading to the differences observed in Fig. 4.7.

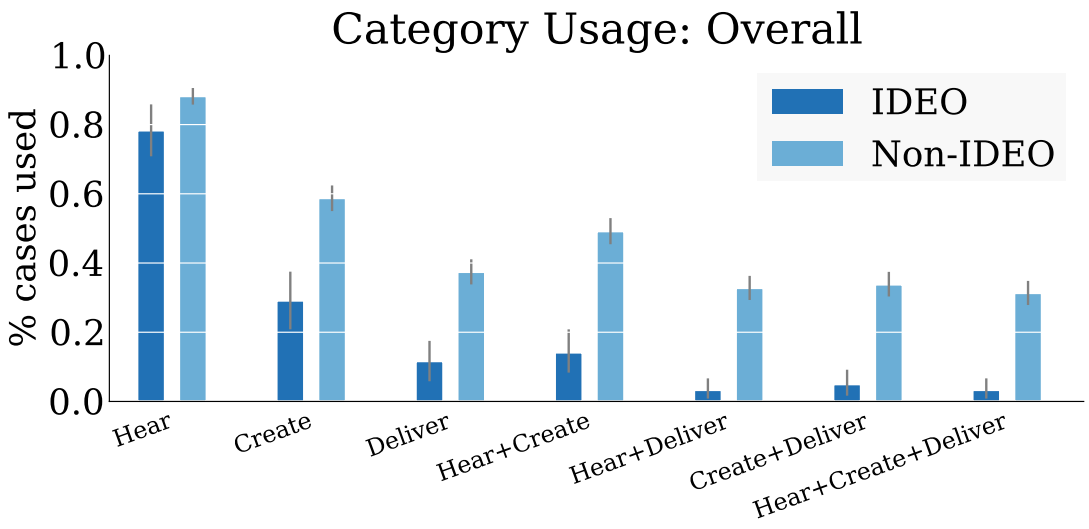


Figure 4.6: Method usage grouped by organizational affiliation. Combined columns, such as "Hear+Create," indicate cases where at least one method from each category was used in the case. IDEO members contribute case studies that typically focus on the first design stage ("Hear"), and rarely submit cases that combine methods across different design stages. In contrast, non-IDEO members contribute cases that use a more even distribution of methods from different design stages, and typically combine methods from different stages in a single case-study.

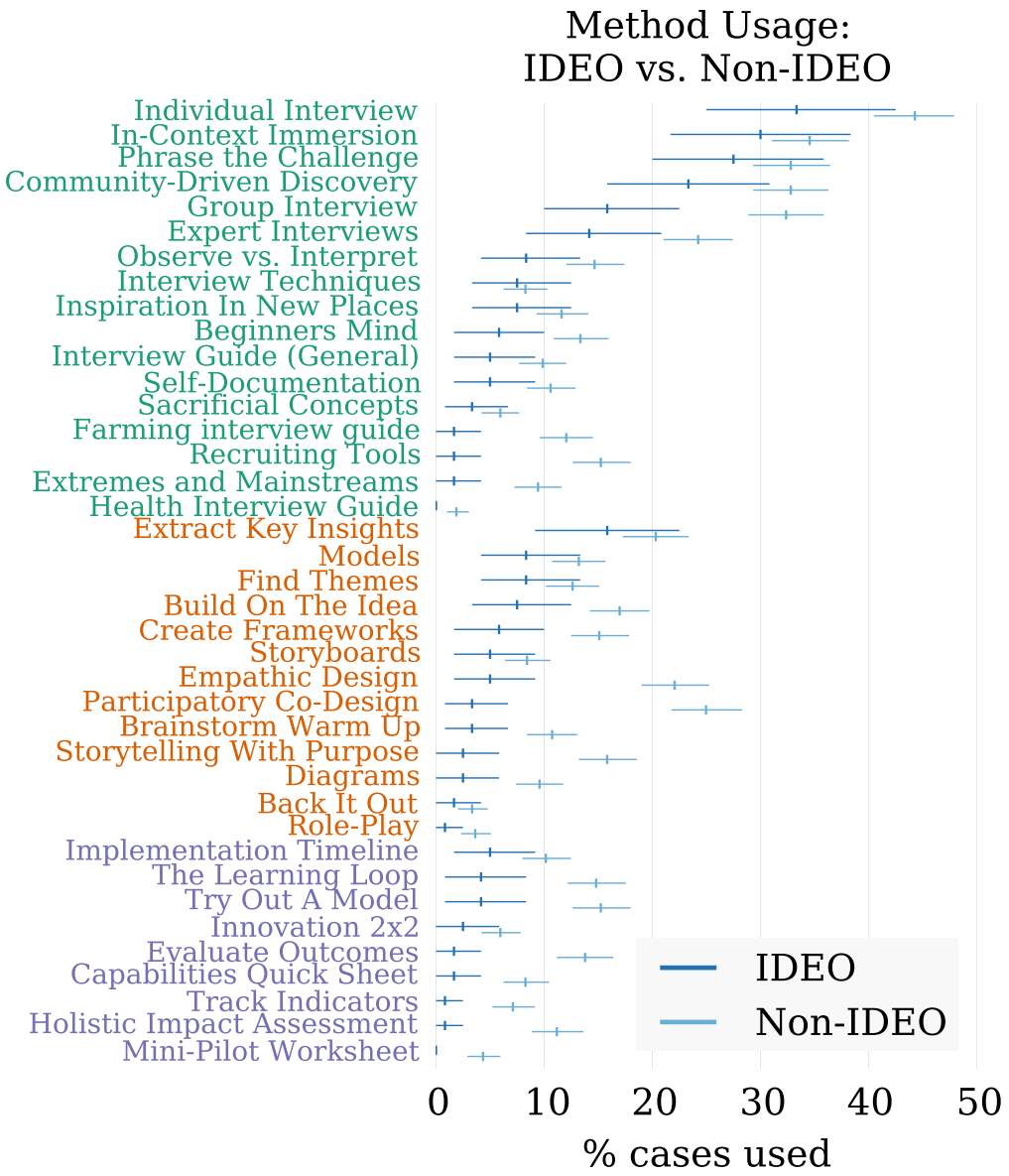


Figure 4.7: Differences in particular method usage between IDEO and non-IDEO members. The methods are grouped by green, orange, and purple for "Hear," "Create," and "Deliver" respectively. As noted in Fig. 4.6, IDEO members tend to use fewer methods per case overall, and particularly focus on the first design stage (Hear).

4.4 Using Data to Group Design Methods

The above section demonstrated how method covariance could lend insight into the relationships between pairs of methods and how they are applied. Taking this one step further, this section shows how that same covariance can be used to uncover global groups among multiple methods, thus providing an automatic way of categorizing design methods. The key technique this section uses to cluster design methods is Spectral Clustering.

Spectral Clustering is an unsupervised learning algorithm which represents design methods as a graph, where each method is a node, and methods are linked to each other with weights equal to their similarity (in this case their covariance or correlation, *e.g.*, Table 4.3). This creates a densely connected graph where all 39×39 methods are connected to each other. Finding groups within the graph is then equivalent to cutting links in the graph until the graph splits into multiple pieces. That problem is called the *discrete graph-partitioning problem*, and is NP-Hard.

Spectral Clustering approximates that problem by relaxing the discrete problem into one based on the eigenvalues of the similarity matrix between methods (*e.g.*, Fig. 4.4), called *spectral graph partitioning*. It works by decomposing the similarity matrix into its eigenvectors and eigenvalues, and then keeping the k-highest eigenvalue/eigenvector pairs, where k is the number of clusters one wants to partition the data into. This projects points that were originally in a 39-dimensional space, down onto a 3-dimensional space. The idea is that points that were similar to one another but spatially far away in a 39-dimensional space will be projected close to one another in the 3-dimensional space. Once in this smaller space, one can then run any standard distance-based clustering algorithm, such as K-means clustering, to group points more effectively.

To use Spectral Clustering on design methods, one needs to determine two things: what should the similarity input matrix contain, and how many clusters should exist? For the former, a reasonable choice of similarity is the covariance matrix between methods (seen in Fig. 4.4 above). However, the empirical covariance matrix produces poor eigenvalue estimates (what Spectral Clustering uses), so one can use the Graphical Lasso [41] instead which transforms the empirical covariance into a matrix with less biased eigenvalue properties.

For the number of clusters (k), IDEO themselves group their methods into three clusters ("Hear," "Create," and "Deliver"), so this section uses the same number ($k = 3$) to compare the automatic groupings to those provided by IDEO. Using cross-validation, Spectral Clustering produced groups that agreed with the IDEO provided clusters to an average of 92% accuracy (36 of 39 methods) on average. Figure 4.8 shows an example of the clusters found using spectral clustering; this is similar to the original labels seen in Fig. 4.4. This result is surprising, since the clustering algorithm

did not have any prior information about the categories, and was able to determine the expert-provided labels using only the method co-occurrence contained in the case studies. This also leads to a straightforward area of future research: could these types of automated clustering algorithms find further clusters within methods, or generate an hierarchical taxonomy of methods?

4.5 Using Data to Recommend Design Methods

While the above sections analyzed existing case studies, this section focuses on recommendation, specifically, how does one use existing case studies to help designers pick better methods for future problems. This section demonstrates the performance of three types of recommender systems on the task of recommending design methods for new problems: Content-based filtering, Collaborative Filtering, and Hybrid Filtering. It finds that Collaborative and Hybrid Filtering, which utilize the method covariance, outperform strictly Content-based Filtering, which only uses attributes of the problem.

4.5.1 Content-Based Filtering

Content-based Filtering for recommending design methods involves summarizing the problem descriptions, and then using that text to predict which methods are most relevant for a given problem—the intuition being that design problems that have similar problem descriptions may use similar methods. This section uses Latent Semantic Indexing (LSI, also referred to as Latent Semantic Analysis—LSA) to quantify that similarity [82]. LSI employs the bag-of-words model for representing a text document, which ignores word order and grammar, and considers only frequency of word occurrence. Given the 886 case study descriptions, Singular Value Decomposition (SVD) projects the word count matrix into a latent space of 50 topics. The resulting 886×50 matrix M contains a row for each case study and 50 columns representing the case's similarity to each of the 50 topics. The algorithm then uses this topic matrix to train a classifier, which outputs the probability that a given method m will be used in each design case c.

This section evaluates four different Content-Based Filtering algorithms:

Random Forests: an ensemble classification technique that fits a number of decision tree classifiers to randomized sub-samples of the dataset; it uses these subsamples to rule out non-useful features and gives us a straightforward method of discarding unimportant text topics.

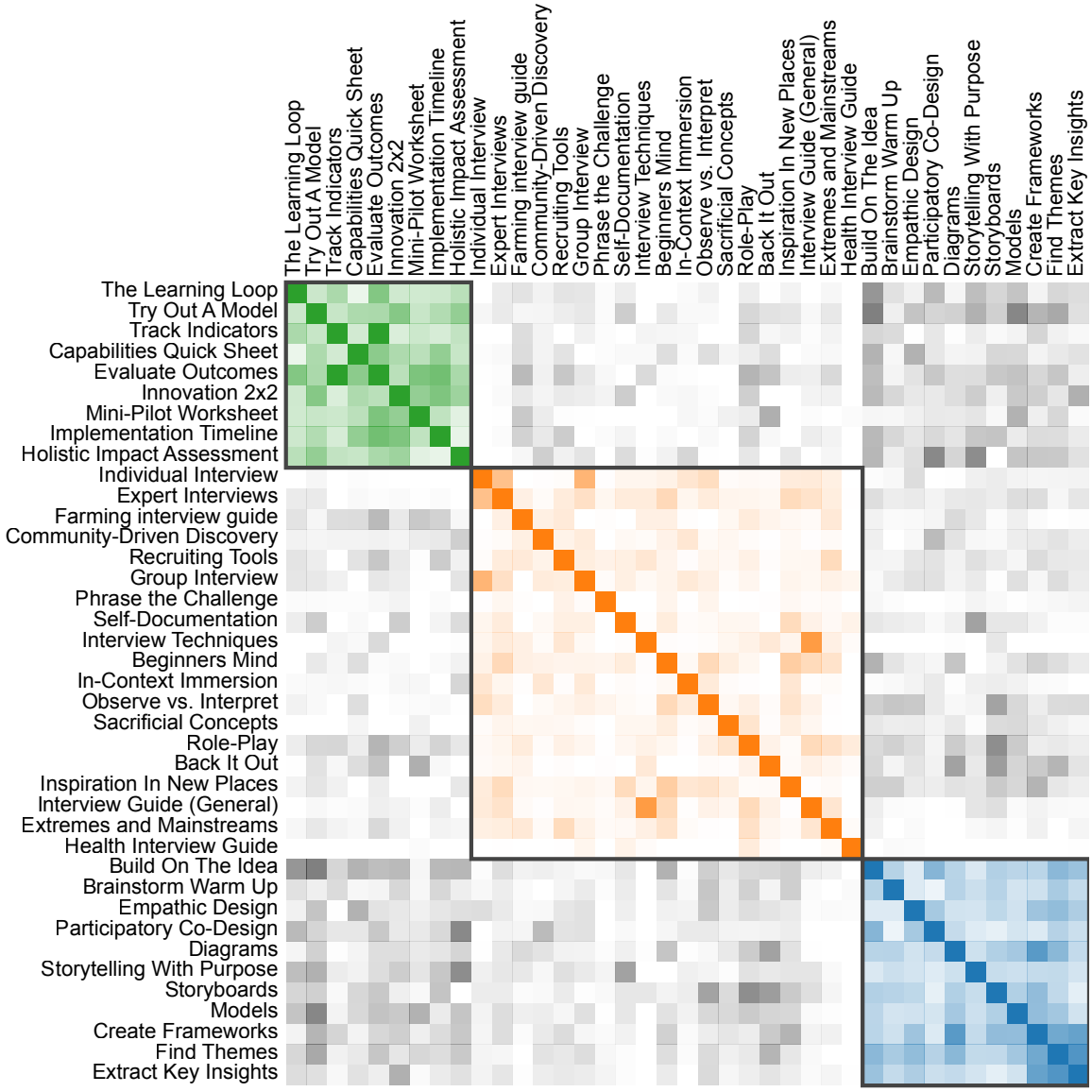


Figure 4.8: Performing spectral clustering on the 39x39 method covariance matrix reveals groups of methods that covary together. Lighter tones represent low covariance, while darker tones represent high covariance. The different hues denote different clusters, with a dark grey box around each cluster of methods. The clusters found by spectral clustering accurately reflect the expert-given categories used by IDEO in their HCD Toolkit; from left to right, the boxes on the diagonal correspond to "Deliver," "Hear," and "Create" methods, respectively.

Support Vector Machines (SVMs): non-linear classifiers that construct a hyperplane in high-dimensional space; it identifies complex boundaries between topics and their interactions.

Logistic Regression: a type of generalized linear model used for the classification; it provides a simple method for determining important text topics (like Random Forests), and is computationally efficient as the number of cases increases.

Naive Bayes: a probabilistic classifier that assumes feature independence across variables and applies Bayes' rule to label categorical data; it also provides a simple method for determining important text topics, as well as easily scaling to a larger number of cases.

Any algorithm hyper-parameters were optimized using randomized search with cross validation using the Scikit-Learn library [99].

4.5.2 Collaborative Filtering

Instead of using problem content to determine which methods are most relevant for a given problem, collaborative filtering approaches analyze the methods that commonly occur together. This section constructs a collaborative filtering model based on Matrix Factorization:

$$f(c, m) = b_c + b_m + < v_c, v_m > \tag{4.1}$$

where $f(c, m)$ represents the score for a particular method m when applied to case c. b_c represents a baseline score for a given case (some cases use more methods than others), and similarly b_m represents a baseline score for a given method (some methods are more popular than others, regardless of the case).

The inner product $< v_c, v_m >$ captures the interaction between methods and cases; v_c and v_m refer to latent dimensional vectors of length k, with a separate vector for each case and method, respectively. For example, using $k = 2$ places each case and method onto a 2-D plane. If the two vectors lie close to one another in the 2-D space, they get a large positive score; if far away, a large negative score.

The algorithm determines the values for k, b_c, b_m, v_c, and v_m, by minimizing prediction error: the mismatch between the methods that it recommends and the methods that were actually used. This section encodes that error through a logarithmic loss function of the following form, where $y_{(c,m)} \in \{1, -1\}$ represents whether or not the case actually used the method:

$$\mathcal{L}(f(c, m), y) = \sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}} \ln \left(1 + \exp(-y_{(c,m)} \cdot f(c, m)\right) \tag{4.2}$$

Evaluating Eqn. (4.1) for each c, m pair yields the expected recommendation score, and Eqn. (4.2) encourages that score towards $+\infty$ for appropriate methods, and towards $-\infty$ for unused methods. The algorithm uses quadratic (L2) regularization to prevent the latent factors from overfitting the training data (improving performance on future data). Combining the loss function in (4.2) with the regularization, the total loss function across the entire dataset becomes:

$$\mathcal{L}(f(c,m),y) + \frac{\lambda}{2}\left[\sum_{c} \|v_c\|^2 + b_c^2 + \sum_{m} \|v_m\|^2 + b_m^2\right] \tag{4.3}$$

In the below experiments, Stochastic Gradient Descent minimizes the combined loss function in Eqn. (4.3), although any descent-based optimizer would suffice since the loss function is convex.

4.5.3 Hybrid Filtering

Hybrid Filtering models incorporate both content and collaborative information by adding case-dependent "focus-area" terms into the collaborative filtering model. Focus areas (tags given by the HCD Connect community) describe which areas the case focuses on, such as "Health," "Education," or "Development" among others (see the first column of Table 4.2 for examples).

To add these content features to Collaborative Filtering, the model gives each focus area its own k-dimensional latent vector (like the methods and cases), and then optimizes the locations of those vectors for each focus area. This section chooses to use focus areas as a content feature because, intuitively, the useful methods for one focus area (*e.g.*, Agriculture), may not be the most useful in a different focus area (*e.g.*, Healthcare). This adds an additional term (v_{cont}) to the Collaborative Filtering model in Eqn. (4.1):

$$f(c,m) = b_c + b_m + < v_c, v_m > + < v_c, \sum_{cont \in c} v_{cont} > \tag{4.4}$$

where $\sum_{cont \in c} v_{cont}$ refers to the addition of all content vectors present in the case. The inner-product $< v_c, v_{cont} >$ acts like the previous inner-product $< v_c, v_m >$, measuring the similarity between the case vector and the combined content vectors. With the exception of the added content vectors, all other aspects of the model are identical to Collaborative Filtering.

4.5.4 Results

On all models, the data were separated into an 80/10/10 stratified random split for training, optimization, and testing, respectively. One hundred iterations of randomized search with 4-fold cross validation optimized all hyperparameters in each model. The best performing parameter choices for each algorithm were tested on the remaining unseen testing data to compute their respective performances.

A Precision-Recall curve is the standard method of comparing different recommender systems [60, 132]. The curve trades off two quantities: precision and recall. Precision is the percentage of recommended methods that were actually used in a case—if the algorithm recommends 10 methods, but only 5 of those 10 were used in the case study, the precision was 50%. Recall is the percentage of methods actually used in a case that were recommended by the algorithm—if the case actually used 8 methods, and the algorithm only correctly recommended 6 of those methods, then the recall was 75%. By changing the number of methods the algorithm was allowed to recommend (*e.g.*, from 0 to 39 methods), one can evaluate the system's performance over a range of precision and recall values—this creates a precision-recall curve, which can be plotted to evaluate an algorithm's performance. This curve essentially summarizes how well the algorithm presented users with meaningful methods for their design problem. (Companies such as Google use similar metrics to evaluate performance for related tasks like web-page ranking [52].) In this use case, higher degrees of precision are more important than higher degrees of recall. Namely, a small set of highly relevant methods is a more valuable recommendation than a complete set including many lower-ranked methods.

For comparative purposes, this section also tests the performance of two baseline algorithms: randomized recommendation and popularity-based recommendation. Randomized recommendation randomly selects k of the 39 methods for recommendation. Popularity-based recommendation rank-orders the most frequently used methods and simply recommends the first k most popular methods, regardless of the case.

Figure 4.9 shows the precision-recall curve for each algorithm. Figure 4.10 demonstrates the 95% confidence bounds (using bootstrap resampling) for the area under those curves—higher area indicates better average precision, and thus better recommendations. As expected, all models outperform the randomized baseline. Popularity performs slightly below that of text-based analysis using Random Forests, while using only a single, efficient predictor. Collaborative filtering uniformly outperforms both the popularity baseline as well as the text-based content features; the added content features in the Hybrid model do not discernibly improve the performance. (Future research may uncover different content features that positively affect performance.)

In addition to general performance, one might also be more interested in how the algorithms perform on more specific or uncommon methods. For example, a designer

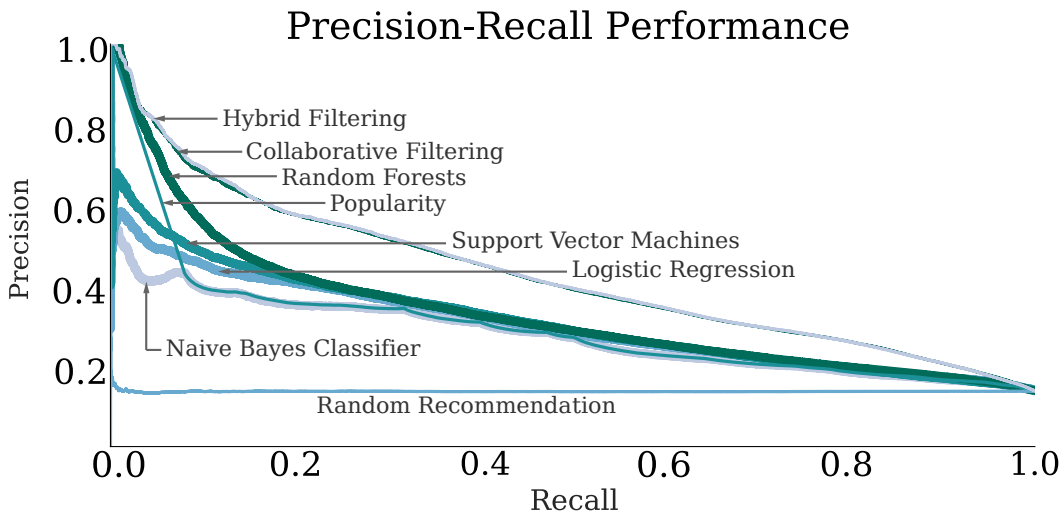


Figure 4.9: Precision plotted as a function of recall. The higher the area under the curve, the better the algorithm's performance.

would likely use popular methods regardless (possibly out of habit), but might only use certain uncommon methods when particularly appropriate—a successful method recommendation algorithm should perform well over uncommon methods, as well as popular ones. To test this, Fig. 4.10 computes the precision-recall performance on the ten least frequently used of the 39 methods, and integrates the area under the precision-recall curve; this total area is called the AUC, for Area Under the Curve, and measures overall recommendation performance (higher AUC is better). Collaborative and Hybrid filtering still perform significantly better than the alternatives.

4.6 Implications for Design Processes

This chapter raises several points regarding the role of community-generated data in the study of design processes, both at the level of understanding a given design process and for recommending design methods for new problems.

4.6.1 Implications for Understanding Design Processes

Sections 4.3 and 4.4 both demonstrate how aggregating method usage across a corpus of case studies can lead to insights about design methods and processes. For

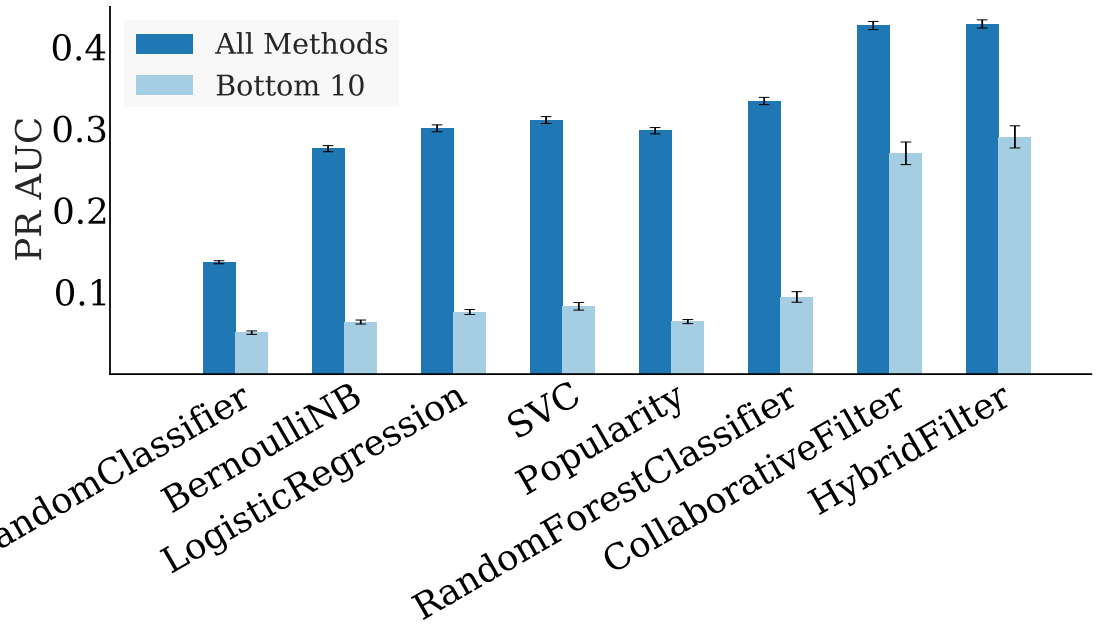


Figure 4.10: The area under the precision-recall curve (AUC) across the models. The error bars represent the 95% empirical confidence bounds about the median AUC for each method, calculated using bootstrap resampling. The hybrid and collaborative filtering models perform substantially better than the popularity baseline. The Random Forest classifier produces a detectable, but small, improvement over the popularity baseline

example, in regards to the design for development projects, statistical analysis techniques highlight the following two implications for designers: determine whether your particular problem requires a specific type of method before diving in, and compare the covariance between methods you use to uncover groups of methods that might complement one another.

Determine whether your particular problem requires a specific type of method before diving in. Figure 4.5 and Table 4.5 used techniques from large-scale hypothesis testing and False Discovery Rate Control algorithms to demonstrate how certain methods work well in particular problem types; One area open to debate lies in determining an appropriate minimum effect size: is a 17% increase in a method's usage important enough? At what threshold is a focus area's effect on a method too large to ignore? Many methods *did not* differ among problem types—this points to a dichotomy between general-purpose methods and problem specific methods that can be illuminated by the data-driven techniques in this chapter. The analysis in section 4.3.2 analyzed relationships between methods and a single factor: the problem focus areas. That type of analysis can easily be extended to other factors, such as budget constraints, time requirements, or applicability to different types of users, for example. Some research has begun to map out factors that might differentiate methods [108] and would be a natural extension of the work in this chapter; both quantitative and qualitative work would complement these existing results.

Compare the covariance between methods you use to uncover groups of methods that might complement one another. Fig. 4.4 and Table 4.3 demonstrate that methods are not independent from one another. Understanding how methods relate to one another, whether by automatic means (such as correlation coefficients) or through qualitative study, would allow one to make more strategic method choices. For example, if you know that Storyboards better complement Role-Play over Group Interviews you can make smarter user research choices and trade off breadth for depth. Section 4.4 showed that features, such as method covariance can accurately group related methods together—in this case replicating human-given groupings to 92% accuracy. This points to using method co-usage as a similarity criterion when attempting to decompose methods into different types.

One limitation of Spectral Clustering is that its groupings are only *single membership* (*i.e.*, methods can only belong to one group), rather than *mixed membership*. This assumption is good for some types of categorizations, but not for others; for example, when methods might be exclusive across different design stages, such as the HCD Toolkit, single membership models will be fine. Future work should certainly explore more complex methods of clustering or grouping methods, including using a

combination of collaborative and content features to define method similarity as well as mixed membership clustering models.

An implication brought to light when comparing Tables 4.3 and 4.4 is the effect of segmenting one's data when analyzing method covariance. In that case, segmenting the case studies by phases presented a different and complementary view of a similar ideas: by performing clustering on covariance between methods, one can uncover grouping that relate the methods. In the one case (Table 4.3), the methods could be divided by design phase, which is temporal, whereas when one removes that temporal aspect (Table 4.4), the methods instead group by how they are used (Visual methods vs. Interviewing methods, *etc.*). This difference is a strength, rather than a weakness of clustering methods, as they provide different means to understand the complex relationships between methods beyond a single factor, such as time.

4.6.2 Implications for Recommending Design Methods

Section 4.5 offers up several possible implications for design method recommendation systems. First, one should not ignore collaborative features in favor of text features, especially when the collaborators have a reasonable level of expertise. Second, future research needs to maximize the benefits of combining content and collaborative features.

Collaborative features have higher predictive accuracy than text-based features. Comparing the precision-recall performance, collaborative-based approaches perform substantially better than the content-based approaches that relied solely on text features. This was unexpected, given the prevalence of text-based recommendation for ranking documents. However, given the use case, it is also understandable—the time needed to apply a method or the people required to execute it (among many other factors) could both affect a method's usage in ways not discernable from the case's description.

One possible explanation for the fact that the content-based features offered little improvement is that the methods and focus areas could be too general to meaningfully distinguish themselves. For HCD Connect, the above sections demonstrated that a small subset of methods do occur more frequently depending on the specific focus area, so one would expect the addition of focus areas to have a meaningful effect. That said, a more thorough description or ontology of methods that accounts for these differences between methods or categories may improve future performance of content-based recommender systems, and some recent work has begun to collect this information [108]. Incorporating improved content features would be a fruitful area of future research.

Combinations of the features offered only marginal improvement. Combining content features (in the form of focus areas) with collaborative filtering did not offer a detectable improvement in performance. This could be attributed to the choice of content features, or possibly to the choice of model in Eqn. (4.4) (although similar models are effective in other domains [8]). More advanced collaborative filtering models, such as Bayesian Probabilistic Matrix Factorization models, could also improve recommendation performance by incorporating prior knowledge or more sophisticated content features [82, 109].

4.6.3 Generalizing beyond HCD Connect

Although this chapter has evaluated recommendation systems against the HCD Connect dataset and its corresponding methods, the proposed algorithms do not depend on the choice of this specific dataset; they are agnostic towards the choice of methods and evaluate only the relationships between cases and methods. This means that design method recommendation systems (section 4.5) can be generalized beyond HCD Connect and user research methods at the beginning of the design process. Design practitioners can use these techniques for other classes of design methods, such as design optimization methods, mechanical design techniques, functional synthesis, and more, so long as one can find appropriate case studies that use those methods.

Likewise, the clustering methods in section 4.4 are designed to transfer to different types of methods or domains. When doing so, covariance will still remain important, though the single-membership clusters assumed by Spectral Clustering might not perform well in domains where methods can be clustering along many dimensions. In these cases, more complex clustering models, such as Co-clustering (clustering both methods and cases) [30] or Infinite Latent Feature Models (uncovering latent features given method similarities) [54] could provide more insight.

4.7 Summary

This chapter covered how to use data-driven techniques to infer properties of design processes that would be difficult to perform manually. It focused on the HCD Connect community, which applies front-end user research methods to design for development projects. Specifically, the chapter tackled three tasks: finding relationships between methods and how they are applied; grouping methods into logical categories; and recommending design methods for new problems.

Using techniques from large scale hypothesis testing, such as False Discovery Rate Control algorithms, section 4.3 showed how community generated case studies can shed light on how methods are applied to different problems. It identified methods

that complement one another or those that are particularly useful in certain types of focus areas (*e.g.*, Agriculture, Community-Development, *etc.*), as well as illuminated differences between IDEO and non-IDEO designers.

By leveraging how methods covary with one another, section 4.4 demonstrated how Spectral Clustering could categorize methods into groups that matched human-given labels to within 92% accuracy. This points the way towards using community-generated data to improve design education around different types of methods.

Lastly, section 4.5 described how to apply techniques from recommender systems to help designers select better methods for new problems. It proposed recommender systems that leveraged both the content of design problems and how methods covary. Of those systems, Collaborative Filtering algorithms outperformed others, although future research could address other content features that may improve performance for all systems.

Beyond quantitative analysis, future work could address several complementary qualitative questions. Further content analysis of the case studies themselves could elaborate why particular methods were chosen, along with what worked well or poorly. Another helpful next step would be to establish a better qualitative understanding about why certain methods were chosen for particular types of problems (*e.g.*, Farming Interview Guide for Agriculture versus Participatory Co-Design for Community Development). Including a wider set of methods and cases from a dataset such as theDesignExchange [108, 107] would broaden the above analysis to outside of design for development methods.

With both a quantitative and qualitative picture of how user research methods are applied in design for development projects, one can be better equipped to make the right resource decisions when embarking on design for development projects, creating better products and services by making sure that designs address the correct user needs.

Thus far, the dissertation has looked at overall design communities, how they form, and how one can understand design processes within that community. The next chapter looks within a design process at the resulting ideas that community members generate. It explores how to evaluate those ideas in a way that scales to the size of typical online design communities.

Chapter 5

Evaluating Design Ideas

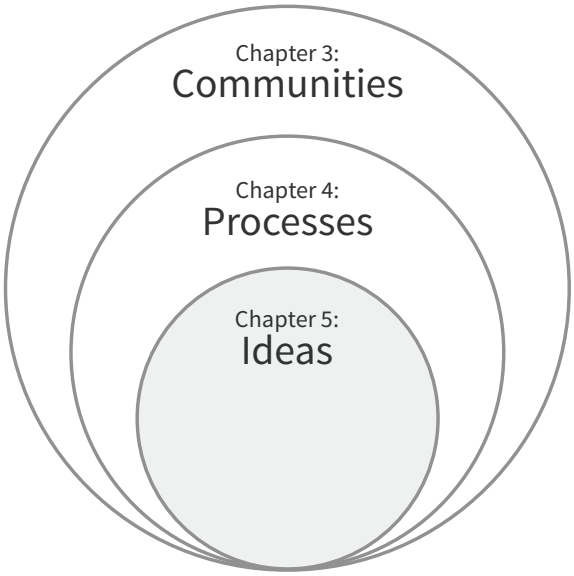


Figure 5.1: This chapter addresses the role of data at the design idea level: how does one evaluate the creativity of thousands of ideas generated by an online community?

Chapters 3 and 4 demonstrated how online design communities form and move through a design process to generate ideas. This chapter now moves to the individual idea level: assuming there is a vibrant community that is generating tens of thousands of ideas, how does one possibly evaluate all of them? Specifically, this chapter answers the question "how does one evaluate the creativity of ideas in a way that scales to tens of thousands of ideas?"

Portions of this chapter appear previously in [45]

To address scalable idea creativity evaluation, the chapter describes how aspects of creativity (*e.g.*, Novelty, Variety, Surprise, *etc.*) connect to a fundamental mathematical object called a submodular function that models diminishing marginal utility. By viewing creativity through the lens of submodular functions, the chapter proposes a type of logistic regression model that can leverage human evaluations of creativity (*e.g.*, "design portfolio A has more variety than design portfolio B", or "design C is more novel than design D") to predict the creativity of a new idea or set of ideas. This allows a small sample of human-given evaluations to train a model that can scale to tens of thousands of ideas or more, combining the strengths of human experts with the scalablity of a computer algorithm. This type of model provides an efficient way to evaluate several orders of magnitude more ideas than was previously feasible.

To validate the model, experiments demonstrate how the algorithm accurately recovers the existing variety metrics of Shah *et al.* [118] and Verhaegen *et al.* [131] to an average of 97.5% accuracy after 500 binary ratings. The chapter also presents results regarding the convergence rate of the algorithm and its robustness under increasing signal-to-noise ratios. Throughout, this chapter will use variety metrics as the working example, though section 5.4.1 describes how the proposed model is extensible to any metric that depends on diminishing marginal utility.

5.1 Related Work

In addition to the background covered in chapter 2, this chapter builds upon two additional bodies of work: 1) design creativity metrics, and 2) submodular functions.

5.1.1 Measuring Design Creativity

If the goal is to reliably and accurately evaluate the creativity of a large set of crowd-generated ideas, this chapter first needs a concrete definition of what design creativity is, and how one can measure it. This is not a straightforward task because researchers have yet to reach consensus on what design creativity metrics are appropriate [13, 16, 26, 47, 18, 101]. Saunders and Gero [113] draw distinctions between what unit creativity is being defined over, such as whether one wants to measure creative design outcomes, design processes, people, or environments. For example,

Creative Design Outcome: people viewed the iPhone as a creative product, when it first appeared, due to its novelty with respect to existing phones.

Creative Design Process: the design consultancy IDEO is often described as having a creative process (or means of producing ideas), irrespective of a particular design itself.

Creative Design Person: famous designers, such as Jonathan Ive and Philippe Starck, can gain reputations for being particularly creative in and of themselves.

Creative Design Environment: certain workspaces, irrespective of the designers or products within that space, can be viewed as intrinsically promoting greater creativity than other workspaces—for example, visitors often remark "this looks like a creative place" upon entering the Berkeley Institute of Design.

Boden describes creativity along a different dimension: historical (H) vs psychological (P) creativity [13]. If an object possessed **historical creativity**, it would be novel with respect to all previously known ideas, whereas if it possessed **psychological creativity** it would only be novel with respect to the particular individual evaluating the idea. For example, the first generation iPhone might be viewed as historically creative, since, at the time of its inception, no other phone had ever permitted anything close the functionality or experience it provided; later smartphones would seem less historically creative by comparison. However, to a seven year old, who has lived around smartphones her entire life, the first generation iPhone may seem antiquated, and not nearly as creative as the newest iPhone model with the latest features; the first generation iPhone would have lower psychological creativity.

In relation to this larger body of work, this chapter considers outcome-based metrics (*e.g.*, the novelty of a particular design) as judged in a P-creative sense (*i.e.*, from the standpoint of an individual's assessment) [13]. This is the primary benchmark that many organizations investing in online design communities care about: will present-day individuals find this particular product or service creative and different? For outcome-based metrics, there have been two primary approaches that past researchers have taken to model creativity: model-based metrics and human judgement-based metrics.

5.1.1.1 Model-based Metrics

Outcome metrics that are **model-based** attempt to mathematically predict the creativity of a set of designs. These come in many kinds, with the most widely used being hierarchical models followed by graph models.

Hierarchical models measure creativity for sets of designs by encoding the set as levels in a genealogical tree. An outcome metric, such as variety, is then calculated by summing over parts of the tree. For example, a founding metric of this type by Shah *et al.* [118] encodes concepts into a tree of functions, and then breaks down creativity into four additive sub-metrics: the quantity and variety of the set as a whole, and the quality and novelty of each idea individually [118].

Several researchers have since altered Shah's hierarchical model for various reasons: Nelson *et al.* [87] offer a refined version that fixes several modeling errors;

Verhaegen *et al.* [131] combine Shah's metric with a tree entropy penalty, called the Herfindahl index, to encourage "uniformness of distribution"—essentially preferring trees that have even branching; Chakrabarti *et al.* [111] propose to ground Shah's approach in a broader set of functional categories.

Graph-based outcome metrics take a similar approach, but instead of breaking down designs into genealogical trees they compute graph features using attributes like similarity or cluster distances and combine weighted sums of those features. For example, Maher [81] defines novelty as how far away a new concept is from clusters of previous concepts, where the clusters are created using a prior concept similarity graph.

The benefit of model-based metrics is that they are easy to compute once someone encodes the features of the ideas, and thus scale well to large design communities. This also makes them reproducible, given the same concepts. Their main disadvantage is that each metric has to be custom designed for a particular domain and application; for example, Shah *et al.*'s metric [118] only works for simple mechanical devices based on hierarchical function decomposition. This makes it difficult to adapt existing work to new types of design or different domains.

5.1.1.2 Human Judgement-based Metrics

Outcome metrics that are **human judgement-based** assume that the full extent of what defines creativity cannot be captured in a mathematical model, and that the judgement regarding what is creative is best given by humans themselves. As a result, these approaches typically use a small set of human raters, often domain experts, who manually rate designs on a Likert-type scale. The desired outcome metric (*e.g.*, novelty, variety, usefulness, *etc.*) is then computed as some combination (typically the average) of the human ratings. Metrics that fall under this category include Amabile's Consensual Assessment Technique [5], Carrol *et al.* 's Creativity Support Index [22], and the Creative Product Semantic Scale [93]. Oman *et al.* [92] offer a comprehensive comparison of different examples of this class of metrics: different methods of evaluation include scale ratings, flow charts, novel models, adjective pairings, and A/B tests. Each of these methods tests for different components of creativity, including novelty, need satisfaction, attention to detail, functionality, and availability of existing solutions.

While human judgement-based metrics have excellent validity (a high score, by definition, is what real humans considered P-creative), they suffer from two fundamental challenges: reproducibility and expense. Even if it were possible for multiple studies to utilize the same expert raters, differences in knowledge or attitude at the time of rating can make the human evaluators inconsistent with prior ratings. This makes it next to impossible to exactly reproduce findings from other studies, even

if the concepts themselves are the same. Indeed Srivathsavai *et al.* [122] found that inter-rater reliability between experts can be low, depending on which aspects of creativity are being evaluated. More importantly, the collection of expert ratings is expensive, requiring multiple raters for every single concept considered. This makes it difficult to judge creativity on a large scale consistent with online design communities.

The model proposed in this chapter falls under the category of a model-based metric. However, it differentiates itself from prior work in that it fits the model to human judgement data automatically. This requires a small amount of expensive human evaluation data, but then pays off with a reproducible metric that has high external validity.

5.1.2 Submodular Functions

Thus far most of the algorithms or models this dissertation has covered are general enough to use for any type of function or statistical learning situation. However, what if the domain of the problem restricts the types of functions that are useful? In this chapter, problems related to design creativity often obey a property called **diminishing marginal utility**. This property states that the more you have of something, the less each additional unit is worth to you. For example, if you are given the choice between a piece of chocolate and an apple, you might, at first, choose to eat the chocolate. However, eating after your tenth piece of chocolate, you might be so tired of eating the same thing that the apple becomes a more appealing choice.

To model diminishing marginal utility, this chapter uses a particular type of mathematical function called a **submodular function**. If a function is submodular then if one has a set of items A, and adds x to it, there will be a greater increase in value than if one had the set $\{A \cup x\}$ and added x' to it—the more things one adds to A, the less each addition is worth. A common example of a submodular function in one dimension would be the logarithm (for each positive δx, δy decreases). This definition is where submodular functions gain their usefulness: it is identical to the principle of diminishing marginal utility. Formally, submodular functions are set-based functions where, for a function ρ and two sets $A, B \in \Omega$: $\rho(A) + \rho(B) \geq \rho(A \cup B) + \rho(A \cap B)$. This is similar to the behavior of the logarithm as described above, except that A and B are sets, rather than a continuous variable x.

Recently machine learning researchers have adapted submodular functions to solve large-scale problems involving diminishing marginal utility. Information overload on the web opened up opportunities in webpage retrieval for developing algorithms that recommended an optimally diverse set of relevant webpages to users (rather than just the most relevant). This led to formally defining the idea of "coverage" for a set of documents—the extent to which a set of items covers all possible elements. Finding the set of documents with maximum coverage is called the **Maximum Coverage**

Problem, and has been proven to be NP-Hard.

To efficiently approximate the Maximum Coverage Problem, submodular functions were applied to document retrieval by El-Arini *et al.* [34] in order to find the optimal approximate solution. This approach was particularly interesting, since various lower bounds on the performance of submodular functions [88, 70] proved that they provide the best-possible approximation to solving the Maximum Coverage Problem. Since that time, others have built upon the use of submodular functions for diverse retrieval, notably the work by Ahmed *et al.* [2], upon which the model in section 5.2.2 is based.

Using these submodular functions, this chapter combines the advantages of both model-based and human judgement-based approaches by using an easily computable and expressive metric that can be automatically trained from collections of human judgements. Specifically, it discusses how many existing model-based metrics are based on minor variations of diminishing marginal utility, and presents a model that ties these existing metrics together under a general theory. This creates strategies for mitigating the two main disadvantages of model-based metrics. By training this model on collections of human judgements, the model can provide the external validity of human assessment with the computational friendliness and repeatability of model-based metrics. Moreover, by generalizing many prior model-based metrics as special cases of diminishing marginal utility, this model allows researchers to adjust existing model-based metrics to better match human assessment.

5.2 Variety Model

This chapter's core insights lie in the following connections:

1. Many common elements of creativity, such as variety or novelty, are naturally expressed via the principle of diminishing marginal utility.

2. Diminishing marginal utility can be expressed in a computationally advantageous way via submodular functions.

3. Submodular functions can easily utilize many of the design representations used in current creativity metrics.

4. Given a set of designs, human experts have a hard time agreeing on real-numbered values (*i.e.*, interval scales) for its creativity, while it is fairly easy for them to make binary "greater than" or "less than" (*i.e.*, ordinal scale) judgements.

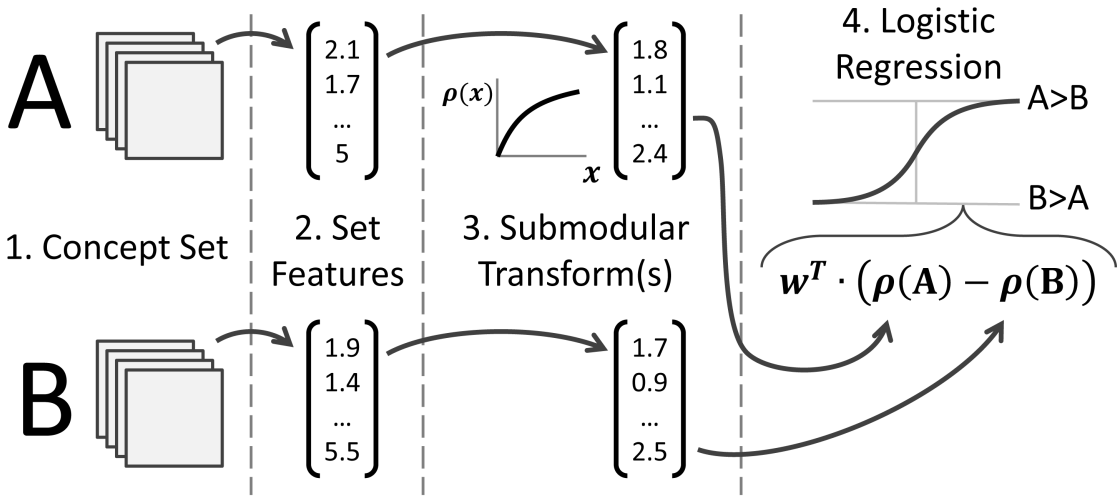


Figure 5.2: The overall approach 1) takes in design representations of two set of concepts (A and B), 2) encodes each of those sets into a vector of features describing the set, 3) transforms those features through one or more submodular functions ($\rho(A)$, $\rho(B)$), thereby introducing diminishing marginal utility, and finally 4) determines which of the input sets has greater variety by using a weighted ($\mathbf{w}$) difference ($\rho(A) - \rho(B)$) between the submodular features of the two sets. The weight of each submodular feature ($\mathbf{w}$) is optimized through a logistic regression such that the model matches expert-rated comparison data as closely as possible.

These connections drive the approach described below and shown in Fig. 5.2. A submodular function approximates variety by using a design feature vector along with a set of human judgements to identify how much the presence of each design feature impacts creativity.

5.2.1 Connecting Creativity, Diminishing Marginal Utility, and Submodular Functions

To see how creativity, diminishing marginal utility, and submodular functions are related, return to the task of estimating variety: Using a variant of the example presented in Shah *et al.* [118], suppose you have a set of student generated designs whose purpose is to move an object from point A to point B, and you want to select the two designs from that set which have the most variety. For simplicity, assume that there are just three designs: 1) a small cart that propels itself forward using a balloon filled with air, 2) a similar cart, but propelled using a fan, and 3) a small catapult. Say that you choose the first cart as one of your two final choices, which

of the other two design do you pick? Since you already have a balloon-propelled cart design, it doesn't give you much value to pick the second cart, which also uses a form of wind propulsion. This additional value is your marginal utility, and it is diminishing because the second cart design is not as valuable to you as the first (even if they are equally good designs). On the other hand, selecting the catapult to go along with the first cart would give you higher marginal utility, since it is a completely new way of transporting the object.

Various existing metrics try to address this notion of diminishing marginal utility in different ways. Hierarchical metrics such as Shah *et al.* [118] and subsequent work [111, 87] represent this principle by assigning a higher reward for solutions at higher functional levels. Verhaegen *et al.* [131] take those metrics a step further by accounting for the entropy of the concept distribution, which is similar in purpose to diminishing marginal utility. Maher [81] models it as a reward for greater aggregate distance from existing cluster centers. Whether a discrete or continuous space, the idea remains the same: if a new idea is similar to what one already has, it is less valuable—it has a diminished marginal utility.

As described above, submodular functions are perfect surrogates for modeling diminishing marginal utility, making them a natural choice for expressing any form of creativity which depends on diminishing marginal utility. In order to operationalize this new knowledge, this chapter first needs to address the following questions: 1) how, specifically, are aspects of creativity expressed as submodular functions, 2) what does the input to these submodular functions look like and how are real-life designs encoded into this input, and 3) how does one use all of this to emulate human judgments?

5.2.2 Modeling Creativity with Submodular Functions

As with almost all other published creativity metrics, the fundamental model here is a linear model where the outcome metric is modelled as a vector of weights multiplied by a vector of features. Returning to variety as the example: $\text{variety}(A) = \mathbf{w}^T \cdot \mathbf{d}(A)$, where A is the set of ideas, $\mathbf{d}$ is a vector of numbers summarizing the features of a set of designs, and $\mathbf{w}$ is a vector of weights for each feature. In prior metrics the feature weights ($\mathbf{w}$) are typically set to some constant value (*e.g.*, Shah *et al.* [118] choose $\mathbf{w} = [10, 6, 3, 1]$).

This is where this work departs from prior work, by using submodular functions ($\rho(x)$) to transform the design features such that they obey specific forms of diminishing marginal utility: $\text{variety}(A) = \mathbf{w}^T \cdot \rho(\mathbf{d}(A))$. This section applies a variant of the model of Ahmed *et al.* [2] for the purposes of modeling design creativity.

Formally, the submodular score for a set A is given by

$$f(A) = \boldsymbol{\alpha}^T \cdot \mathbf{d}(A) + \boldsymbol{\beta}^T \cdot \rho(\mathbf{d}(A)) \tag{5.1}$$

where the weight vector $\mathbf{w}$ has been broken into two parts: $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, representing the modular and submodular contributions to variety, respectively. This has the advantage of allowing the algorithm to automatically determine the extent to which a feature obeys diminishing marginal utility. Either $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ can be forced to zero to use only the submodular or modular parts, respectively. For the example of variety, this chapter assumes that it behaves fully sub-modularly, and sets $\boldsymbol{\alpha} = 0$, resulting in $f(A) = \boldsymbol{\beta}^T \cdot \rho(\mathbf{d}(A))$. Altering standard notation slightly, this section defines $\rho(\mathbf{d}(A))$ to be a vector where ρ has been applied to each element in the vector $\mathbf{d}(A)$.

While any submodular function can be used for ρ, some useful options given by Ahmed *et al.* [2] include:

Set Cover: $\rho(x) = 1$ if $x > 0$; 0 if $x = 0$

Probabilistic Cover: $\rho(x) = 1 - e^{-\theta x}$ for $\theta > 0$

Logarithmic Cover: $\rho(x) = \log(\theta x + 1)$ for $\theta > 0$

This chapter demonstrates in the experiment section below that the metrics of Shah *et al.* [118] are a special case of this chapter's model where $\rho =$ set cover, while the metric of Verhaegen *et al.* [131] is well approximated by $\rho =$ probabilistic cover.

5.2.3 Encoding Design Concepts

With some candidate submodular functions at hand, the next step is to define $\mathbf{d}(A)$, *i.e.*, how a specific set of designs (A) becomes a vector of numbers that can be used by the submodular function—a process this section refers to as *encoding* the design concepts. This encoding can be any set of real numbers which summarize different aspects of a design or set of designs, so long as a consistent encoding is used across all concepts.

For example, in Shah *et al.* [118] a set of designs is encoded as a functional tree decomposition where each of four levels is summarized by the number of bifurcations in a particular level of the tree—these form four features, summarized as four real numbers, that describe the set. Various authors have proposed different sets of encodings, but the end result is the same: designs become a feature vector of real numbers ($\mathbf{d}(A)$) that eventually get used in a linear model.

This is important for the model, because as long as one can take any given design or set of designs and encode it as a vector of features, this chapter's model can use that vector to determine how each of those features impacts creativity. As Fig. 5.3 demonstrates, this approach works for linear design encoding, such as the standard linear logit models commonly used in consumer preference models, hierarchical encodings, such as Shah *et al.* [118] and others [111, 87, 131], and graph-based encodings,

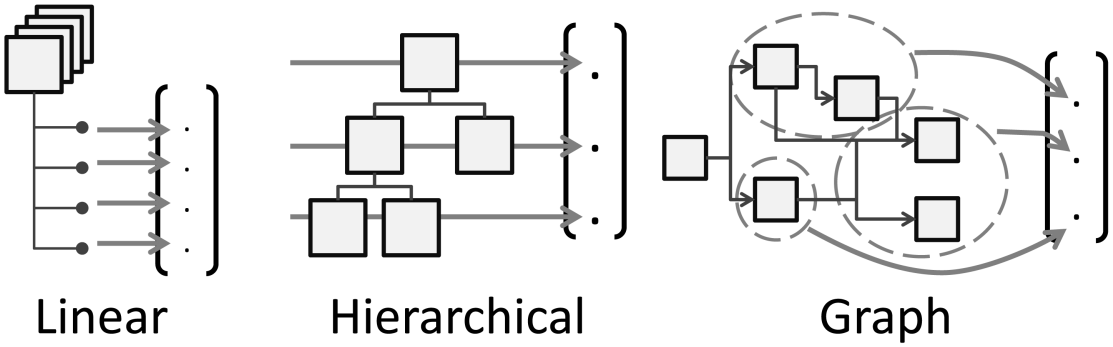


Figure 5.3: This model can encode more than just hierarchical creativity metrics; it can handle graph-based creativity features, over which linear and hierarchical features are a subset.

such as the cluster model of Maher [81] or the Function-Behavior-Structure (FBS) model [48].

This grants the model some flexibility: by picking the appropriate encoding, one can use this chapter's model to measure the creativity of any aspect of design one wants, including as a drop-in replacement for many existing metrics, or one can use the model to compare different encodings and determine which is best for a given problem.

5.2.4 Model Inference

Given the above model and a particular encoding, the next task is to estimate the weights $\mathbf{w}$ (or for variety just $\boldsymbol{\beta}$) and any submodular hyperparameters (*e.g.*, θ) using a dataset of human given ratings. Given perfectly aligned and consistent human raters, this would be possible through simple linear regression by asking human judges: "on a scale from zero to ten, how much variety does this concept set have?" Unfortunately, this task is next to impossible to achieve in practice, since different humans have different anchoring points or opinions about what justifies variety, making simple numerical answers difficult to interpret.

An easier task for a human evaluator is that of comparing two sets: "Given a set of concepts A and another set B, which set has greater variety?" This is still prone to differences in opinion or background (as are all creativity metrics that depend on human evaluation), but is less prone to differences in absolute measure between individuals. A possible alternative could be ordinal ranking of more than two sets, which is easily translated to binary greater than/less than judgements.

Given a dataset of binary judgements between various pairs of sets, standard logistic regression can determine the optimal weights $\mathbf{w}$ that best match the judgements

given by human experts. Formally, the likelihood function for predicting whether a human would rate a set $A > B$ is given by:

$$\mathbf{P}(A > B|A, B) = \left[1 + e^{-(f(A)-f(B))}\right]^{-1} \tag{5.2}$$

where $f(A)$ and $f(B)$ are given by Eqn. (5.1). Using the entire dataset, maximum likelihood estimation on the above likelihood function determines the optimal weights. The value of the hyperparameters, if needed, can be determined either through grid search or through stochastic gradient descent for certain forms of the submodular function.

5.3 Experimental Results

To assess the validity of this approach, data is used to train the model to predict which of two randomly generated concept sets had greater concept variety, as derived from two different variety metrics, Shah *et al.* and Verhaegen *et al.* . These metrics are used to simulate human judgements for the purpose of testing the model and to make these results reproducible by others[1].

This chapter uses Shah's metric because of its broad adoption within the community and because it provides a useful example of how hierarchical metrics can be encoded as a linear model. Verhaegen's metric was chosen since it attempts to account for "uniformness of distribution" within the hierarchical branches using the Herfindahl index. This is similar in spirit to modeling diminishing marginal utility, and makes that metric a natural candidate with which to assess the utility provided by different submodular functions.

The results in this section were generated using the following experimental procedure:

1. Select a variety metric to simulate human judgements (*i.e.*, Shah or Verhaegen).

2. Randomly generate multiple sets of 10 concepts and calculate their variety with respect to the chosen metric. These are stored as the ground truth variety scores ($V(X)$).

3. For each set of 10 concepts, transform its feature vector using a submodular function (set cover for Shah, probabilistic cover for Verhaegen)—this is $\rho(X)$.

[1]Full experiment code is available here, for those who wish to replicate or extend the results: http://www.markfuge.com/dissertation

Metric	n=100	200	300	400	500
Shah	95.2	96.9	97.6	98.1	98.6
Verhaegen	91.9	94.4	95.5	96.0	96.3

Table 5.1: In both cases, the algorithm was able to achieve 95% prediction accuracy in less than 300 samples. Randomly guessing achieves a baseline score of 50%.

4. Use Eqn. (5.1) to determine the difference vector between two sets of concepts—$\mathbf{x}_i = \{\rho(A) - \rho(B)\}$ in the case of fully submodular features. These become the input features for the logistic regression.

5. Recall the ground truths for each set (A & B). The decision as to whether the variety of A is greater than B is then decided by $y_i = \text{sign}(V(A) + \epsilon_A > V(B) + \epsilon_B)$, where $\epsilon = N(0, \sigma)$. This becomes the classification label for the logistic regression.

6. Steps 4 & 5 are repeated for as many training samples as desired ("Number of A/B Comparisons used in training" in Figs. 5.4-5.6)

7. Using the matrix of noisy binary ratings from step 5 corresponding to submodular difference vectors from step 4, use logistic regression to learn the optimal weights ($\mathbf{w}$). Evaluate the model on unseen test data via 30 randomized cross-validation trials and compare the predicted decisions with the original ground truth labels to determine prediction accuracy.

Using the above procedure, the model generated the following results:

Figures 5.4 and 5.5 demonstrate the convergence and robustness of the model under Shah *et al.* 's and Verhaegen *et al.* 's metrics, respectively. As Table 5.1 shows, in both cases the algorithm converges to above 90% accuracy within the first 100 ratings, and to above 95% accuracy within the first 300 ratings.

In the presence of noise, the convergence rate is slower, as expected. In both cases, the model ultimately is able to recover the underlying metric to high accuracy, given sufficient training samples.

Using different submodular cover types unveils differences in the behaviour of each metric: In the case of Shah's metric, the usage of set cover makes the model equivalent, and thus it captures the metric with complete accuracy. Using probabilistic cover reduces the accuracy, since Shah's metric does not encode diminishing marginal utility within each level of the tree. Under Verhaegen's metric, using set cover does not capture the model as accurately as using probabilistic cover, since their metric does attempt to encode diminishing marginal utility within each level of the tree. Both of these results confirm expectations. Logarithmic cover and probabilistic cover achieve

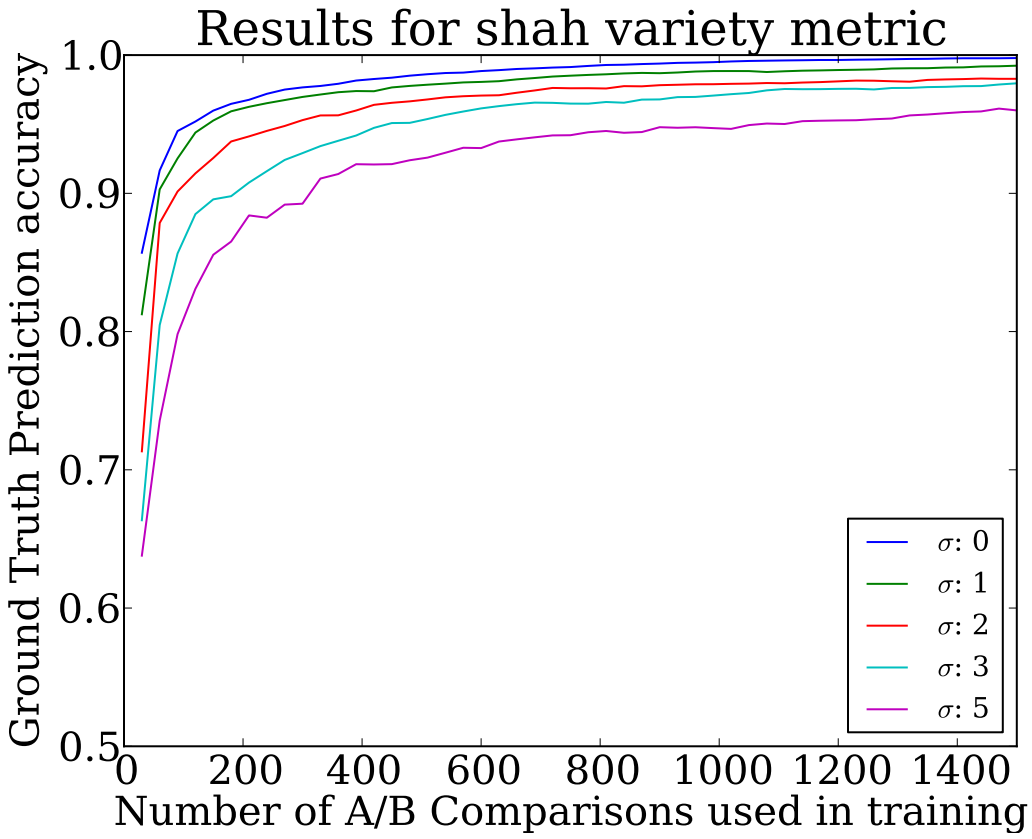


Figure 5.4: With no error (top-most curve), the model recovers the Shah metric to within 95% by 100 ratings. As expected, when more random error is added, the algorithm's convergence is slower. Even with substantial noise, the model can recover the true variety score to good accuracy, given enough samples. Randomly guessing results in a prediction accuracy of 0.5 (50%).

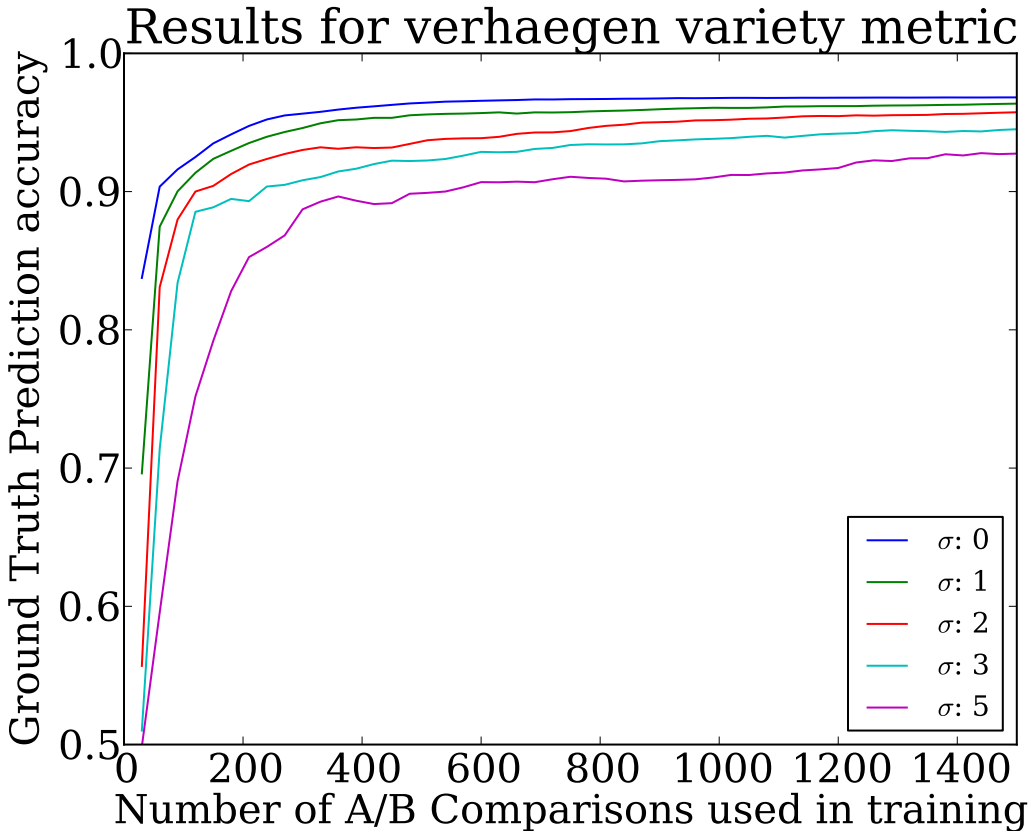


Figure 5.5: Similar results were seen with the Verhaegen metric. The average scores are lower, since, unlike Shah's metric, Verhaegen's metric is not perfectly contained within the model class. Regardless, the model still achieves 95% prediction accuracy within 300 samples. Randomly guessing results in a prediction accuracy of 0.5 (50%).

similar results in both cases, so only probabilistic cover is shown to improve figure clarity.

5.4 Implications for Evaluating Design Ideas

The above results raise the following questions for discussion:

1. To what extent does this model extend to aspects of creativity other than variety?
2. How would these results fair under actual human evaluation, instead of simulated sources?
3. What do these results mean for other work in Design Creativity measurement?

5.4.1 Extensions to Other Aspects of Creativity

This chapter provides two possible avenues for extending existing metrics: 1) testing other structures and domains for variety, and 2) modeling other aspects of creativity, such as novelty and usefulness. In each of these cases, the resulting mathematical model and inference procedures remain unchanged, with the only change being how design concepts are encoded. This results in a general procedure for experimenting with various encodings and datasets.

The experiments presented in this chapter focused primarily on hierarchical variety metrics, as these are currently the most commonly used in analyzing ideation results in mechanical engineering design. However, as mentioned in the above sections, this model extends to any variety description that can be expressed as a linear set of features. This opens the door for future work to analyze both different formulations of variety on existing domains, as well as how existing metrics transfer to different domains. For example, a new graph-based metric could be applied to studying sets of FBS models or patent networks to determine which features accurately predict variety. Likewise, by training on a different set of experts, a hierarchically structured metric similar to Shah *et al.* could be adapted to describe functions in a organizational or service context, instead of purely physical functions, extending the applicability of existing work.

The presented model can also be adapted to describe any creativity metric that utilizes a form of diminishing marginal utility. For example, novelty metrics (*e.g.*, [118, 111, 81]) or surprise (*e.g.*,[81]), can be recreated simply by altering which sets are used for training.

Novelty would be the marginal utility between the current set A, and a new set $B = \{x \cup A\}$, and these two sets could be rated by judges to determine what aspects

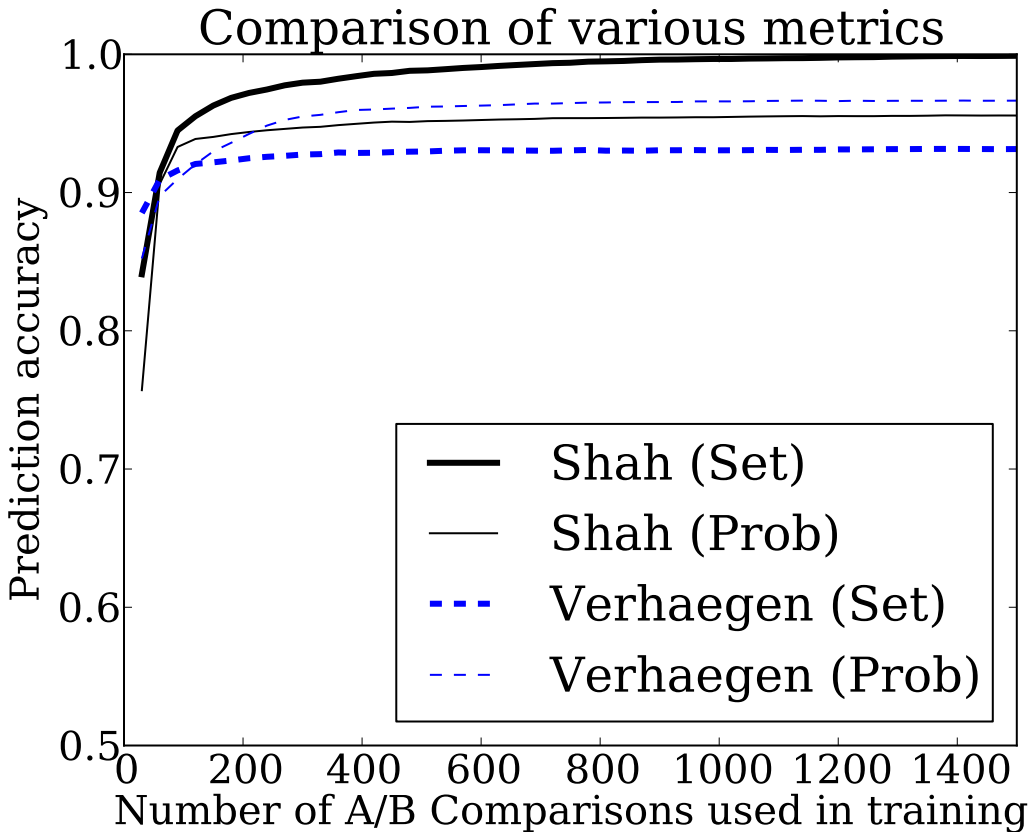


Figure 5.6: Comparing the convergence rates across the two metrics, differences in how each model handles diminishing marginal utility become evident: Shah's metric performs better under under set cover, where Verhaegen's performs better under probabilistic cover. This matches expectations, since Verhaegen's metric attempts to encourage uniformity in the branches—the same goal as maximizing marginal utility. It is also clear that the further the variety model class is from the actual evaluator's model, the more bias there will be in the final accuracy: Shah's metric is replicated exactly, whereas Verhaegen's metric is only approximated by submodular functions.

affect novelty. Coincidently, this also allows an easy way of differentiating H vs P creativity [13]: the former uses a set containing all past designs, and the other uses a set that contains only the designs known to a particular agent.

Unexpectedness, at least as described by Maher [81], can be formulated similarly to novelty above. As with novelty, the marginal utility is used to measure the increase in value for a particular design, however, unlike novelty, unexpectedness only considers some number of the most recently seen designs. This allows the model to essentially forget old designs over time, and become surprised if something new breaks a chain of similar designs.

The diminishing marginal utility property of submodular functions, combined with the binary decision ratings from experts, makes the proposed model a flexible platform for future research in modeling different parts of creativity.

5.4.2 The Utility of Human Evaluation

The basis for the proposed model hinges on collecting sets of human judgements comparing conceptual variety. This raises a natural concern: If the human judgement data are noisy, or even contradictory, will this model be of any use? What if humans are consistently poor judges of a certain aspect of creativity?

In the case of noisy observations, Fig. 5.4 and 5.5 demonstrate this exact effect: increased noise translates to increased convergence time, but does not significantly effect the final accuracy of the results. Even under high levels of noise ($N(0, 5)$ for a 10 point variety metric), the model is able to combine the estimates of multiple ratings to uncover the true underlying variety score. This offers a significant advantage when using human evaluation: even if the raters' assessments of variety are off from each other by a large amount, the model handles the noise gracefully. These experimental results assume that there is an underlying true variety metric, and that experts' assessments are normally distributed around that true score. While that assumption is not completely true in reality, it is a reasonable approximation that offers initial results as to the robustness of the model.

This being said, the proposed model has a natural way of accounting for differences between individual raters, or groups of raters: By extending the score function (Eqn. (5.1)) with a set of user or group-specific bias terms, this model can automatically learn these effects given additional training data. This is a common approach for capturing possible bias terms (*e.g.*, [68]).

If human judgements contradict each other, or if expert judgements are consistently wrong about a set of metrics, then the proposed model would suffer as well—it would be just as useful as the inconsistent experts it is modeling. However, the model can easily provide confidence estimates around its predictions, providing an easy method of judging its coherence (a non-trivial task for a human evaluator).

A relevant question is how to select an appropriate population of human judges. Does one need domain experts or professional designers, or will students or random people do? This question is best answered through appropriate controlled studies, such as the one conducted by Kudrowitz and Wallace [71] who showed that Mechanical Turk raters showed strong correlation with novelty but poor correlation with feasibility. The results of those types of studies directly inform the applicably of this chapter's model.

Lastly, the convergence behavior tells us something about the number of ratings that might be needed: both models, even under exceptionally noisy conditions, reached or exceeded 90% accuracy within 500 samples. Given 10 raters, this amounts to 50 ratings per person—even if actual human evaluation requires collecting an order of magnitude more data, due to addition noise or other factors, this is still well within feasible amounts. Obviously, as the complexity of the model grows to account for additional creative factors the amount of data needed also increases, but initial results appear promising.

5.4.3 Impact on Design Creativity Measurement

The generalizability of the proposed approach opens up many new questions and future work opportunities for those working in design creativity measurement:

What are good ways of encoding concepts in order to detect creativity? The difference in scores between Fig. 5.4 and Fig. 5.5 is one of modeling assumptions: In the case of Shah *et al.* , the proposed model could completely replicate the original metric, and thus was able to achieve 100% accuracy over time. Since Verhaegen *et al.* use the Herfindahl index to penalize lack of variety, the proposed model was not able to achieve 100% accuracy. The Herfindahl index does not behave exactly like a submodular function, though the similarity is close enough that the model still matched Verhaegen's metric to within 96%.

This difference in accuracy provides a means to test the appropriateness of certain design encodings, as well as the magnitude to which certain features have diminishing marginal utility. By using the same set of human judgements, published and new metrics can be compared against one another to assess how closely they match a particular set of human judgements. This is essentially a form of model selection, but applied to creativity metrics.

Are some features more important to creativity than others? In order to use the proposed approach to evaluate how important different design attributes are for creativity, one can do one of two things: 1) compare a large number of different design encodings, determine one that fits human data the best, and then inspect that

model's weights ($\mathbf{w}$) to determine importance, or 2) create a giant design encoding with as many design features as possible and train the model using L1 regularization in Eqn. (5.2) to encourage unimportant weights to be driven to zero. In addition, new computational algorithms can be derived to identify important features not yet known: algorithms that cluster ideas according to marginal utility could be given to domain experts to uncover patterns in human evaluation.

Do different domains, experience levels, or backgrounds judge the same creativity metric differently? By using a particular design encoding and training the model on different groups of people, future work could formalize differences in opinion regarding the same metric. For example, given a Shah-like variety metric, would architects and engineers view the importance of physical function differently? Comparing the learned weights of the metric for each group could provide an answer.

5.5 Summary

This chapter draws a theoretical connection between certain aspects of creativity, such as novelty and variety, and the principle of diminishing marginal utility. By utilizing submodular functions to express diminishing marginal utility, this chapter described a creativity model capable of tying together many existing metrics under a common framework. Not only can this model generalize different configurations of creativity metrics, such as linear, hierarchical, or graph based metrics, but it can adapt to human evaluators from different backgrounds. It does so by requiring only simple A/B comparisons between sets of concepts, a rating task that is easily interpreted by human evaluators, making data collection simple.

As validation, this chapter demonstrated how the proposed model can reliably reproduce several published creativity metrics. Using simulated comparison data, it was able to capture the variety metrics of Shah *et al.* [118] and Verhaegen *et al.* [131] with 100% and 96.4% accuracy, respectively, in the no-noise condition. Under increasingly noisy input conditions, the model is still able to recover the metric accurately, at the cost of some convergence speed.

The use of submodular functions to model diminishing marginal utility carries with it several advantages: 1) the model parameters can be interpreted easily, 2) the likelihood and variety function are convex allowing for efficient optimization, and 3) Computational Design Synthesis systems can use the model to perform optimal set selection in an efficient way.

Rather than claiming to provide *the* universal metric for creativity, this chapter instead acts as a catalyst by which design creativity researchers can ask new questions: How does human evaluation of a particular creativity metric vary across

different conditions (disciplines, countries, professional experience, *etc.*)? What are the important elements that determine creativity? To what extent can one discover those structures given human rating data?

Thus far, the dissertation has looked at overall design communities, how they form, how one can understand design processes within that community, and lastly how to evaluate the ideas that the community produces. The next chapter will briefly summarize these results and address some broader limitations of design communities, such as what to do about the quality of the design input and some of the dangers inherent in using statistical models. Finally, it will point to some future research questions that arise out of this dissertation.

Chapter 6

Conclusion

As design becomes increasing complex, companies are turning to online design communities to help solve their problems. For example, this dissertation opened with a story about DARPA and its FANG challenges—where they used crowdsourcing to design the next generation of armored vehicles. They were not alone: General Electric, Proctor and Gamble, IDEO, and many others leveraged online design communities to bring innovative new products and services to market.

This thesis addressed what problems arise when these communities produce too much data than they can effectively analyze, and how machine learning techniques can solve those problems. It did this through applying machine learning and computational models to problems across different levels of analysis: across the entire community, within a design process, and for a particular set of ideas. This chapter first summarizes the main results of the dissertation and then discusses some broader implications that tie together each chapter. It concludes by pointing to future research directions.

6.1 Dissertation Summary

This dissertation started off by introducing online design communities, in all their shapes and sizes, and discussing why one would use those communities in the first place. It brought up factors such as reduced cost, faster lead times, increased diversity, and truly massive scale beyond what could be accomplished by traditional companies. In doing so, it also highlighted the Achilles' heel in online design communities: they generate more data then they can effectively use. This laid the groundwork for the main question addressed by this dissertation:

> How can online design communities effectively use the design data they generate to help manage their operations and improve their designs?

Answering this question required background in a range of areas, from design to crowdsourcing and online communities to machine learning. Chapter 2 reviewed these areas, and placed this dissertation at their intersection. With respect to Quinn and Bederson's taxonomy of human computation [102], this dissertation sits across areas such as crowdsourcing, social computing, and data mining. It lay separate from the field of Human Computation itself because people within online design communities are not typically performing computational tasks, but rather providing the input for algorithms that might later assist designers. That chapter also discussed different types of machine learning approaches (supervised vs. unsupervised, generative vs. discriminative), and how those approaches have been applied in both design and crowdsourcing domains.

After covering the requisite background, the dissertation began exploring online design communities at the broadest unit of analysis: the community level. Chapter 3 introduced OpenIDEO, a real-world online design community where thousands of people participate in design challenges for social causes. The chapter used techniques from the field of Network Analysis to analyze the information flow and community structure of OpenIDEO, both on a per-challenge basis, and over time. It highlighted how various network attributes of OpenIDEO, such as efficiency, clustering, and assortativity influence idea generation, and what might have driven the growth of those properties. OpenIDEO possesses a unique structure where those who collaborate most frequently do so primarily with those who do not collaborate frequently. This atypical network behavior has many benefits, including network robustness and efficiency for transferring ideas. The chapter also addressed possible incentives that one might use to manage online design communities: encouraging commenting and feedback, using designated community managers, and helping users in the center of the network reach out to those on the periphery of the network.

Looking within design communities, chapter 4 covered the design processes that a community uses and how data-driven techniques could help users understand and improve them. It focused on HCD Connect, an online design community where users uploaded case studies of design problems they faced and described which design methods they used to address each problem. Intended as a learning resource, HCD Connect's large corpus of case studies can be difficult for novice designers to search through or navigate. The chapter demonstrated how techniques from large scale hypothesis testing, unsupervised learning, and recommender systems could ameliorate many of those issues. For example, False Discovery Rate Control algorithms identified which methods were relevant to particular types of problems (*e.g.*, Agriculture or Community-Development). The chapter highlighted the importance keeping track of which methods were used together in each case study, as that information can help automatically group methods (*e.g.*, using Spectral Clustering) or recommend design methods (*e.g.*, using Collaborative Filtering). Leveraging data-driven techniques for

analyzing design processes turned the wealth of design case studies into an advantage, and provided insight that would be unavailable through strictly manual means.

Lastly, the dissertation looked within design processes at how to evaluate ideas themselves. Specifically, chapter 5 addressed how to evaluate the creativity of a set of ideas in a way that is both flexible (*i.e.*, can be used in many domains), as well as scalable (*i.e.*, can be used on tens of thousands of submissions). It addressed this by comparing two classes of design creativity metrics, Model-based and Human Judgement-based, and drew an important connection between many areas of design creativity and the principle of *diminishing marginal utility*. This connection drove the application of a logistic regression model that used a small number of expert ratings to train a repeatable and scalable model-based metric that can efficiently evaluate tens of thousands of ideas. The key technical piece lay in applying a type of mathematical object called a *submodular function* that could account for diminishing marginal utility. This allowed the chapter to generalize several previous model-based metrics under a single umbrella, and created a new theoretical foundation for measuring design creativity.

6.2 Broader Implications

Herbert Simon, in *Sciences of the Artificial* [120], states that

> "Human beings, viewed as behaving systems, are quite simple. The apparent complexity of our behavior over time is largely a reflection of the complexity of the environment in which we find ourselves."

Online design communities are changing that environment, and researchers need methods that can scale to the task. This dissertation demonstrated how inductive models from Machine Learning could provide a means of scientific inquiry by using the data generated by a community in a way that would not be possible using traditional statistical and qualitative research methods. However, this comes with its own limitations: How can one trust the data generated by these online communities? Will not the choice of mathematical model limit the kind of information and interpretations one can draw? This section addresses some of these implications, and then points the way to future research avenues that can address new questions.

Just because someone did it, does it mean its right? Machine learning is fundamentally descriptive in nature—the goal is to predict future events by matching, as best as possible, the reality of the present. In that sense, it is not really suited for prescribing actions far outside what we have already seen. Chapters 3, 4, and 5

all used community generated data as the ground truth: collaborations drove network analysis models, submitted case studies defined method groupings, and binary creativity assessments trained a regression analysis. This all lied on top of inductive reasoning's major assumption: that the input itself is valid.

If design communities themselves might produce nonsense, then will not machine learning methods also be susceptible to this? There are two answers to this, depending on whether one views design research as a descriptive or prescriptive process. For those who view design research as **descriptive**, then, regardless of the community, machine learning methods can be used to introspect on the "behaving systems" that Simon pointed to above. The spectral clustering methods in section 4.4 would locate methods that *are presently used* together rather than the methods that *should be used* together; it makes no prescriptive claim other than to discover how design is performed in practice.

For those who view design research as **prescriptive**, one would need to be careful as to how one sampled the human input within the community. This raises issues such as who controls or validates community membership, or how one can make their machine learning methods robust to noise. Quinn and Bederson [102] review some useful insights from similar issues in the Human Computation literature, such as the aggregation and quality control strategies.

Statistical misspecification and causality. The application of machine learning and statistical techniques is becoming almost ubiquitous across a variety of domains, due to their ability to, almost seemingly by magic, take complex multi-dimensional datasets and deduce interesting hypotheses. However, this magic comes with a price: statistical models cannot claim causality and must only claim predictive accuracy, unless a particularly stringent set of experimental conditions are met [39].

For example, chapter 5 introduced a logistic regression model built around particular submodular functions, the purpose of which was to accurately emulate the creative judgements given by experts. It would be a mistake to train that model and then treat its individual coefficients as though they represented the true importance of different measures of creativity; that would be treating the model as causally valid, rather than as intended—solely for prediction. With that in mind, the studies herein should be paired with both qualitative and properly controlled laboratory experiments in the future to present a fuller picture of how design communities operate.

6.3 Future Research Avenues

This dissertation focused primarily on using learning algorithms to help existing communities deal with the data they had already generated: existing collaborations (chap-

ter 3), existing case studies (chapter 4), and existing ideas (chapter 5). However, there is no reason why similar algorithms could not be used to help create new things: generating new communities or teams, directing online design process mentorship and education, or inspiring designers to come up with new ideas. In that vein, this section highlights some of future research avenues across the three levels considered in this dissertation: community-level, process-level, and idea-level.

6.3.1 Community-Level Research

Chapter 3 demonstrated how social collaboration drives network structure, and how certain structures were advantageous to idea generation. Given this knowledge, one avenue for future research could dynamically adjust the collaboration network in order to optimize for certain network structures. For example, an algorithm might recommend certain ideas over others or request targeted feedback from certain members, in order to achieve better flow of information. This is similar to the fields of Nework Formation and Network Game Theory [65], where one tries to build up a network, piece by piece, to achieve networks with particular properties.

6.3.2 Process-Level Research

Chapter 4 demonstrated how to leverage user-submitted case studies to help understand the design process and provide important information to novice designers, such as which methods complement one another, or which should be used for certain types of problems. It also demonstrated how to use a recommendation system to help recommend methods for designers. A similar type of approach could be used not just for capturing existing design processes, but for encouraging new ones, or performing training.

For example, it would be possible to detect when a particular method was being used out of order, or for an inappropriate problem. An algorithm could recommend something different, to improve the design process. Or it could purposefully recommend methods as part of a training program to help novice designers recognize specific difficulties or practice particular skills. Online design communities offer a great opportunity to explore distributed mentoring and education, and algorithms can play a role in customizing this learning for individuals.

6.3.3 Idea-Level Research

Chapter 5 demonstrated how small collections of expert advice could be used to evaluate thousands of ideas. However, what about creating those ideas in the first place? There are two future research avenues where the online communities and

machine learning algorithms might help: locating inspirations and guiding the process of creation.

In chapter 3, Fig. 3.3(a) showed how ideas could build on each other, acting as future inspirations for new ideas. However, finding these inspirations can be like finding a needle in a haystack; no individual is going to crawl through the thousands of ideas generated by a community to find inspiration. This is precisely where some kind of recommendation system (chapter 4) or variety measure (chapter 5) might be of use. By using relationships between existing concepts, or analyzing their content, an algorithm would be able to collect a diverse, but small, set of ideas to present to a designer as possible inspirations.

Even once a designer has been inspired, not all designers possess the technical skills necessary to execute a design (*e.g.*, experience with a CAD program). Here, intelligent design interfaces might leverage the previous designs of others to guide a novice through simplified interface to produce a limited set of designs. Such tools are already taking shape in computer graphics [104, 126, 124], and similar approaches could be adapted for mechanical design.

This dissertation closes with another quote from Herbert Simon:

> "The proper study of mankind has been said to be man. But I have argued that man—or at least the intellective component of man—may be relatively simple, that most of the complexity of his behavior may be drawn from man's environment, from man's search for good designs. If I have made my case, then we can conclude that, in large part, the proper study of mankind is the science of design, not only as the professional component of a technical education but as a core discipline for every liberally educated person."[119, pg. 82]

In this, he argues that not only is *everyone* a designer, but that studying design provides a window into ourselves. This dissertation has addressed both parts of this. For the former, it described online design communities, a new platform that can empower the designers in all of us to produce something greater than the sum of our parts. In the latter, it described statistical tools from machine learning that permit us to introspect on our actions, to learn from our own collective behaviors, and to improve upon them. Ultimately, this story is not merely about understanding communities, but also about understanding ourselves.

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] Amr Ahmed, Choon H. Teo, S. V. N. Vishwanathan, and Alex Smola. Fair and balanced: learning to present news stories. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 333–342, New York, NY, USA, 2012. ACM.

[3] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure)*. Oxford University Press, later printing edition, August 1977.

[4] Genrich Altshuller, Lev Shulyak, Steven Rodman, and Uri Fedoseev. *40 principles: TRIZ keys to innovation*, volume 1. Technical Innovation Center, Inc., 1998.

[5] Teresa M. Amabile. Social psychology of creativity: A consensual assessment technique. *Journal of Personality and Social Psychology*, 43:997–1013, 1982.

[6] G. Badaro, H. Hajj, W. El-Hajj, and L. Nachman. A hybrid approach with collaborative filtering for recommender systems. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 349–354, July 2013.

[7] A. L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4):590–614, August 2002.

[8] Robert M Bell, Yehuda Koren, and Chris Volinsky. All together now: A perspective on the netflix prize. *Chance*, 23(1):24–29, 2010.

[9] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[10] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[13] M. Boden. Computer Models of Creativity. *AI Magazine*, 30(3):23–34, 2009.

[14] G. Broadbent and A. Ward. *Design methods in architecture*. AA Papers. Lund Humphries, 1969.

[15] Geoffrey Broadbent. The development of design methods. *Design Methods and Theories*, 13(1):41–45, 1979.

[16] David C. Brown. The Curse of Creativity. In John S. Gero, editor, *Design Computing and Cognition '10*, chapter 9, pages 157–170. Springer Netherlands, Dordrecht, 2011.

[17] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[18] Jean-Marie Burkhardt and Todd Lubart. Creativity in the Age of Emerging Technology: Some Issues and Perspectives in 2010. *Creativity and Innovation Management*, 19(2):160–166, 2010.

[19] Matthew I. Campbell, Jonathan Cagan, and Kenneth Kotovsky. A-Design: An Agent-Based approach to conceptual design in a dynamic environment. 11(3):172–192, 1999.

[20] Matthew I. Campbell, Rahul Rai, and Tolga Kurtoglu. A stochastic Tree-Search algorithm for generative grammars. *Journal of Computing and Information Science in Engineering*, 12(3):031006+, 2012.

[21] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[22] Erin A. Carroll, Celine Latulipe, Richard Fung, and Michael Terry. Creativity factor evaluation: towards a standardized survey metric for creativity support. In *Proceeding of the seventh ACM conference on Creativity and cognition*, C&C '09, pages 127–136, New York, NY, USA, 2009. ACM.

[23] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81(2):591–646, May 2009.

[24] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3D modeling. *ACM Trans. Graph.*, 29(6), December 2010.

[25] Sung-Bae Cho. Towards creative evolutionary systems with interactive genetic algorithm. *Appl. Intell.*, 16(2):129–138, 2002.

[26] Henri Christiaans and Kees Venselaar. Creativity in Design Engineering and the Role of Knowledge: Modelling the Expert. *International Journal of Technology and Design Education*, 15(3):217–236, January 2005.

[27] Peter Csermely, Andras London, Ling-Yun Wu, and Brian Uzzi. Structure and dynamics of core-periphery networks, September 2013.

[28] DARPA. Vehicleforge. http://cps-vo.org/group/avm/vehicleforge, 2014.

[29] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[30] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[31] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 Conference*

on Computer Supported Cooperative Work, CSCW '12, pages 1013–1022, New York, NY, USA, 2012. ACM.

[32] Steven Dow and Burr Settles. Modeling online creative collaborations. *XRDS*, 19(4):21–25, June 2013.

[33] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1994.

[34] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, Paris, France, June 2009.

[35] StevenD. Eppinger, DanielE. Whitney, RobertP. Smith, and DavidA. Gebala. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1):1–13, 1994.

[36] Susan Finger and John R. Dixon. A review of research in mechanical engineering design. part i: Descriptive, prescriptive, and computer-based models of design processes. *Research in Engineering Design*, 1:51–67, 1989. 10.1007/BF01580003.

[37] Susan Finger and John R. Dixon. A review of research in mechanical engineering design. part ii: Representations, analysis, and design for the life cycle. *Research in Engineering Design*, 1:121–137, 1989. 10.1007/BF01580205.

[38] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010.

[39] David A. Freedman. *Statistical Models and Causal Inference: A Dialogue with the Social Sciences*. Cambridge University Press, 1 edition, November 2009.

[40] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[41] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, July 2008.

[42] Mark Fuge and Alice Agogino. How online design communities evolve over time: the birth and growth of OpenIDEO. In *ASME International Design Engineering Technical Conferences*, Buffulo, U.S.A., August 2014.

[43] Mark Fuge and Alice Agogino. User research methods for development engineering: A study of method usage with IDEO's HCD Connect. In *ASME International Design Engineering Technical Conferences*, Buffulo, U.S.A., August 2014.

[44] Mark Fuge, Bud Peters, and Alice Agogino. Analysis of collaborative design networks: A case study of OpenIDEO. *Journal of Mechanical Design*, Under Review.

[45] Mark Fuge, Josh Stroud, and Alice Agogino. Automatically inferring metrics for design creativity. In *ASME International Design Engineering Technical Conferences*, Portland, U.S.A., August 2013.

[46] Mark Fuge, Kevin Tee, Alice Agogino, and Nathan Maton. Analysis of collaborative design networks: A case study of OpenIDEO. *Journal of Computing and Information Science in Engineering*, 14(2):021009+, March 2014.

[47] J. Gero. Computational Models of Innovative and Creative Design Processes. *Technological Forecasting and Social Change*, 64(2-3):183–196, June 2000.

[48] John S. Gero. Design prototypes: a knowledge representation schema for design. *AI Mag.*, 11(4):26–36, October 1990.

[49] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. A scalable, accurate hybrid recommender system. *International Workshop on Knowledge Discovery and Data Mining*, 0:94–98, 2010.

[50] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In *Proc. Neural Information Processing Systems (NIPS)*, 2011.

[51] Prem K. Gopalan and David M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013.

[52] Michael Gordon and Praveen Pathak. Finding information on the world wide web: the retrieval effectiveness of search engines. *Information Processing & Management*, 35(2):141 – 180, 1999.

[53] Pierce Gordon, Mark Fuge, and Alice Agogino. Examining design for development online: A qualitative analysis of OpenIDEO using HCD/UCD metrics. In *ASME International Mechanical Engineering Congress and Exposition (ASME IMECE'14)*. ASME, 2014.

[54] Thomas L Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. In *NIPS*, volume 18, pages 475–482, 2005.

[55] Zhenyu Gu, Ming Xi Tang, and John H. Frazer. Capturing aesthetic intention during interactive evolution. *Computer-Aided Design*, 38(3):224–237, March 2006.

[56] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 203–212, New York, NY, USA, 2010. ACM.

[57] Kurtis Heimerl, Brian Gawalt, Kuang Chen, Tapan Parikh, and Björn Hartmann. Communitysourcing: Engaging local crowds to perform expert work via physical kiosks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1539–1548, New York, NY, USA, 2012. ACM.

[58] Helen Hamlyn Centre for Design. Designing with people: Methods. http://designingwithpeople.rca.ac.uk/methods, Accessed November, 2013 2013.

[59] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[60] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[61] S. Hewens. Smartlife is open for business selling pure drinking water. https://www.ideo.org/stories/smartlife-is-open-for-business-selling-pure-drinking-water, February 2013. [Online; accessed January-2014].

[62] John J. Horton, David G. Rand, and Richard J. Zeckhauser. The Online Laboratory: Conducting Experiments in a Real Labor Market. *Social Science Research Network Working Paper Series*, April 2010.

[63] Julie S. Hui, Michael D. Greenberg, and Elizabeth M. Gerber. Understanding the role of community in crowdfunding work. In *Proceedings of the 17th ACM*

Conference on Computer Supported Cooperative Work & Social Computing, CSCW '14, pages 62–74, New York, NY, USA, 2014. ACM.

[64] IDEO.org. Hcd connect: Where optimists take on our word's challenges by sharing stories, questions, and resources. http://www.hcdconnect.org, 2014. [Online; accessed January-2014].

[65] Matthew O. Jackson, Gabrielle Demange, Sanjeev Goyal, and Anne Van Den Nouwel. A survey of models of network formation: stability and efficiency. In *In Group Formation in Economics: Networks, Clubs and Coalitions*, 2003.

[66] Walter Johnson. Meta II: Formal co-verification of correctness of large-scale cyber-physical systems during design. Technical report, Palo Alto Research Center Inc., Palo Alto, CA, March 2012.

[67] John C. Jones. *Design Methods*. Wiley, 2 edition, September 1992.

[68] Yehuda Koren. The BellKor Solution to the Netflix Grand Prize. 2009.

[69] Gueorgi Kossinets and Duncan J. Watts. Origins of homophily in an evolving social network. *American Journal of Sociology*, 115(2):405–450, September 2009.

[70] Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM Trans. Intell. Syst. Technol.*, 2(4), July 2011.

[71] Barry M. Kudrowitz and David Wallace. Assessing the quality of ideas from prolific, early-stage product ideation. *Journal of Engineering Design*, pages 1–20, April 2012.

[72] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Scott R. Klemmer, and Jerry O. Talton. Webzeitgeist: Design Mining the Web. In *Proceedings of the 31st Annual ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[73] Vijay Kumar. *101 Design Methods: A Structured Approach for Driving Innovation in Your Organization*. Wiley, 1 edition, October 2012.

[74] Karim Lakhani, Anne-Laure Fayard, Natalia Levina, and Stephanie Pokrywa. Openideo. *Harvard Business School Technology & Operations Mgt. Unit Case*, (612-066), May 2012.

[75] Zsolt Lattmann, Adam Nagel, Tihamer Levendovszky, Ted Bapty, Sandeep Neema, and Gabor Karsai. Component-based modeling of dynamic systems using heterogeneous composition. In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, MPM '12, pages 73–78, New York, NY, USA, 2012. ACM.

[76] Qize Le. *Analysis and Modeling of the Product Structure and Community Structure in Open Source Processes*. PhD thesis, Washington State University, August 2012.

[77] Qize Le, Zhenghui Sha, and Jitesh H. Panchal. A generative network model of product evolultion. *Journal of Computing and Information Science in Engineering*, October 2013.

[78] Antonia Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: GitHub as a collaborative social network. In *Eigth Internatioanl AAAI Conference on Weblogs and Social Media*. AAAI, 2014.

[79] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, March 2007.

[80] Greg Madey, Vincent Freeh, and Renee Tynan. The Open Source Software development phenomenon: An analysis based on social network theory. In *Proceedings of the Eighth Americas Conference on Information Systems*, pages 1806–1815. 2002.

[81] Mary L. Maher. Evaluating creativity in humans, computers, and collectively intelligent systems. In *Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design*, DESIRE '10, pages 22–28, Lancaster, UK, UK, 2010. Desire Network.

[82] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[83] Victor Margolin and George R. Buchanan. *The Idea of Design*. The MIT Press, February 1996.

[84] Winter Mason and Duncan J. Watts. Collaborative learning in networks. *Proceedings of the National Academy of Sciences*, 109(3):764–769, January 2012.

[85] Jay P. McCormack, Jonathan Cagan, and Craig M. Vogel. Speaking the buick language: capturing, understanding, and exploring brand identity with shape grammars. *Design Studies*, 25(1):1–29, January 2004.

[86] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[87] Brent A. Nelson, Jamal O. Wilson, David Rosen, and Jeannette Yen. Refined metrics for measuring ideation effectiveness. *Design Studies*, 30(6):737–743, November 2009.

[88] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, December 1978.

[89] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126+, February 2003.

[90] Mark Newman. *Networks: An Introduction*. Oxford University Press, March 2010.

[91] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an Algorithm. In *Neural Information Processing Systems*, pages 849–856, 2001.

[92] SarahK. Oman, IremY. Tumer, Kris Wood, and Carolyn Seepersad. A comparison of creativity and innovation metrics and sample validation through in-class design projects. *Research in Engineering Design*, 24:65–92, 2013.

[93] Karen O'Quin and Susan P. Besemer. The development, reliability, and validity of the revised creative product semantic scale. *Creativity Research Journal*, 2(4):267–278, 1989.

[94] Gunay Orbay, Luoting Fu, and Levent Burak Kara. Shape spirit: Deciphering form characteristics and emotional associations through geometric abstraction. In *ASME International Design Engineering Technical Conferences/DTM*, Portland, Oregon, 2013.

[95] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[96] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[97] Jitesh H. Panchal. Co-evolution of products and communities in mass-collaborative product development-a computational exploration. In *International Conference on Engineering Design (ICED'09)*, 2009.

[98] Jitesh H. Panchal. Coordination in collective product innovation. In *ASME International Mechanical Engineering Congress and Exposition (ASME IMECE'10)*. ASME, 2010.

[99] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[100] Emilie Poirson. Eliciting user perceptions using assessment tests based on an interactive genetic algorithm. *Journal of Mechanical Design*, 135(3):031004+, January 2013.

[101] Gerard J. Puccio, John F. Cabra, J. Michael Fox, and Helene Cahen. Creativity on demand: Historical approaches and future trends. *AI EDAM*, 24(Special Issue 02):153–159, 2010.

[102] Alexander J. Quinn and Benjamin B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, New York, NY, USA, 2011. ACM.

[103] Er J. Quinn, Benjamin B. Bederson, Tom Yeh, and Jimmy Lin. Crowdflow: Integrating machine learning with mechanical turk for speed-cost-quality flexibility. 2010.

[104] Yi Ren and Panos Y. Papalambros. A design preference elicitation query as an optimization process. *Journal of Mechanical Design*, 133(11):111004+, 2011.

[105] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[106] M. Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Core-Periphery structure in networks, April 2013.

[107] Celeste Roschuni. The DesignExchange: Interactive portal for the design community of practice, 2013.

[108] Celeste Roschuni, Alice Agogino, and Sara Beckman. The DesignExchange: Supporting the design community of practice. In *International Conference on Engineering Design, ICED'11*, Denmark, August 2011.

[109] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887, New York, NY, USA, 2008. ACM.

[110] Nilesh Saraf, Andrew Seary, Deepa Chandrasekaran, and Peter Monge. Evolution of an open source community network an exploratory study. *Social Science Research Network Working Paper Series*, February 2012.

[111] Prabir Sarkar and Amaresh Chakrabarti. Assessing design creativity. *Design Studies*, March 2011.

[112] Somwrita Sarkar and Andy Dong. Community detection in graphs using singular value decomposition. *Physical Review E*, 83(4):046114+, April 2011.

[113] Rob Saunders and John S. Gero. How to Study Artificial Creativity. In *Proceedings of the 4th conference on Creativity and cognition*, pages 80–87. ACM Press, 2002.

[114] Nitin Sawhney, Saul Griffith, Yael Maguire, and Timothy Prestero. Thinkcycle: sharing distributed design knowledge for open collaborative design. *International Journal of Technologies for the Advancement of Knowledge and Learning (TechKnowLogia)*, 4(1):49–53, 2002.

[115] John Scott. Social network analysis: developments, advances, and prospects. *Social Network Analysis and Mining*, 1(1):21–26, January 2011.

[116] Burr Settles and Steven Dow. Let's get together: the formation and success of online creative collaborations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2009–2018, New York, NY, USA, 2013. ACM.

[117] Zhenghui Sha and Jitesh Panchal. Estimating the node-level behaviors in complex networks from structural datasets. In *2013 ASME Computers and Information in Engineering (CIE) Conference*. ASME, 2013.

[118] Jami J. Shah, Steve M. Smith, and Noe Vargas-Hernandez. Metrics for measuring ideation effectiveness. *Design Studies*, 24(2):111–134, March 2003.

[119] Herbert A. Simon. The science of design: Creating the artificial. *Design Issues*, 4(1/2):pp. 67–82, 1988.

[120] Herbert Alexander Simon. *The Sciences of the Artificial*. MIT press, 1996.

[121] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8. IEEE, June 2008.

[122] Ramesh Srivathsavai, Nicole Genco, Katja Holtta-Otto, and Carolyn C. Seepersad. Study of existing metrics used in measurement of ideation effectiveness. In *Volume 5: 22nd International Conference on Design Theory and Methodology; Special Conference on Mechanical Vibration and Noise*, pages 355–366. ASME, 2010.

[123] Andrew T. Stephen, Peter P. Zubcsek, and Jacob Goldenberg. Product Ideation in Social Networks. *Social Science Research Network Working Paper Series*, March 2013.

[124] Jerry Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graphics*, 30(2), April 2011.

[125] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah D. Goodman, and Radomír Měch. In *Learning Design Patterns with Bayesian Grammar Induction*, 2012.

[126] Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. Exploratory modeling with collaborative design spaces. In *Proceedings of the 2nd Annual ACM SIGGRAPH Conference and Exhibition in Asia*, New York, NY, USA, 2009. ACM Press.

[127] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam T. Kalai. Adaptively Learning the Crowd Kernel, June 2011.

[128] Olivier Teboul, Iasonas Kokkinos, Loïc Simon, Panagiotis Koutsorakis, and Nikos Paragios. Shape grammar parsing via reinforcement learning. In *IEEE Conference on Computer Vision and Patter Recognition*, 2011.

[129] Christian Terwiesch and Karl Ulrich. *Innovation Tournaments: Creating and Selecting Exceptional Opportunities*. Harvard Business Review Press, 5.2.2009 edition, May 2009.

[130] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The Anatomy of the Facebook Social Graph, November 2011.

[131] Paul-Armand Verhaegen, Dennis Vandevenne, Jef Peeters, and Joost R. Duflou. Refinements to the variety metric for idea evaluation. *Design Studies*, September 2012.

[132] Wen Wu, Liang He, and Jing Yang. Evaluating recommender systems. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pages 56–61, Aug 2012.

[133] Anbang Xu, Shih-Wen Huang, and Brian Bailey. Voyant: Generating structured feedback on visual designs using a crowd of non-experts. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1433–1444, New York, NY, USA, 2014. ACM.

[134] Jin Xu, Yongqin Gao, S. Christley, and G. Madey. A Topological Analysis of the Open Souce Software Development Community. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, volume 7, page 198a, Los Alamitos, CA, USA, January 2005. IEEE.

[135] Zhang Yujie and Wang Licai. Some challenges for context-aware recommender systems. In *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pages 362–365, Aug 2010.

[136] Shuo Zhang, Yang Hu, Xiaodong Zhang, and Yingzi Li. Simulation study on collaborative behaviors in mass collaborative product development. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pages 375–379. IEEE, June 2013.

Index