Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2012
August 12-15, 2012, Chicago, Illinois, USA

DETC2012-70325

DRAFT: THE STRUCTURE OF CREATIVE DESIGN: WHAT PROBLEM MAPS CAN TELL US ABOUT PROBLEM FORMULATION AND CREATIVE DESIGNERS

Andreea Danielescu
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
Tempe, AZ, 85821

Mahmoud Dinar
Design Automation Lab
Mechanical & Aerospace Engineering
Arizona State University
Tempe, AZ, 85287

Christopher MacLellan
Computer Science and Engineering
Arizona State University
Tempe, AZ, 85281

Jami J. Shah
Design Automation Lab
Mechanical & Aerospace Engineering
Arizona State University
Tempe, AZ, 85287

Pat Langley
Computer Science and Engineering
Arizona State University
Tempe, AZ, 85281

ABSTRACT

Problem formulation is an important part of the design process that has been largely underexplored. Similarly, the relationship between how experts formulate problems, and creative outcome is not well understood. To shed light on what the process of problem formulation can tell us about creativity in design, we use the problem map model – a flexible, domain-independent ontology for modeling the design formulation process – to analyze protocols from eight expert designers. In this paper, we discuss the effectiveness of using problem maps for coding design protocols and what the problem map model can tell us about the creativity of designers. In this exploratory study, we use the problem map model to code and analyze the problem formulation stage of the design process. Ultimately, our aim is to use the patterns identified in this study as a basis for developing a tool to foster designer creativity.

INTRODUCTION

Problem formulation allows the designer to define the problem space and affects how the solution space will be explored. While there has been considerable research on the design process, little research has been conducted to understand the design space or how the designer constructs the problem space to begin with. Additionally, how a designer constructs and defines the problem space may have strong implications for the creativity of the design itself. In this exploratory study, we aim to understand the problem formulation stage of the design process using the problem map ontology. Furthermore, we investigate whether using the problem map ontology [1] -- a flexible, domain-independent ontology for modeling the design formulation process -- to encode and analyze protocol data can provide us with information about the designer's creativity levels and the process by which different designers approach problem formulation.

While protocol studies have been conducted on engineering design [2-4], few have used a pre-defined analysis method. One exception is the study conducted by Pourmohamadi and Gero [5]. Instead, most studies allow their categories to emerge during the coding and analysis process [6,7]. In this paper, we claim that the problem map framework can be used as a predefined method of encoding existing protocol data. Additionally, we show how using the problem map framework can be used to analyze the data and how we can further attempt to identify the role of creativity in the problem formulation process.

In the remaining sections of the paper, we begin by introducing the problem map model. We then discuss related work, including other methods of protocol analysis that have been used to study problem formulation and design.

Afterwards, we discuss our protocol analysis method using the problem map ontology and our results. Finally, we discuss how we may further apply the problem map model and what changes may be necessary to the ontology based on what we learned from our data.

THE PROBLEM MAP MODEL

The problem map model [1] is a flexible and domain independent ontology designed as a tool for understanding problem formulation. It consists of five groups of entities: requirements, functions, artifacts, behaviors and issues. These entities are composed into hierarchies that support multiple disjunctive decompositions. For example, an automobile will have and engine and a transmission, and the engine may run on either gas or electricity.

Requirements are the entities that describe the specifications of the design problem. These may either be hard requirements, which are either fulfilled or not, or goals, which may be satisfied to varying degrees.

Functions contain the activities that the design will execute at some point (e.g. rupture disk, carry passenger, amplify torque). Functions are realized by artifacts, and are motivated by the requirements.

Artifacts realize functions and are the entities that describe the physical components of the design or the concepts the design may be using.

Behaviors are the physical properties and laws that the designer is using. These entities include equations and physical effects, as well as the parameters that are relevant to both artifacts and functions.

Issues are entities that describe the problems associated with other entities in the design formulation. For example, an issue would describe whether a chosen component would violate one of the requirements.

In order to represent hierarchical information, either different levels of abstraction or disjunctive decompositions, the ontology provides the intra-group ParentOf relationship.

Additionally, the problem map model has intergroup relationships linking the five different groups of entities together. For an example of the graphical representations of some of the problem maps generated from the protocol data see Fig. 1 and Fig. 2. This ontology was represented in Answer Set Prolog (ASP) and then used to encode the protocol data. Fig. 3 shows the ASP encoding of a portion of the problem map in Fig. 1. For a more in depth discussion of the problem map model, see [1].

RELATED WORK

In this section we discuss three different types of related works: concept maps, semantic networks, and the function, behavior, structure model (FBS) and its variants.

Concept Maps

Concept maps are already used in design education, as a means of providing students with an easy and intuitive way of documenting and explaining their designs. This provides them with insight into the systems they design [8]. Novak and Cañas [9] originally proposed the use of concept maps to identify changes in students' understanding over time. Additionally, concept maps have been used to understand the differences between the knowledge of experts and novices. Several metrics have been developed to assess concept maps and understand these differences. For an overview, see [10].

Though concept maps have nodes and links between concepts, and provide a hierarchical representation, they are still relatively unstructured. Since our intended application is engineering design, we can take advantage of the structure imposed by the design process itself. Therefore, we have opted to use problem maps, which have been developed with the structure of engineering design in mind.

Semantic Networks

Semantic Networks [11], based on the work of Charles Peirce [12], are a type of graphical network, which relate conceptual nodes with binary links. Within a semantic network concepts are usually organized in a taxonomic hierarchy and often rely on the use of hierarchy and inheritance [13]. In contrast to Concept Maps, Semantic Networks are not often associated with design formulation, but they provide an intuitive and expressive means to represent conceptual structures.

Though semantic networks are as expressive as first order logic they have many shortcomings. First, they struggle to represent disjunction and non-taxonomic knowledge and can only do so with the use of work arounds [13]. We believe that disjunction and non-taxonomic knowledge play an important role in conceptual design. Furthermore, Semantic Nets were designed to be general and therefore do not take advantage of the implicit structure of engineering problems. These are the primary reasons that we chose to use problem maps to represent our design formulations; they can represent disjunction and non-taxonomic knowledge and they take advantage of the implicit structure of the engineering design process.

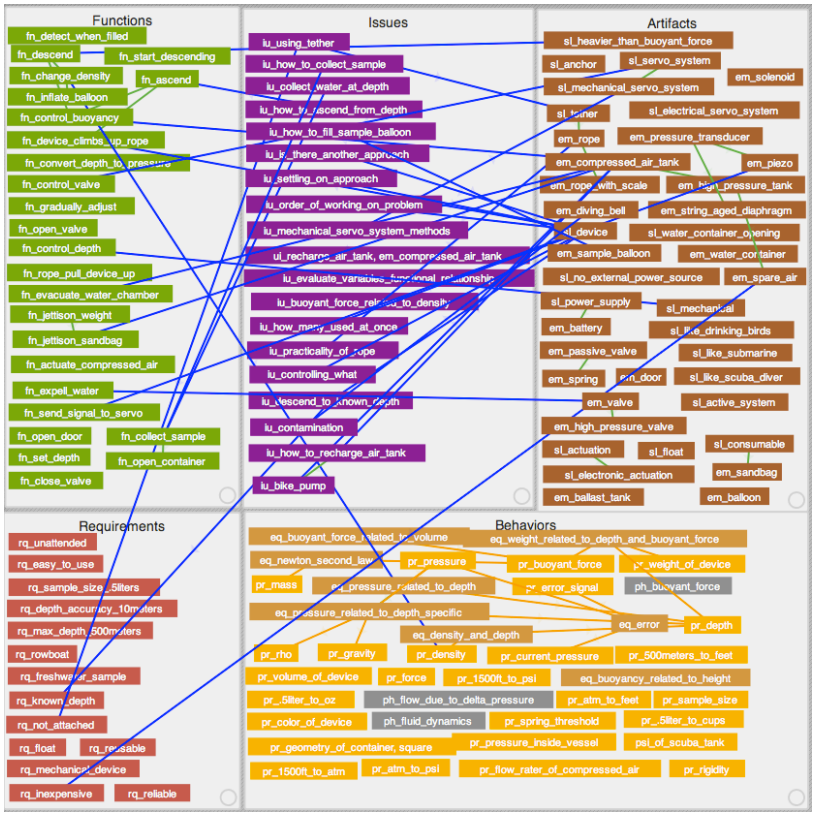

Figure 1. A PROBLEM MAP FROM A MORE CREATIVE DESIGNER. THIS DESIGNER SPENT A SIGNIFICANT AMOUNT OF EFFORT ELABORATING ON BEHAVIORS.

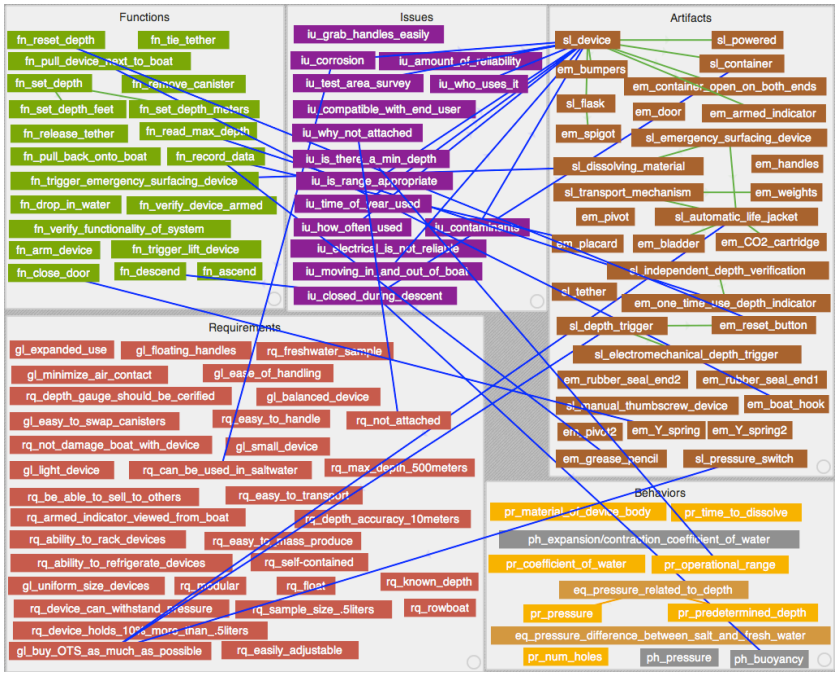

Figure 2. A PROBLEM MAP FROM A LESS CREATIVE DESIGNER. THIS DESIGNER SPENT MORE TIME ON THE HIGH LEVEL REQUIREMENTS AND LESS TIME ON SPECIFIC FUNCTIONS, ARTIFACTS AND BEHAVIORS.

```
requirement(rq_not_attached)
requirementType(rq_not_attached, given)
physicalEmbodiment(em_rope)
solutionPrinciple(sl_tether),
parentOf(sl_tether, em_rope)
solutionPrinciple(sl_anchor)
issue(iu_using_tether, "needs to be unattached")
related(iu_using_tether, rq_not_attached)
related(iu_using_tether, sl_tether)
physicalEffect(ph_buoyant_force)
function(fn_control_buoyancy)
controls(ph_buoyant_force, fn_control_buoyancy)
equation(eq_newton_second_law, f = mass * gravity, concrete)
parameter(pr_gravity)
parameterEquation(pr_gravity, eq_netwon_second_law)
```

Figure 3. THE EQUIVALENT ASP ENCODING OF A SMALL PART OF THE PROBLEM MAP IN FIGURE 1.

Function, Behavior, Structure and variants

Gero [14] described design as a series of transformations among sets of functions, behaviors and structures, which led to a design description. For example, catalog lookup is a direct mapping from functions to structures. The FBS framework has become a prominent part of research in design theory in the past twenty years and has evolved since its introduction. The initial framework made a distinction between expected behavior and actual behavior. Gero and Kannengiesser [15] extended this distinction to the situatedness of design in general. Gero and associates have used the FBS framework to model the design process [4,16] and to encode protocol analysis [5].

Other variants of the FBS framework have been developed independently. Chandrasekaran [17] developed Functional Representation as a language to describe the function of an artifact in terms of causal processes in order to explain how an artifact works. Later, Chandrasekaran and Josephson [18] added an environment-centric view to allow for more precise ways of representing design knowledge. Goel et al. [19] have developed the Structure-Behavior-Function modeling language for a teleological description of complex systems. In this language, structure, behavior and function are represented with components and their connections, transitions among a sequence of states, and pre and post conditions. The model is a nested top-down schema, where each concept in the model is defined by another concept at a lower level of abstraction. At the top there is a high level SBF instance while at the bottom there are the building blocks e.g. strings and integers. For example, a component in a structure model is defined by an integer for Id, a string for name, a string for description, an optional set for other properties, and an integer to refer to a sub-element Id.

FBS has been used to create a common consensus among design researchers in defining fundamental concepts of design theory. However, Gero and Kannengiesser [20] contend that the framework leads to a high level model of design. There is a need for a more expressive and flexible model that goes beyond the general concepts of the FBS framework.

EXPERIMENT

We conducted eight hour-long sessions with expert designers working at the same company. None of the designers had any previous knowledge about the problem map framework. In the sessions we provided them with the following prompt from [21]:

"Design a mechanical device to be used from a rowboat by a researcher who wishes to collect samples of water from fresh-water lakes (e.g. Lake Tahoe) at known depths down to a maximum of 500 m.

After release, the device must not be attached to the boat and must descend to within 10 m of an easily adjustable pre-determined depth. It must return to the surface with a 0.5 liter sample of water from that depth and then float on the surface until picked up.

The device should be reliable, easy to use, reusable, and inexpensive."

We asked them to think aloud [22] and verbalize any thoughts as they worked on the design task. During the one-hour design sessions, we digitally recorded the sketches they made and videotaped the designers during their sessions. Additionally, each designer was administered a divergent thinking test [23].

The recordings were transcribed by native English speakers. Segmentation and coding was initially done by one of the researchers, and a final protocol coding was decided through a process of arbitration with a second researcher. Both researchers read the transcription and listened to the recordings. Although the sketches contained some additional spatial information about the designers, for the purposes of this study we only used the sketches to clarify questions the researchers had about the designers' verbal descriptions.

Additionally, we did not code any reasoning that would result in an immediate deletion of the entity just added. For example one designer stated the following: "So P over V assuming constant temperature, I'm going to assume constant temperature in this water and that probably will go into my margin of 10 meters because I'm going to assume constant temperature." Then he refutes the previous statement immediately with "But as anyone knows, there's a temperature gradient – as you get deeper it's going to get colder."

For the purposes of analysis, each time step was associated with only one code. The coded transcriptions were used to create problem maps, as shown in Fig. 1 and Fig. 2. Additionally we calculated the following metrics for each designer:

- Total number of overall entities
- Total number of links between entities – this includes both intra- and intergroup relationships.
- Average number of vertices – this metric was used as a measure of connectedness of entities within a problem map.

	# of entities	# of links	avg. vertices	total reqs.	total fun.	total art.	total beh.	total issues	total parentOf
overall creativity	0.627	0.600	0.058	-0.305	0.158	0.250	0.870	0.684	0.464
fluency	0.544	0.403	-0.301	0.137	0.157	0.165	0.707	0.397	0.288
flexibility	0.487	0.433	-0.014	-0.082	0.404	0.356	0.391	0.167	0.401
originality	0.435	0.430	0.095	-0.533	0.697	0.687	-0.018	0.012	0.772
mx originality	0.621	0.676	0.292	-0.615	0.620	0.608	0.420	0.356	0.724
quality	0.262	0.335	0.256	-0.319	-0.343	-0.172	0.815	0.778	0.035
decomplexability	-0.118	-0.450	-0.875	0.561	-0.263	-0.113	-0.113	-0.174	0.068
detailability	0.126	-0.116	-0.587	0.441	0.440	0.355	-0.368	-0.506	0.325
abstractability	-0.140	-0.097	0.057	0.054	-0.723	-0.610	0.587	0.525	-0.494

Figure 4. THIS TABLE SHOWS THE CORRELATIONS BETWEEN SEVERAL DIRECT AND INDIRECT METRICS OF CREATIVITY (ROWS) AND DIFFERENT METRICS BASED ON THE PROBLEM MAP ONTOLOGY (COLUMNS). CORRELATIONS WITH A $P < 0.05$ ARE HIGHLIGHTED IN YELLOW, WHILE THOSE WITH A $P < 0.10$ ARE SHOWN IN ORANGE.

- Total number of requirements
- Total number of functions
- Total number of artifacts
- Total number of behaviors
- Total number of issues
- Total number of parent-child relationships – the intra-group relationship specifying hierarchical information in the problem map.

We ran a linear correlation of these metrics against four direct measures of creativity (fluency, flexibility, originality and quality) and three indirect measures of creativity (decomplexability, detailability, abstractability). Due to the fact that we used only the problem map representing all of the entities that had been generated by the end of the design session, afixability could not be used. The final measure of creativity used was an overall creativity score calculated by averaging the four direct measures of creativity. For a more detailed discussion of these creativity measures, see [24]. The results can be seen in Fig. 4.

RESULTS AND DISCUSSION

Our aim was to test whether the problem map model was an effective way to study creativity. To show its effectiveness, we will begin by discussing our quantitative results. This includes examples of how we can use graphical representations of the data collected through the problem map ontology to gain insight into designers' problem formulations. We will discuss some of the correlations we found in our exploratory study. Finally, we will discuss some qualitative observations that were made during the coding and analysis, that provide us with some insight into what can and cannot be represented with the problem map ontology.

Quantitative Results

Due to the fact that our exploratory study was run with only industry level expert mechanical engineers, the overall creativity of the designers was relatively similar. The creativity scores for the designers ranged between 4.91 and 6.69 on a scale of 1 – 10. Though the difference between our designers' creativity levels was small and this study was largely exploratory, we discovered some interesting correlations between the number of entities within individual designer's problem maps and their creativity scores.

As seen in Fig. 4, the number of overall behavior entities the designers specified was strongly correlated with the overall creativity level of the designers, as well as two direct metrics of creativity: fluency and quality. Additionally, we found that the total number of issues the designers specified positively correlated with the quality metric. We also found that amount of elaboration as measured through the number of parent-child relationships correlated with the designer's measured originality.

In addition, decomplexability was inversely correlated with the average number of vertices. This may point to more creative designers elaborating downwards rather than thinking of many alternatives or decompositions. This would require additional investigation. Finally, the total number of functions specified during the session was inversely correlated with abstractability. We found no correlations between the number of requirements and any of the creativity metrics, but in our sample size, the designers all had a similar number of specified requirements, usually directly given by the prompt.

To better understand the differences between our less creative and more creative designers, we plotted the overall number of entities within the five groups over a normalized timescale to eliminate differences in the length of the design sessions. These graphs can be seen in Fig. 5 and Fig. 6. Fig. 5 shows the visualized graphs for our two most creative designers, and Fig. 6 the graphs for our two least creative designers. As is clearly visible from the graphs the more creative designers specified more behaviors than any other entity. On the other hand, the number of behaviors specified by the less creative designers, was significantly less than the number of functions and artifacts specified. Additionally, with the exception of one designer, all the others specified all their requirements at the very beginning of the design session only, and then focused on the other four groups of entities.

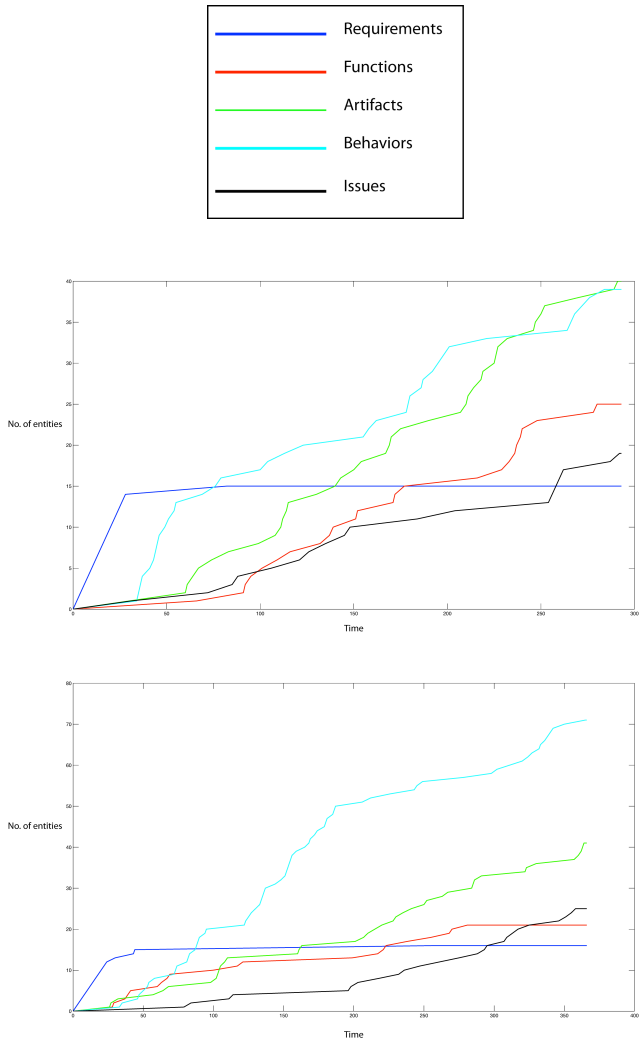

Figure 5. TWO GRAPHS FROM THE MORE CREATIVE DESIGNERS SHOWING THE CUMULATIVE NUMBER OF ENTITY IN EACH OF THE FIVE GROUPS OVER TIME. NOTE DIFFERENCE BETWEEN THE NUMBER OF BEHAVIORS SPECIFIED (CYAN LINE) IN THESE TWO GRAPHS AND THOSE IN THE FOLLOWING FIGURE.

To see whether or not our designers approached their design sessions in a similar order, we created several graphs similar to Fig. 7 showing a box and whisker plot of when two of our designers defined their requirements, functions, artifacts, behaviors and issues. As mentioned previously, the designer that defined requirements throughout the design process was atypical, and in fact, the rest of our designers only specified requirements towards the beginning of their sessions, similar to the second designer in Fig. 7. Additionally, most of our designers specified the vast majority of the issues they had with the design towards the end of their design sessions as they were reviewing the design space they had explored, though other issues were identified throughout the sessions. Though

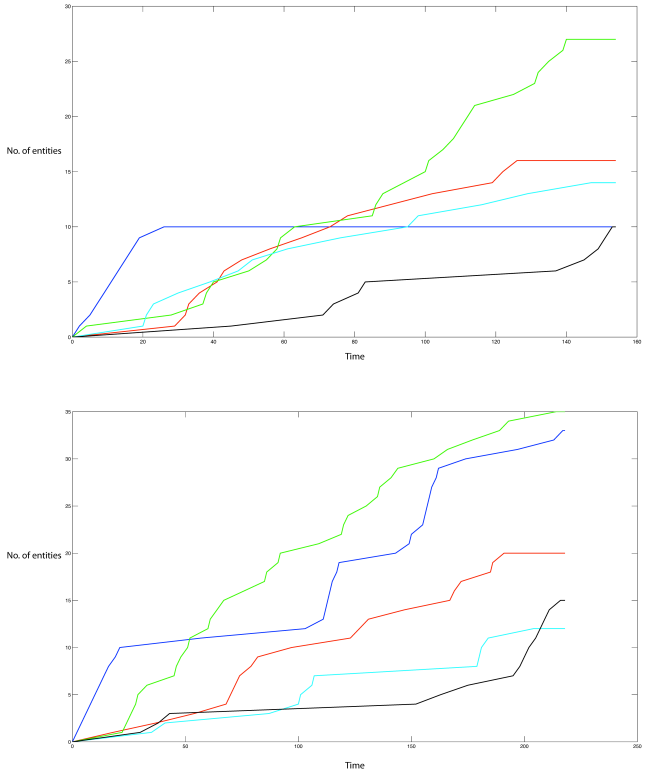

Figure 6. TWO GRAPHS FROM LESS CREATIVE DESIGNERS SHOWING THE CUMULATIVE NUMBER OF ENTITY IN EACH OF THE FIVE GROUPS OVER TIME.

designers have different styles of problem solving, which are not dependent on the solution [25], there are some similarities in the ways in which designers move between the five groups of entities.

Finally, as can be observed in Fig. 8, which shows a timeline of how one designer moved between the five groups of entities, the process of defining artifacts, behaviors and functions was strongly intertwined. This can also be seen in the second graph in Fig. 5 and the first graph in Fig. 6. Our designers often went back and forth quickly between defining their artifacts and their functions. For those designers that also spent a large amount of effort identifying behaviors, the behaviors were also intertwined with functions and artifacts.

Qualitative Observations

Several qualitative observations were made throughout the coding process. In this section we discuss these observations as they relate to the problem map model. We will also show examples where possible.

Some of these observations relates to the issues in the problem map ontology. Unlike, FBS, the problem map model also incorporates issues and requirements. Although we had initially anticipated that the number of issues generated would

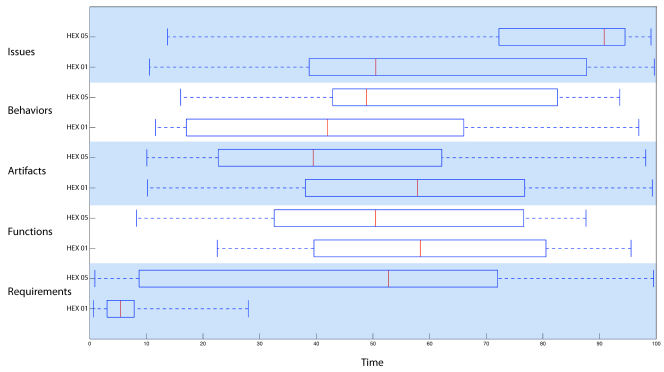

Figure 7. THE GRAPH SHOWS HOW DESIGNERS ADD DIFFERENT ENTITIES AS THEY PROGRESS THROUGH THEIR PROBLEM FORMULATION.

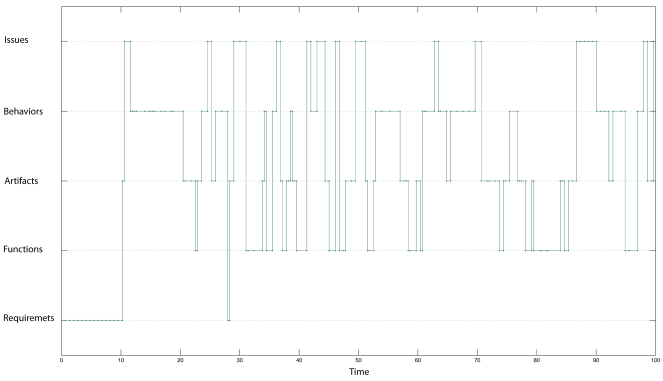

Figure 8. A TIMELINE OF HOW ONE DESIGNER MOVED BETWEEN THE FIVE GROUPS OF ENTITIES.

provide insight into the creativity of the designer, there was only a moderate correlation in our data between the creativity metrics and the number of issues. Additionally, while the model supports hierarchies in all five groups, we did not observe the designers specifying hierarchical information in the issues or the requirements. Instead, most issues were not related to each other at all, and instead were related to different parts of the design space. Issues were also related to all four of the other groups of entities, though it was much more unlikely that they would be related to behaviors, than to requirements, artifacts and functions.

Additionally, one of our more creative designers made the comment that: "Staying balanced is another issue, I'll put that up on my issues list. I'm going to start an issues list now. Okay, issues – maintaining balance." Although this was not a typical strategy, this particular designer found it helpful. Other designers usually discussed the issues as they went along and either became stuck, or ran into a conflict between the requirements and the functions or artifacts. Another observation regarding the issues was that, while issues do not have importance levels, unlike goals, some designers did specify whether they thought an issue was critical or whether it could be safely ignored at this stage of the design process. Throughout the sessions, designers may resolve some of the issues that they bring up, but they often did not explicitly specify that they had done so in the think aloud process.

Proto-solutions are partial solutions or intentions that the designer contemplates in the solution space, plugs into the problem space, and checks whether there is a fit with the problem space or not [26]. Some of the designers also specified the pros and cons of different proto-solutions through describing the different issues associated with each proto-solution. For example, consider this quote from one of the transcriptions: "Uh, this is kind of interesting because when we open, when we open the sampling chamber it dramatically changes the buoyancy. That was one really nice advantage of the cable. The cabling method we can have the sampling device be heavily weighted. It can be very heavily weighted so that it is immune in the difference of buoyancy of taking the half liter sample....Okay and as I was talking before…"

As we have mentioned before, many of the requirements were specified at the beginning of the design session, and additional requirements were often not added after this point. If designers did add additional requirements, it was often only a few. It was significantly more likely for them to add goals to help them throughout the session instead. Although the problem map model supports the ability to specify importance levels of goals and requirements, we noticed that designers often did not specify that they found a specific goal or requirement more important than another. One contradiction to this, was a designer who stated: "I'm going to do it in order where I'd rather it be reliable, easy-to-use and reusable than inexpensive at a point of purchase. You want this thing to be reusable, so you want something that you buy and isn't going to just completely fall apart."

We also observed some designers abstracting upward during the design session. For example, one designer first specified that he was going to use a sandbag to control buoyancy, and then later modified this to a consumable object that realized that same function. This is visible in the coding through the following pattern:

- physicalEmbodiment(em_sandbag)
- solutionPrinciple(sl_consumable)
- parentOf(sl_consumable, em_sandbag)

Sometimes this process of abstracting upwards happened shortly after introducing the physical embodiment that they would use, other times it happened much later in the design process when looking for lower cost or easier to use alternatives. In these instances, often the designer would abstract up and then immediately create an alternative to the initial physical embodiment proposed. These types of observation are made clear by using the problem map framework.

Many of the designers also used analogies in their design sessions. These analogies were coded through solution principles. Though we initially thought we would not use the problem map model for analogical reasoning, we found that it is helpful to use it during the coding process. This allowed us to

see that one of our more creative designers used several analogies throughout his design session. One was that he wanted to design a device that would work like a submarine or a scuba diver. He also mentioned that instead of using an external power source, he would rather design something that worked like those drinking birds that rock up and down through purely mechanical means instead of using power. A few other designers also mentioned that they wanted to design a device that worked on the same principles of buoyancy that a submarine or a scuba diver uses.

Though it was not a common occurrence for designers to delete an entity they had previously considered, it did happen. For example, one designer was initially considering using a one-way valve. Later in the design session, he decided that the one way valve would not realize the function he wanted it to, so he scratched it out on his paper and specified that he would no longer use it. Throughout all eight transcriptions, we only encountered a few instances of a designer deleting something previously considered. Instead, if a designer realized that a specific physical embodiment had too many issues associated with it, or it would not suit the purpose he had intended, often he would just start considering alternatives and stop pursuing the entity that was problematic.

Though many of the designers mentioned that if they were going through a "real" design session they would look up certain pieces of information such as physical equations, existing technologies, etc, for the purposes of the session, they often would make their own assumptions and continue exploring those design sessions based on their assumptions. Designers also mentioned that they would talk to the customer to verify that their assumptions are correct, or to gather additional information that was not provided to them (for example, how often the device would be used, who would use it, what the usual depth of operation would be, etc.). This observation is consistent with Gero and Kannengiesser [15], who state that designers use their experiences to interpret representations that are augmented with implicit requirements.

Others pursued multiple possible design solutions, rather than making these assumptions. This is especially clear in one instance in which a designer considered designing both an autonomous sampling device and one controlled through remote control by the researcher taking the sample. These proto-solutions were dependent upon the number of devices that the researcher would want to deploy at once, which was information that was not available to the designer during the design session. If the researcher would want multiple devices deployed simultaneously then an autonomous device would be better than a remote controlled one, but if only one device would be deployed, the designer thought remote control would be more reliable, and therefore preferred. These observations regarding the approaches designers took to an incomplete problem statement are consistent with Fricke [27], who noted that the completeness of the problem statement affects the problem solving strategy used. The problem map model allows for coding such design choices.

One unanticipated observation was that designers often specified functions that are not to be realized by the object they are designing, but instead by people or other devices. For example, in our water sampler prompt, the designer may specify that the researcher should put the device in the water, push a button to calibrate it, and measure weights based on the depth at which the researcher wants to collect his sample. Though the problem map model was not intended to support this type of coding, it is possible to code these functions in the ontology.

Though we did not observe designers pursuing wildly different solutions through the course of the session, they did often consider alternatives for parts of their designs. For example, they might consider using several different trigger mechanisms for determining whether or not they're at depth, such as electronic pressure sensors, diaphragms, etc.

Limitations of the problem map model

While the problem map model allowed us to represent a large part of the problem formulation process the designers went through, there were some things that could not be coded using the model as it is now. One of these limitations is that specific groundings, or proto-solutions, the designer specified cannot be identified using the problem map model. Though the problem map model represents the space the designer could have explored, they may not have necessarily explored all groundings in the space.

Another limitation is that the model is designed to be domain independent. While this is a major strength of the model, this also means that without domain knowledge, the different combinations of possible designers that may be generated from the problem map may contain artifacts, or other entities that may not combine well or at all in reality. This is a specific example of the limitation introduced in the previous paragraph. In order to allow for this information to be entered, the problem map model would need to allow the designer to specify when two entities cannot be combined into one proto-solution.

There is also currently no way to specify whether the children of a parent are both required or if they are disjunctive when interpreting the transcriptions. For example, a device may either require a regular valve or a one-way valve, or both may be required in different parts of the device. These valves would be coded in the following way regardless of whether they are conjunctive and disjunctive:

- physicalEmbodimet(em_one_way_valve)
- physicalEmbodiment(em_valve)
- parentOf(sl_device, em_valve)
- parentOf(sl_device, em_one_way_valve)

This is due to the nature of how these physical embodiments are often introduced in the protocols and the fact that proto-solutions often overlap, sharing many entities. This information can be encoded using the problem map framework, but encoding hierarchical information from protocol studies is prohibitive. Automated tools that code this information as the

user specifies it would make the hierarchy easily accessible for analysis.

Additionally, in some instances, designers will connect components to the high-level solution principle of the device. When a more specific device is mentioned, it may be the case that the child does contain the components connected to the high level device, or it may be the case that those high level components are actually connected to a disjunctive solution. This is another piece of information that is prohibitive to code during a protocol study but may be supported through automated tools.

Functions specified by the designer may be used in a sequence multiple times with different parameter values. While the problem map model does code sequential information with the before relation, there is no way to specify which parameter value goes with which instance of the function. For example, one designer ascended three times during the process of collecting the sample. The first time, the designer wanted to ascend ten meters, puncture a balloon, ascend another predetermined amount, collect the sample, and then drop the weights and ascend the remaining distance to the surface.

Another piece of information that is hard to encode is whether a parent solution principle of a physical embodiment is an abstract solution principle guiding the selection of entities, or a parent, which contains the child physical embodiment. For example one designer specified that the design should incorporate disposable liners for the water-sampling container to avoid contamination between samples. This liner therefore was specified as both a child of the solution principle sl_disposable and as a child of the sl_water_sampler though these relationships are different. In another example, one designer first specified that he wanted a water container, and that this device should have a balloon. Later he elaborates and says that he wants a pressure containment vessel as his water container. Both pressure containment vessel and balloon would have been coded as children of the higher lever water container.

Another observation was that the coding scheme links parameters, such as spatial location to the entity the location information belongs to, but not necessarily the entity that it affects. For example, if a solutionPrinciple sl_device has an embodiment em_hatch, the parameter (pr_hatch_location) would be linked to the device, without any sort of link to the em_hatch. While this type of information was not necessary for the analyses presented earlier in this paper, it may become more relevant when looking at determining the quality, quantity, fluency, and originality of a problem map.

Finally, we found that designers will often specify information about what does not need to be considered in the design space. For example, one designer concluded that, since the device was intended for freshwater use only, salt erosion, oxidation or any contamination of the materials could be safely ignored. There is currently no clear way to code this information. On the other hand, the model does allow for statement such as "the device should be made out of materials that do not become contaminated and that should be resistant to salt erosion or oxidation."

FUTURE WORK AND CONCLUSION

We are currently developing an interactive tool that will allow designers to create their own problem maps as part of the problem formulation process. In the future, we will use the tool to collect more information about the problem formulation process. Additionally, we will use the insights gained from this exploratory study to develop new ways of fostering creativity during the design process.

In the near future, we aim to develop ways to measure the quality, quantity, fluency and originality of the designer through the problem maps. In the paper, we presented some encouraging results that provide insight into how these metrics can be obtained. Additionally, we would like to run a larger study using the tool to collect data, instead of coding protocols. The analysis techniques presented in this paper can be used for the data collected, hopefully further providing us with insight into the process of problem formulation.

Finally, we hope that we can develop ways to aid creative design through automatic issue generation. We would also like to develop methods to encourage designers to focus their attention on parts of the design space that are more critical and more likely to result in the generation of creative solutions.

In this paper, we claimed that the problem map framework could be used as a new tool in design research by providing us with a predefined method for analyzing protocol data. We then presented an exploratory study showing how one could use the problem map model to code and analyzing our protocol data. We discussed the strengths and weaknesses of this method and how we can use the method to identify the role of creativity in the problem formulation process. While the model has some limitations, we believe that this method shows promise and can be further used to discover how creativity plays a role and how to develop tools based on the problem map model to understand and encourage creativity in the problem formulation stage.

ACKNOWLEDGMENTS

This study is supported by the National Science Foundation, CMMI grant number 1002910. The opinions expressed in this paper are those of the authors and not endorsed by NSF.

REFERENCES

[1] Dinar, M., Shah, J. J., Langley, P., Hunt, G., Campana, E., 2011. "A structure for representing problem formulation in design". Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, pp. 1–10.

[2] Adelson, B. and Solway, E., 1988. "A Model of Software Design." In Chi, M.T.H., Glaser, R. and Farr, M.J., eds., *The Nature of expertise*, L. Erlbaum Associates, Hillsdale, N.J., pp. 434.

[3] Ullman, D.G., Dietterich, T.G. and Stauffer, L.A., 1988. "A model of the mechanical design process based on empirical data." *AI EDAM*, 2(01), pp33-52.

[4] Gero, J. S., and Mc Neill, T., 1998. "An approach to the analysis of design protocols." *Design studies*, 19(1), pp. 21–61.

[5] Pourmohamadi, M., and Gero, J. S., 2011. "LINKOgrapher: An Analysis Tool to Study Design Protocols Based on FBS Coding." Proceedings of the International Conference on Engineering Design. Copenhagen, Denmark.

[6] Cross, N., Dorst, K. and Roozenburg, N., eds., 1992. "Research in design thinking: proceedings of a workshop meeting held at the Faculty of Industrial Design Engineering", Delft University of Technology, the Netherlands, May 29-31, 1991. Delft University Press.

[7] Cross, N, Christiaans, H, and Dorst, K., eds., 1996. *Analysing Design Activity*. John Wiley, Chichester.

[8] Oxman, R., 2004. "Think-maps: teaching design thinking in design education." *Design Studies* 25: pp. 63-91.

[9] Novak, J. D., and Cañas, A. J., 2008. "The Theory Underlying Concept Maps and How to Construct and Use Them." Tech. rep., Florida Institute for Human and Machine Cognition (IHMC).

[10] Hao, J., Chi-Wai Kwok, R., Yiu-Keung Lau, R., and Yan Yu, A., 2010. "Predicting problem-solving performance with concept maps: An information-theoretic approach." *Decision Support Systems* 48: pp. 613-621.

[11] Richens, R. H., 1956. "General program for mechanical translation between any two languages via an algebraic interlingua". *Mechanical Translation* 3 (2), pp. 37.

[12] Peirce, C. S., 1906. "Manuscripts on existential graphs." *Collected Papers of Charles Sanders Peirce,* Vol. 4, pp 320-410, Harvard University Press, Cambridge, MA.

[13] Lehmann, F., 1992. "Semantic Networks." *Computers & Mathematics with Applications*, 23, (2–5), pp. 1-50.

[14] Gero, J. S., 1990. "Design prototypes: a knowledge representation schema for design." *AI Magazine*, 11(4), pp. 26-36.

[15] Gero, J. S., and Kannengiesser, U., 2004. "The situated function-behaviour-structure framework." *Design Studies*, 25(4), pp. 373-391.

[16] Gero, J. S., and Kannengiesser, U., 2007. "A function–behavior–structure ontology of processes." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 21(04), pp. 379-391.

[17] Chandrasekaran, B., 1994. "Functional Representation: A Brief Historical Perspective." *Applied Artificial Intelligence*, 8(2), pp. 173-197.

[18] Chandrasekaran, B., and Josephson, J. R., 2000. "Function in Device Representation." *Engineering with Computers*, 16(3-4), pp. 162-177.

[19] Goel, A. K., Rugaber, S., and Vattam, S., 2009. "Structure, Behavior and Function of Complex Systems: The Structure, Behavior, and Function Modeling Language." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1), pp. 23-35.

[20] Gero, J. S., and Kannengiesser, U., 2007. "Locating Creativity in a Framework of Designing for Innovation." In *Trends in Computer Aided Innovation*, N. León-Rovira ,ed., Vol. 250, Springer, Boston, pp. 57-66.

[21] Pahl, G. and Beitz, W., 1996. *Engineering Design — A Systematic Approach*, 2nd ed., Springer-Verlag, London.

[22] Ericsson, K. A., and Simon, H. A. 1993. Protocol analysis: Verbal reports as data. Cambridge, MA: MIT Press.

[23] Shah, J. J., Millsap, R. E., and Woodward, J. (2012). Applied Tests of Design Skills — Part 1: Divergent Thinking. Journal of Mechanical Design, 134(February), 1-10.

[24] Shah, J. J., Vargas-Hernandez, N. and Smith, S. M., 2003. "Metrics for measuring ideation effectiveness." *Design Studies* 24 (2): 111-134.

[25] Eisentraut, R., 1999. "Styles of problem solving and their influence on the design process." *Design Studies*, 20, pp. 431-437.

[26] Harfield, S., 2007. "On design "problematization": Theorising differences in designed outcomes." *Design Studies*, 28(2), pp. 159-173.

[27] Fricke, G., 1999. "Successful approaches in dealing with differently precise design problems." *Design Studies*, 20(5), pp. 417-429.

