From Software Engineering to Information Design

Yvette Shen, The Ohio State University, Columbus, Ohio, USA
shen.1049@osu.edu

Abstract

Most academic methodologies are developed from a prescribed methodological process that is limited to a specific area of study. However, the disciplinary landscape in which the knowledge is established is being rapidly reconfigured. Given the vast varieties of practices and knowledge base required from information designers, it is even more crucial for them to look outside of the traditional visual design fields and seek diversities for better research and creation methods.

The two disciplines, software engineering and information design, are often perceived as one provides technical solutions to the other. This essay intends to move beyond the common perception, and identify relevant issues in software engineering design that resonate with the information design process. The issues include the multi-component planning approach; the human-oriented agile method; design concepts such as abstraction, decomposition, component modularity, hierarchical relationship, and extensibility. The perspectives from software engineering design and information design is examined through units of analysis, terminology explanations, and forms of communications. The collective design methods and principles provide a systematic framework to the methodological thinking in information design. The discussion serves the purpose of encouraging more conceptual-based conversations between information design and other disciplines, especially in the fields of science and technology.
Keywords: information design process; information design principles; software engineering design; cross-disciplinary design methodology; design and technology

Information is a very broad term. It can be data, knowledge, instruction, location, fact, story, sensory experiences, etc. Information design has always been practiced and systemized by different areas of professions: science, journalism, business, geography, humanity, engineering, architecture, and design. Finding a single comprehensive definition for the term "information design" is almost impossible. While different practitioners may have distinct interests for the purpose of information design, their core objectives and concerns are similar: to present information in a clear and effective manner. Making attractive and accurate information design requires more than a computer drafting software or cut-and-paste template. The aesthetic concerns are as important as how the narrative is clearly and efficiently portrayed. Successful information design creates visual solutions that help to define, explain, or order contents, simplify complexity, relate the seemly unrelated, discover what has been hidden, and create insights (Cairo, 2013).

Software engineering is the development of software in a systematic method. The design process helps the software engineers turn specifications into a working software system. It usually includes architectural views, as well as detailed component and algorithm implementation issues (Laplante, 2007). The design aspects in software engineering often reflect the goals the software is trying to achieve. A well-designed and implemented software system requires creativity, accuracy, and a human-centered approach.

The two disciplines – software engineering and information design – are already highly related to each other as one often supplies technical solutions to the other. Software engineers are often responsible for bringing visual designs to life. A successful information design is where user needs and technical feasibility overlap to create efficient and effective communication. Information content is managed, analyzed, and filtered by software programs. The visual elements being used to depict the information are also created, processed, and arranged by software programs.

Most academic methodologies are developed from a prescribed methodological process that is limited to a specific area of study. The disciplinary landscape in which knowledge is established is being rapidly reconfigured, however. Given the vast varieties of practices and knowledge-bases required for information designers, it is even more crucial for them to look outside of the traditional visual design fields to seek diversity that will lead to more effective research and creation methods.

Literature Review

Design is often referred to as "problem solving". A well-known creative strategy for problem solving in design practice and beyond is *design thinking* – an iterative process of empathizing, defining, ideation, prototyping, and testing. It takes a human-centered approach to bridge the designers' toolkit and the target audiences and users' needs (Brown). While virtually all forms of design are human-centered, information design focuses on the accurate understanding of a specific knowledge set and the unique needs of the end user receiving clear and accessible communication with complex problems. Complex problems usually are made up of many interrelated elements (Hansen, 1999, p.193). They are situated in specific contexts and their parts interact continuously at many levels and across shifting boundaries. In order to convey or even resolve the problem with visual forms, it takes a flexible process that can help reveal relationships among parts.

Many information designers follow an established design method and creative process that usually associates with other design tracks, such as the *Goal-Directed Design* approach developed by Cooper consultancy for human-centered products and service designs that focus on achieving a series of goals. The *Goal-Directed Design* process involves project planning, research, modeling, requirements definition, framework definition, detailed design, and implementation support (Goodwin, 2009). Another design method- *Atomic design*- is a rising design methodology in the field of interface system design that advocates for creating design systems with the big picture in mind. This method breaks down user interfaces (responsive in most cases) into their basic components so that they can be reused throughout the system. It is composed of five distinct stages: atoms, molecules organism, templates, and pages (Frost, 2016).

Although information design shares many foundational basics with other design areas, it has its own unique characteristics. It is a broad and exploratory discipline that encourages research and analytical thinking. It requires a galaxy of disparate tactics being bound together to create successful information solutions. Given the complicated information problems that people will face increasingly in the future, the need for information design could not be more profound. The sheer volume of information and the increasing cost of time demand a more intuitive framework to ensure effective communication. When modern design becomes increasingly relevant to technology, how to employ graphic language to analyze bits of information from particular points of view also demands diverged research methods and processes.

Information design rests on a large range of research foundations: psychology, communication studies, sociology, linguistics, statistics, computer science, cognition science, and various design focus areas (Horn, 199, p. 22). Digital technology has always had inextricable connections with visual communication. The exponential growth in one area leads to increased production of the other. Historically, designers have responded to technological changes by updating their tools and output media. Since information design demands a very analytical mindset to understand and present data and information, it seems natural to make a conceptual connection between information design and disciplines in science and technology.

There has been a constant debate about whether designers need to learn programming in today's technology-driven world (Stinson, 2017). Although software development increasingly infuses how design is planned, produced and consumed, designing and programming are often perceived as two areas pointing to opposite directions. However, when one investigates closely, the two fields of study may share more in common than what is presumed. In the field of software engineering, the word "design" often occurs. Software *design* means the process of transforming user requirements into some suitable form. It is the process of implementing software solutions to *solve problems*. Software design often relies on suitable *design patterns*. There are *design models* and *design principles* that should be followed during the process (Pressman, 2014). There is a definite parallel between the perspectives from design and software engineering. Software engineering seems to have an even stronger connection with information design because both require precision, innovation, creativity, organization, and rigorous logic.
The relevance between the two areas of study may be beyond the simple perception of one being the "front-end" and the other as the "back-end". The following sections will examine the detailed design process, design principles, and some key design concepts from software engineering, and attempt to apply the framework to the methodological thinking in information design.

Research Methods

Any discipline has an epistemic dimension and a social dimension. The epistemic side uses specific analytic tools, concepts, and methods. From the social side, a discipline also entails related "subsets" that practice its techniques. Interdisciplinary work builds on disciplinary perspectives as described above and combines them to achieve a result that is unobtainable through a single disciplinary means (Miller, 2004). Thus, the disciplines are not simply juxtaposed, but deeply intertwined. Findings in one domain illuminate the other. In what follows, the perspectives from software engineering design and information design will be examined through units of analysis, terminology explanations, and forms of communications.

Some concepts and methodologies may be more obviously identified as mutually shared than others. However, a different viewpoint can always benefit the development of one discipline, especially a field like information design that has its roots in a variety of areas. The cross-disciplinary efforts also rely on recognizing the differences and exploring new potentials for how different areas of study can sustain each other.

Design Process

In software engineering, the design process starts from defining the problem and moves toward the solution of the problem. The process usually requires iterations between macro- and micro- level perspectives (Schmidt, 2003).

The macroscopic steps involve decomposing the required specification into various components; identifying various components and how components interact with each other; as well as a more detailed description of the components that allow them to be constructed.

The multi-component approach to product design and development has evolved since the industrial revolution. The approach can be naturally tailored for an information design planning process (consider Ikea furniture and their corresponding assembly manuals). Information design relies on using visual vocabulary to convey complex sets of facts or data. An effective strategy to analyze a complex problem is to break down big problems into smaller pieces that can be addressed more easily. The smaller pieces are not independent entities, they relate to each other in obvious or not-so-obvious ways. Therefore, the correlation of different components is as important as the contents of the individual piece. This multi-component approach is one of the key foundations in software engineering design, and the concept is also reflected in software design principles.

The microscopic level in software engineering design is an iterative process that involves navigating through multi-level decision points based on the specifications, hierarchy, comprehension, and implementation. The Agile Process, a methodology that was first coined in 2001, has been a growing movement in the software development process that is human-oriented and adaptive rather than a traditional process-oriented and sequential approach. The general idea of Agile means to divide time into iterative and incremental cycles known as "sprints" to address a process need (Figure 1). The method stresses customer satisfaction (early and continuous customer and stakeholder involvement). It advocates adaptive planning (sustainable design that can quickly response to changes and continuous development), and it utilizes prototyping during the design process by present working software to clients during the design milestones rather than comprehensive documentation. It also promotes the creation of a collaborative work environment such as *co-location* and *pair programming* (James, 2015).

Test/Review

Test/Review

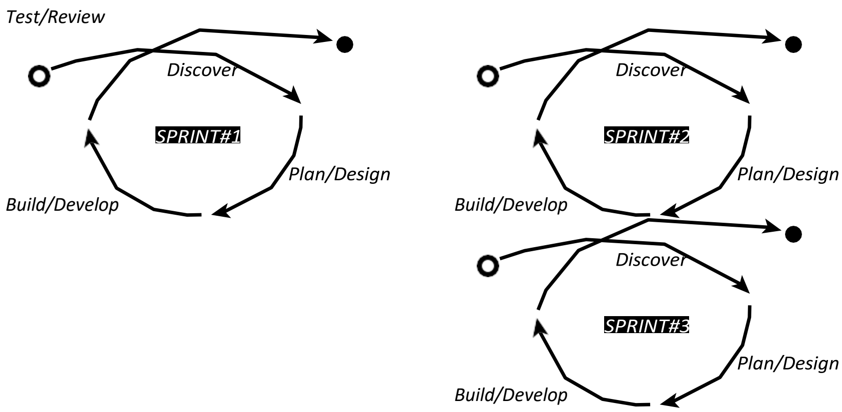

Figure 1: Agile process in a series of “Sprints”.

The agile methodology has already been widely adapted into the interactive design and user experience design fields. When elements of novelty are involved, such as evolving requirements, changing priorities, and flexible deliverables, the Agile approach seems like a natural fit. The cross-functional team environment also facilitates transparency and allows issues to be identified early. Some essential design principles in software engineering-- many derived from the agile philosophies-- can be valuable for mapping the features, structure, and information flows of an information design system.

Design Principles

Some of the key design principles in software engineering include: abstraction, decomposition, component modularity, hierarchical relationship, and extensibility. The following sections will explain how these principles may as well be applied to the building blocks of information design.

Abstraction

In software engineering design *abstraction* manages complexity by emphasizing essential characteristics and suppressing implementation details. It allows postponement of certain design decisions that occur at various levels of analysis.
In information design, a designer can be asked to be a narrator or an editor as well as a visual designer. The act of clarification and determination goes beyond the scope of making aesthetic choices. The cycle of turning data into useful information is a crafty process. Different methods of organizing can create different vantage points, and how designers approach data, understand, and perceive information affects how the information is communicated to audiences.
Although a wide range of sources is essential for initial information gathering and analysis, focusing on too many details at the earlier stage of design will create unnecessary distractions for information filtering. The selection of certain information will always carry a certain bias. The designer’s perception will also be reflected on the design decisions. Therefore, how information is selected and organized has a broad impact on what will ultimately be communicated.

Since visual decisions are better made when the analysis of the problem becomes thorough, the principle of *abstraction* can be an effective strategy to filter and organize information. It can be applied to judge whether the collected information is relevant, useful, and explicit to the content, visual form and composition. The principle suggests starting with the simplified solution – the essential characteristics of complex information and ignore the remainders. Then progressively move back-and-forth between different levels of abstraction from individual modules to the entire system. It is from this point that patterns may be observed and new leads for further investigation arise (Figure 2). The goal is not only to determine what information is necessary or where it should be, but to establish an underlying design logic. A well-defined problem is the foundation of successful visual communication.

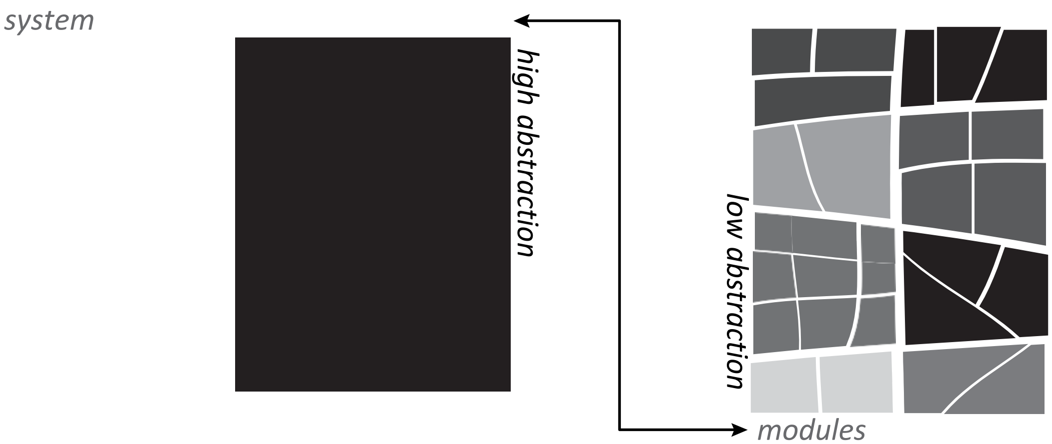

Figure 2: High abstraction for the broad view of the problem.

Decomposition

Decomposition uses the "divide and conquer" concept to dissect large problems into smaller problems. The components within the large piece is determined using a design paradigm, such as functional, structured, object-oriented, etc. The descriptions concern not only the individual component, but also the components' interactions.

Most information design problems can be viewed as an entangled network that entails cause and effect, facts and comparisons, data and analysis. The essence of visually presenting the system in a clear and effective way lies in a shift of mind: being able to see interrelationships rather than the linear chains, and to recognize processes of change rather than snapshots (Hansen, 1999, p. 200). To dissect a complex problem into smaller building blocks can help designers better understand the overall objective of the information conveyed.

Decomposition can be helpful for information designers to understand the problem as well as to find the right visual form to depict the problem. One of the biggest advantages of using visual representations over text is pattern detection. Designers can first look for visual components that already exist for the functionality they wish to achieve, because familiarity can benefit the perception and understanding of information. For example, a timeline is a proven visual form to show the passage of time – a series of steps reflecting change. However, sometimes simply using a line along which to place marks is not adequate to depict the given information. Some other visual forms, many derived from a basic timeline such as Gantt Chart or Swim Lanes, allow more layers of information. For example, the visual form of a Gantt Chart can be deconstructed into three components: time, tasks, and how different tasks relate to each other and within the same given time period (Figure 3).

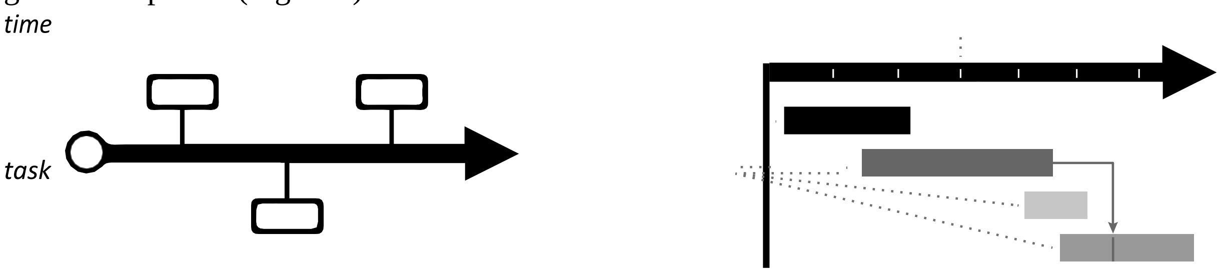

Figure 3: From Timeline to Gantt Chart

One of the most important responsibilities from an information designer is to help audiences identify and detect the interrelationship between pieces of visual information. When individual graphic components are perceived as an intertwined union, viewers' minds can sense the underlying order within apparent chaos. Sometimes audiences may not look for what they see, but they can notice the repetition and perceive a pattern emerging.

Component modularity

A *modular system* in software engineering is one that's structured into identifiable abstractions called *components*. A component can be a software entity that represents an abstraction, or it can be a unit of code that encapsulates data or has identifiable boundaries. It is self-contained and provides a distinct functionality (Pressman, 2014). A component-based application is assembled by connecting the components together. Cohesion is one of the key concepts that are stressed in a component-based software system. It is the degree to which the components belong together within a module. It aims to group together related functionality that contributes to a single lucid task. The goal for a software engineer is to achieve high cohesion so that each component has a well-defined yet distinctive function and the system performs better when the components have a stronger relationship to one another.

The process of information design is a non-linear and recurrent matrix system. The cycle that involves defining, research, analyzing, visualizing, and evaluation are iterative, but also flexible and connected. If the concept of component modularity and cohesion were adopted in relation to the information design process, then the modules can be categorized as semantic or syntactic. The components within these two types of modules will be a number of variables that need to be addressed in the network of concerns. The semantic components include, but are not limited to, objective, content, context, audience, and reference. The syntactic components may include line, shape, form, color, texture, space, type, and medium. The connection between the variables forms a systematic approach that results in a framework of the semantic components to be expressed through a particular set of syntactic components (Figure 4). The individual component has its own purpose, yet they rely on each other to function as a unit. A modification of one component may affect the rest. For example, to visually explain the sales figures by month, a designer can place a simple bar graph on a time-quantity-based coordinate system. The information of each month's sales amount as well as their comparisons can also be well represented. But if a designer fine-tunes the diagram by adding another syntactic component to it, such as applying contrasting colors to the tallest and the shortest bars, then the visual diagram represents a different objective: to identify the best and worst sales month (Figure 5).

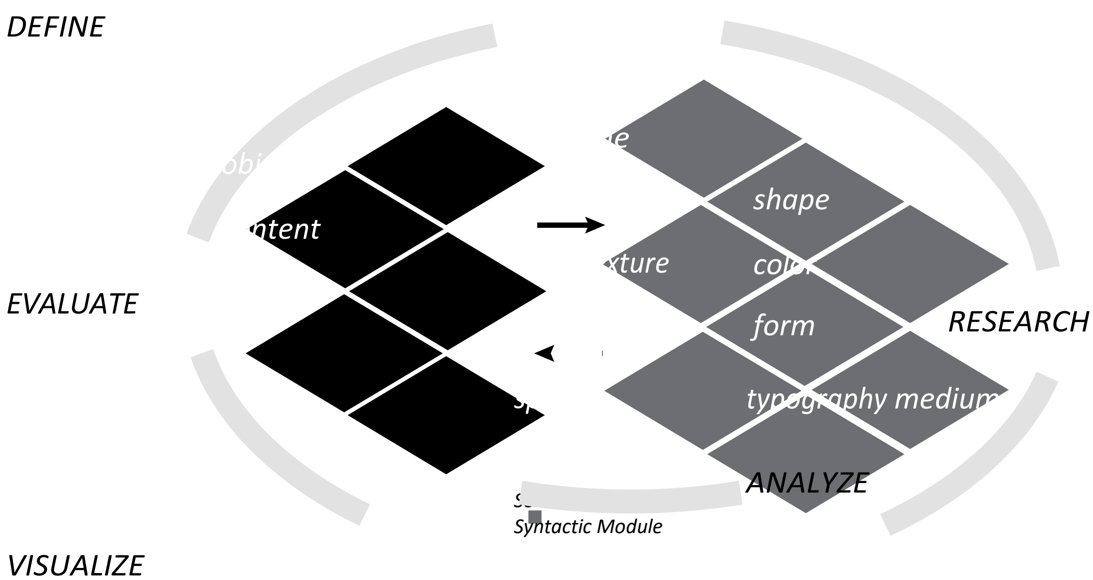

Figure 4: Semantic components and syntactic components in information design.

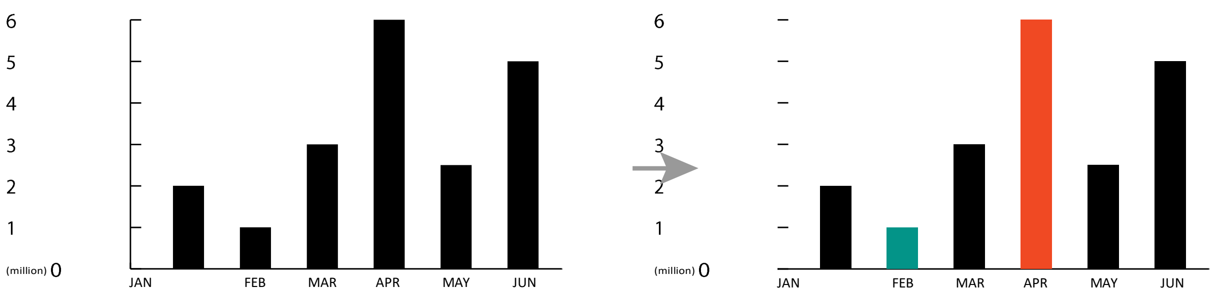

Figure 5: Changing one syntactic component affects the syntactic module.

One benefit of a component-based approach in software design is its reusability. The idea of "reusable" components can be applied to design knowledge rather than the reuse of artifacts produced in prior design efforts. Then a strategy pattern will be the record of how a particular recurring problem was solved in the past. The design strategy is general enough to be adapted and reused in a way that matches a similar situation. The best visual solution can be found by recognizing and recording different levels of components that are proven to be successful.

Hierarchical Relationship

Hierarchy organizes information by measure (scale of size, scale of weight, etc.) or by perceived importance (rank, stature, etc.). Being able to identify the hierarchical relationship depends on a clear understanding of context. A class diagram in the Unified Modeling Language (UML) is usually used to describe the structure of the software system. It serves as a transition from "what" the system shall do to "how" the system will do it. It resembles a flowchart in which the classes and their relationships are identified. The relationship that defines the hierarchy including "uses", "is-composed-of", "is-a", and "has-a" (Schmidt, 2003).

The systemization of the content transforms information into information design. Just like software engineering design, it is imperative to examine hierarchical relationship throughout the process. All information needs to be understood in relation to itself within a framework of comparisons and in a wider context. Richard Saul Wurman, a professor of architecture, suggests five ways to organize everything – location, alphabet, time, category, and hierarchy – also known as the LATCH model (Shedroff, 1999, p. 275). Besides information hierarchy, the concept of "hierarchical relationship" may also be adopted to visual hierarchy – using aesthetics to implement the order of information. The two are closely related. The visual hierarchy not only depends on information hierarchy, but the former is also constrained by the latter.

How should you identify hierarchy in information design? A writing organizing structure used by journalists has been suggested to apply to information design contents. The framework is called "*the inverted pyramid*"- a journalistic writing style that places the most important information at the beginning of the story, and supporting details are included in order of importance (Figure 6).

The most important information is a lead that usually answers questions about who, what, when, where, why, and how. This should be followed by supportive information, and concluded with technical details. This structure allows readers to gain a quick yet solid understanding of the most relevant facts (O'Grady, 2008, p.84). Readers' consumption of an information design project usually involves two steps: first, a quick overview; and second, a more detailed look at the content. During the first step, the reader searches for the core concept or message. Based on the result, he or she will decide whether to take the second step. Therefore, strong contrasting visual elements such as color, size or spatial arrangement are fitting to catch readers' attention from the beginning. As readers proceed through levels of information, they process how the content supports the main message. Then levels of hierarchy can be reflected in the typography, alignment, or color-coding, while maintaining a rhythmic visual consistency (Figure 7).

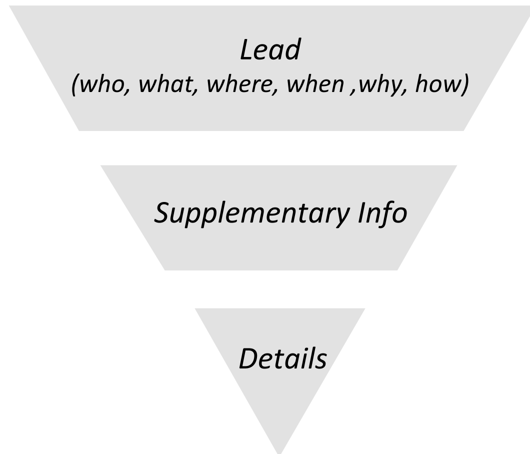

Figure 6: The Inverted Pyramid – a Journalism Writing Framework.

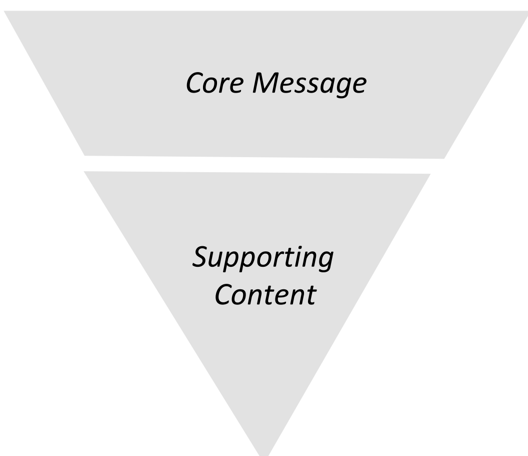

Figure 7: Inverted Pyramid in Information Design.

Extensibility

Extensibility is a software system's ability to take future growth into consideration. When new functionalities or requirements are extended, the existing structure and data flow are minimally or not affected. An extensible design in software engineering means the design is open-ended, and it solves a class of problems rather than a single instance.
The constant evolution of new ideas and new technology demand more flexible designs.

If the design project is meant to be consumed via various media outlets, then the visual and performance quality shall not be affected by technological constrains. If the design project is integrated with a dynamic data system, then extensibility also becomes an important aspect of planning. The visual framework shall demonstrate the same capacity and behavior regardless of how the data progresses. Besides technology–related concerns, the idea of extensibility can also be adopted in relation to the conceptual development during the design process.
Some may identify the objective of information design as to design a product or visual representation that delivers information. The objective, however, should be to achieve a more general design goal (Passini, 1999). Receiving information is only the beginning. What users or viewers can do or gain with the given information is more important. The objective of designing

a way-finding system for an airport is not finding the perfect icon and type, but to assist people to move around quickly and effortlessly (and accurately). Therefore, extensibility is also about seeking sustainable design solutions. When new variables or components need to be added to an established information system, the design can still satisfy its broad-view goal without major structural changes.

Conclusion

This essay identifies some issues in software engineering design that resonate with information design. The issues include the multi-component planning approach, the human-oriented agile method, and software design principles such as abstraction, decomposition, component modularity, hierarchical relationship, and extensibility.

The *multi-component planning* approach suggests breaking down big and complex information problems into interrelated pieces. Designers can build a mental model of the smaller components so the connections between the pieces can be perceived from the big picture. The *agile method* emphasizes the team work environment, adaptive planning, and utilizing prototyping during various stages of the design process. The principle of *Abstraction* helps information designers to evaluate the relevance and usefulness of the collected information.

Decomposition can be a strategy for information designers to create the appropriate visual forms by breaking down the required functionality and building upon existing visual representations. Inspired by the *component modularity* principle, a process model that is composed of a semantic module and syntactic module is proposed. The dynamic relationship between the components inside the two modules constructs the framework that connects the information and the corresponding visual forms. Establishing a logical *hierarchical relationship* shall be applied to both information contents and related visual elements. The value of *extensibility* lies in the functionality and sustainability of the design, as well as helping designers keep the design goal in their minds – how the design will inform and help people.

Discussion

Interdisciplinary collaborations are valued now more than ever. Assessing the leverage afforded by a different perspective demands a meta-disciplinary form of awareness (Miller, 2004). Considering and understanding a different disciplinary perspective can raise questions and propose a hypothesis that has been missing. Some perspective may provide explanatory leverage, while others may offer pragmatic leverage. This essay intends to encourage more conceptual-based conversations between information design and other disciplines, especially in the field of science and technology.

Design is a fundamentally creative activity. Once the barriers of communication are overcome, it may become apparent that many concepts and ideas in design-related domains share more similarities than differences. Even though one design system is built with visual elements and the other is built with machine languages, design procedures and the principles have more to share than the traditional “dividing-up-tasks” approach. There is a great potential for synthesizing knowledge and thinking creatively across perspectives. A systematic integration of different design-related disciplines may create new theories, methods, and concepts that can dissolve borders between disciplinary domains.

References

Beck, K., & Andres, C. (2015). *Extreme programming explained: Second edition, embrace change*. Boston: Addison-Wesley.

Brown, T. (n.d.). Design Thinking. Retrieved March 17, 2017, from http://www.ideou.com/pages/design-thinking

Buschmann, F., Kircher, M., & Schmidt, D. C. (2008). *Pattern oriented software architecture*. Chichester: Wiley.

Cairo, A. (2013). *The functional art an introduction to information graphics and visualization*. Berkeley: New Riders.

Frost, B. (n.d.). Atomic Design Methodology. Retrieved March 17, 2017, from http://atomicdesign.bradfrost.com/chapter-2/

Goodwin, K. (2009). *Designing for the digital age: how to create human-centered products and services*. Indianapolis: Wiley Publishing.

Hansen, Y. (1999). Visualization for Thinking, Planning, and Problem Solving In *Information Design* (pp. 193-220). MIT Press.

Horn, R. E. (1999). Information Design: An Old Hag in Fashionable Clothes? In *Information Design* (pp. 15-34). MIT Press.

IEEE Standard Glossary of Software Engineering Terminology. (n.d.). doi:10.1109/ieeestd.1990.101064

James, M. (2015). Scrum Methodology. Retrieved August 24, 2017, from http://scrummethodology.com/

Laplante, P. (2007). What Every Engineer Should Know about Software Engineering. *What Every Engineer Should Know*. doi:10.1201/9781420006742

Manifesto for Agile Software Development. (2001). Retrieved March 17, 2017, from http://agilemanifesto.org/

Miller, M., & Boix-Mansilla, V. (2004). Thinking Across Perspectives and Disciplines. *GoodWork Project Report Series,27*.

O'Grady, J. V., & O'Grady, K. V. (2008). *The information design handbook*. Cincinnati, OH: How Books.

Passini, R. (1999). Sign-Posting Information Design in *Information Design* (pp. 83-98). MIT Press.

Pressman, R. (2014). *Software engineering*. New York: Mcgraw-Hill.

Schmidt, D. (2003) Software Design Principles and Guidelines Design | FlipHTML5. Retrieved March 17, 2017, from http://fliphtml5.com/bqjb/mqoa

Shedroff, N. (1999). Information Interaction Design: A Unified Field Theory of Design in *Information Design* (pp. 267-292). MIT Press.

Stinson, L. (2017, March 15). John Maeda: If You Want to Survive in Design, You Better Learn to Code. Retrieved March 17, 2017, from https://www.wired.com/2017/03/john-maeda-want-survive-design-better-learn-code

Author Biography

Yvette Shen

Yvette Shen is a visual communication designer and educator. She currently works as an Assistant Professor of Visual Communication Design at the Department of Design, The Ohio State University. She worked as a professional web, interactive, and print designer before she joined the academia. Her current research lies in the domain of information and data visualization, reflecting both the development of her own creative work and the evolvement of visual communication design pedagogical methods. Her creative works and research essays have been recognized and awarded by various design organizations and publications. Her work has also been exhibited worldwide.