Activity-Centered Domain Characterization for Problem-Driven Scientific Visualization

G. Elisabeta Marai

Abstract—Although visualization design models exist in the literature in the form of higher-level methodological frameworks, these models do not present a clear methodological prescription for the domain characterization step. This work presents a framework and end-to-end model for requirements engineering in problem-driven visualization application design. The framework and model are based on the activity-centered design paradigm, which is an enhancement of human-centered design. The proposed activity-centered approach focuses on user tasks and activities, and allows an explicit link between the requirements engineering process with the abstraction stage—and its evaluation—of existing, higher-level visualization design models. In a departure from existing visualization design models, the resulting model: assigns value to a visualization based on user activities; ranks user tasks before the user data; partitions requirements in activity-related capabilities and nonfunctional characteristics and constraints; and explicitly incorporates the user workflows into the requirements process. A further merit of this model is its explicit integration of functional specifications, a concept this work adapts from the software engineering literature, into the visualization design nested model. A quantitative evaluation using two sets of interdisciplinary projects supports the merits of the activity-centered model. The result is a practical roadmap to the domain characterization step of visualization design for problem-driven data visualization. Following this domain characterization model can help remove a number of pitfalls that have been identified multiple times in the visualization design literature.

Index Terms—Design studies, Tasks and requirements analysis, Visualization models, Domain characterization, Activity-centered design, Functional specifications

1 Introduction

Visualization relies significantly on data from other domains, from biomedicine to computational fluid dynamics. As we train and train others into research methodology, we almost always train also into interdisciplinary collaboration [35]. These interdisciplinary visualization projects may aim to improve or adapt an established visualization technique, or to develop a totally innovative visualization tool with no obvious precedent. Regardless of the aim of the project, the domain experts' needs, tasks and goals, the conditions under which the visualization will be used, and the constraints on the visualization performance have to be first discussed, characterized, clarified, and sometimes re-scoped. In existing visualization design models, this first step is referred to as "characterizing the task and data in the vocabulary of the problem domain" [38], and is the visualization equivalent of the "requirements engineering" first step in the Human Computer Interaction (HCI) design process or in the software design process. Subsequent layers of the visualization design process, from visual encoding to implementation, depend on this first step of domain characterization.

Characterizing the application domain presents considerable challenges for both visualization designers and domain experts, due to the inherently exploratory nature of the domain problems and to the variety of data involved. Some of the challenges inherent in bridging the gaps of knowledge and interest between designers and users are discussed by van Wijk [55]. The designers may not have sufficient domain knowledge to extract or even understand the expert's needs. They may have trouble abstracting fuzzy needs into visualization terms, and aligning those needs with their own research interests. As a result, designers sometimes end up solving the wrong problem, or using an uninformative visual encoding. From the other end, the experts may have limited time available to "apprentice" a visualization researcher. They may not notice what they do, may not know how to articulate what they do, and may misrepresent reality, precisely because the domain activities seem complex and fuzzy [39].

• *G.E. Marai is with the Electronic Visualization Laboratory at University of Illinois at Chicago. E-mail: g.elisabeta.marai@gmail.com.*

Manuscript received 31 Mar. 2017; accepted 1 Aug. 2017.
Date of publication 28 Aug. 2017; date of current version 1 Oct. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2017.2744459

While general prescriptions for requirements engineering do exist, domain characterization in scientific visualization differs in multiple ways from HCI or software engineering, and, to a minor extent, even from the equivalent step in information visualization. First, spatial data can seldom be imitated outside the target domain. Data also often requires an upfront commitment to potential cleaning, integration, and pre-processing, as well as understanding the mathematical or computational underpinnings of the domain problem. This constraint often delays the production of a prototype. Second, domain experts have limited availability, when compared to the general public. There is also less emphasis on user feelings and emotions, compared to HCI. When compared to general software design, there is additional emphasis on the relevance of the human visual system and perception to the user tasks. Last but not least, the field emphasizes novelty in either the problem or in the approach. This set of differences makes it difficult to articulate how the visualization domain characterization process relates exactly to other models in the literature, for example Human-Centered-Design (HCD). HCD models implement a cycle with four sequential steps: observation, followed by ideation, followed by prototyping, and only then followed by user testing [40].

A step further, although visualization design models exist in the literature [14, 24, 37, 38, 45, 52, 57] in the form of higher-level methodological frameworks, these models do not present a methodological description of each step to be followed. For example, with respect to the domain characterization step, no model discusses how to improve the tasks abstraction process when the user tasks are seemingly ill-defined.

This work presents a next step to the higher level frameworks, and goes deeper into a specific model for the initial step of domain characterization in visualization. The model is based on the activity-centered design paradigm [40], which is inspired by Activity Theory [16, 25, 56]. Activity-centered design focuses on activities, not the individual person. In the 2013 assessment of Don Norman (The Design of Everyday Things, 3rd edition), the underlying principle is that "since people's activities around the world tend to be similar, and because people are quite willing to learn things that appear to be essential to the activity, activity should be allowed to define the product and its structure" [40]. Since the activities are performed by people and for people, activity-centered design is an enhancement of HCD [40].

The model proposed in this work under the activity-centered frame is organized in four chronological stages (Fig. 1): 1) Notification; 2) Activity Inquiry; 3) Establishing Context; and 4) Analysis, Specifi-

cation and Validation. As part of the activity-centered approach, this model recasts the problem of domain characterization into the larger, and better studied, framework of software development and functional specifications. The software development field, in which scientific visualization is partly rooted, has been confronting similar challenges to problem-driven visualization research, and may offer solutions to common visualization pitfalls. In particular, the activity-centered paradigm allows an explicit integration of the concept of workflows and functional specs into the existing Human-Centered visualization design models.

The activity-centered model focuses on a specific technique for engineering requirements for visualization design, and provides a practical roadmap and a specific agenda for the process. This model has been derived from a collective experience of building multiple award-winning, user-adopted visualization tools, as well as several flawed projects, and is informed by the visualization, human computer interaction and software engineering literature. While this model was developed based on experience mostly with scientific visualization projects, the messages and lessons within the model may be applicable to the wider visualization community. The contributions of this work are:

- An activity-centered framework and an end-to-end network model for the requirements process in visualization design. The framework is based on the activity-centered design paradigm. In a departure from existing visualization design models, the resulting model: a) assigns value to a visualization based on user activities; b) ranks user tasks before the user data; c) partitions requirements in activity-related capabilities and nonfunctional characteristics and constraints; d) explicitly incorporates the user workflows, as a first-class citizen, into the requirements process; and e) establishes the need for visualization early in the design process, based on the user activities.

- An extension of the nested visualization design model [38] through the explicit integration of *functional specifications* as a systematic means to validate the domain characterization and to communicate back with the user, prior to the prototyping stage. Functional specifications (not related to the functional programming paradigm) originate from the software development field; they have not been integrated explicitly before with HCD models. Furthermore, this model: a) articulates the difference between requirements (what the user needs) and functional specifications (what the designer intends to do); b) ensures the functional specs include what the product will *not* do; c) mandates that the functional specs be reviewed by the user; and d) promotes practical advice on engaging spec writing, based on the user workflows.

- A quantitative evaluation and qualitative discussion of the merits of this framework, and of the fit of the model with existing visualization design frameworks. In particular, this work shows how performing the domain characterization step correctly through the activity-centered model can remove a number of pitfalls that have been identified multiple times in the visualization design literature, including lack of real data and solving the wrong problem.

2 Background and Vocabulary

Domain Characterization. Domain characterization is the first stage of visualization design, where visualization design denotes developing a digital system that allows users to find insight into data through visual representations and interaction methods. During this characterization stage, the visualization designer must learn about the tasks and data of target users in some particular target domain [38], similar to the process of establishing requirements in software engineering and HCI. The visualization literature uses, in fact, interchangeably the terms "domain characterization" and "requirements elicitation". The output of this stage is a set of requirements about the visualization design, which may also include information about the user needs, tools, or stakeholders.

Requirements. A "requirement" is a statement about an intended product that specifies what the product should do or how it should perform [42]. The purpose of establishing requirements is to collect relevant, sufficient, and accurate data so that a core set of requirements can be established. This core requirements set can then be used as a starting point for the visualization design process, by enabling the designer to first discover and define the right problem, and second, develop and deliver the right solution [40]. Requirements may be *functional*—requirements which describe product capabilities (the "doing" of a system), and *nonfunctional*—requirements which describe the characteristics and constraints for the product's behavior (the "being" of a system).

Techniques for Establishing Requirements. Three main techniques are used for gathering data when establishing requirements: interviews, questionnaires, and observation. Additional techniques include focus groups, studying documentation, researching similar products, and contextual inquiry—an approach which can be viewed as an unstructured interview taking place at the user's workplace, and during which the designer works as an apprentice to the user [3, 20]. Rogers et al. [42] provide a detailed description of each approach. Although these basic techniques form a small set, they are flexible and can be combined and extended in multiple ways. Rogers et al. [42] argue that an informal, open-ended interview is the best approach when aiming to gain first impressions about a problem or to explore issues—which is often the case in the domain characterization stage for problem-driven research. Indeed, open-ended interviewing is often used in the design of visualization tools, alongside observation, although a roadmap for this process has not been discussed or proposed before.

Tasks and Activity Terminology. As noted by Munzner [38], the word task is overloaded in the visualization literature, where it has been used at multiple levels of abstraction and granularity. In some works [38,44] the terms *problem, operation*, and *task* are used to denote respectively: a task described in domain terms, an abstract task, and a task that crosscuts these two levels.

This work uses the activity-centered design definitions: with increasing granularity, users have activities (problems) and tasks [40]. An activity is a high-level structure such as "go shopping" or "understand the relationship between *E.coli* genomes", while a task is a lower-level component of an activity such as "drive to market", "find a shopping basket", "use a shopping list to guide the purchases", respectively "load the complete *E.coli* dataset (673 genomes)", "locate an ortholog cluster in the 673 genomes", "examine the gene neighborhood of the ortholog cluster" [2] etc. An activity is a collected set of tasks, but all performed together toward a common high-level goal. A task is an organized, cohesive set of operations directed toward a single, low-level goal.

Functional Specifications. *Functional specifications* (sometimes called specs) describe how a product will work as viewed entirely from the user's perspective. A functional specification describes features, and it does not concern itself with how the product is implemented. Functional specs should not be confused with a *technical specification*, which describes the internal implementation of the program (data structures, relational database models, choice of programming languages and tools, algorithms, etc.) [6]. Furthermore, functional specs are a different concept than formal specifications generated through a *functional programming paradigm* [4, 10, 26]. The functional programming paradigm is a style of building the structure and elements of computer programs that treats computation as the evaluation of mathematical functions. In contrast, functional specs capture what a system does (or its "function").

Domain Characterization Prerequisites. This work describes—from a visualization design perspective—a formal script and model for guiding the requirements session. This work does not address the key issues that are prerequisites for any data gathering session: goal setting, identifying participants, or the relationship between the interviewer and the data provider; these issues have been previously discussed in the existing literature [8, 15, 44].

3 Network Model

Given the usually ill-defined nature of problems tackled in visualization research, domain characterization is arguably best served by an informal, open-ended interview approach [48]. Such interviews generate

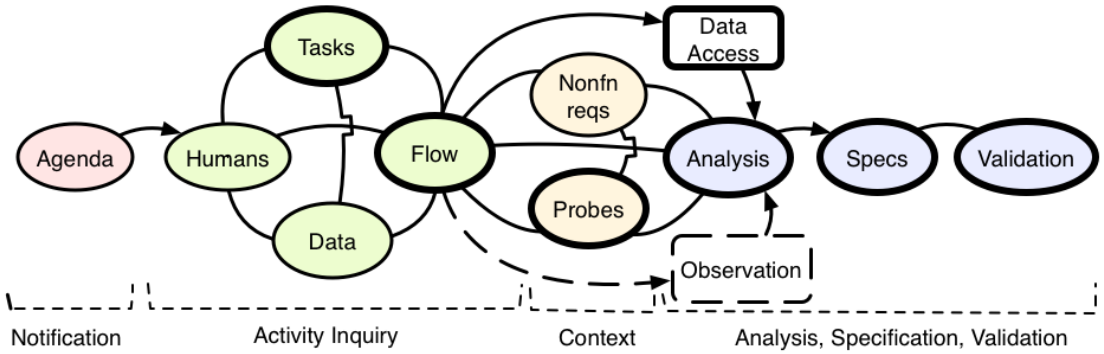

Fig. 1. Activity-centered network model for domain characterization in problem-driven scientific visualization. The model has four chronological steps, color-mapped in the figure. Critical nodes (Tasks, Flow, Probes, Data Access etc.) are heavier outlined; these components can act as gates in this model, and are sufficient reason to abort or postpone the prototyping of a project. Dashed nodes are optional. In this network model, arrows indicate unidirectional flow, while arcs indicate bidirectional flow.

rich information that is often interrelated and complex—in other words, information that gives a deep understanding of the topic. The model proposed in this work follows an open-ended interview approach.

This section defines a first model for domain characterization. The model can serve as a basic script to the main topics to be covered while engineering requirements. This model is based on the activity-centered design paradigm [40], which is an enhancement of the human-centered design paradigm. The paradigm is supported and reinforced by the collective Software Engineering Body of Knowledge [6], where emphasis in the software design and development process has been placed, for at least two decades, on the *user point of view* of the *functioning* of a software system.

This activity-centered model (Fig. 1) is organized in four chronological steps: 1) Notification; 2) Activity Inquiry; 3) Establishing Context; 4) Analysis, Specification and Validation. The overall four-step structure, and in particular the emphasis on steps 1 and 4, are adapted from the software engineering literature. However, this work develops and specifies steps 2 and 3—the core of the activity-centered model presented here—beyond the high-level software engineering framework, for the specific purpose of problem-driven visualization design. Step 4 marks a critique and an extension of the nested visualization design model, through the integration of functional specifications. The sections below describe each component of this model, and conclude with a summary of the model and its output.

3.1 Notification

Why: The primary purpose of this stage is to allow the domain experts time to prepare for the tasks and workflow component of the activity-centered model. The notification further allows the designer to set the context for the session, and to make the most of the limited availability of experts.

How: The notification is emailed in advance. In the activity-design experience, notifications issued at least a few days and up to a week in advance give the most effective results. The notification specifies how the interview will be run, who will participate, how long it will take to complete (a typical interview lasts 50-60 minutes), and the agenda for the requirements session. In accordance with the activity paradigm, the notification explicitly asks the users to think of twenty example tasks, as well as scenarios, to share during the meeting (Sec. 3.2.2). The request is accompanied by examples of correct and incorrect task and flow granularity. At the start of the meeting, introductions and a short review of the agenda are appropriate. The interviewer also lets the users know s/he will be taking notes; taping may seem efficient, but it is usually not [15].

Output: The result of this component is an agenda for the session. On the user side, the users are, ideally, brainstorming about example tasks and workflows.

Model Perspective: From the activity-centered perspective, the notification explicitly asks the users to think of example activities to share during the meeting. The users will typically keep the request in mind and refine their response during their daily workflow, in the context of their own work.

3.2 Activity Inquiry

The second step, specified here for visualization design, revolves around four core components: Humans, Tasks, Data, Flow. The components are interconnected and feeding each other. While the conversation may flow freely along new lines of inquiry that were not anticipated, the interviewer needs to make sure that questions along these four topics are asked, and that the answers are mapped to each component.

The Humans component describes the overall potential use of the system, and originates from the human-centered roots of visualization design [27, 52, 54, 57]. Because of the experts' familiarity with the default concept of "users", the activity model uses this component for the warmup, at the beginning of the requirements process.

The Tasks and Data components are present in most visualization design models [9, 38, 41, 48, 53]. In a departure from existing models, the activity-centered model proposed here ranks Tasks before Data. Tasks are a core component of the activity-centered paradigm, and they constitute the central, focal point of the resulting model.

In further accordance with the activity framework, the model explicitly incorporates user workflows into the design process. The Flow component describes the flow of data among tasks, as well as the mapping of data and users to the tasks. The Flow component is explicitly included in this model because of its unparalleled potential to reveal previously ignored tasks and data, hidden data sources, unusual user mappings to data and tasks, as well as the sequencing of tasks along activities. The Flow can be further used to generate functional specifications, as well as later in the design process, during the user evaluation of a design. The sections below describe each of these four components.

3.2.1 Humans

Why: The Humans component of the model facilitates a soft warmup of the conversation. The information gathered in these first minutes of conversation also helps the designer better understand both the tasks and data, and the potential impact of the project.

How: Questions asked as part of this component include: "How many users would use this visualization?" "Are the users part of the interviewee's group, other groups on site, other groups at remote locations?", "What background do these users have?", "How frequently would these people use the visualization?", "For how long?", "In what setting—in the lab, at home, during group meetings, anywhere?". General information about current software and the hardware the visualization will be used on is also helpful, in particular with respect to screen size, or usage in a location with no networking.

Output: The result of this component is an itemized list of project and user characteristics: how many users there are, what type of background they have, who will be performing the activities, how often, for how long, and in what setting.
Model Perspective: The visualization methodology is often human-centric or user-centric. From this perspective, the impact of a project has been measured, in the past, by the number of users [54]. From the same perspective, frequent use that is central to the users' work is desirable.

In the activity-centered approach, however, the user activities take precedence: a project commissioned by the two researchers who will find a cure for Alzheimer's Disease is just as important as a social media project with thousands of users—because the activity and goals in the first project are so important. Similarly, in the activity-centered approach, the value of a visualization tool to the users may not relate to its frequency of use: an important, yearly activity which lasts for days and leaves participants dreading the next year's activity can also serve as a strong project motivator [12]. Overall, potential impact in the activity-centered model weighs in both activities and humans.

3.2.2 Tasks

Why: The Tasks component is the core of the model, and, along with the Flow, implements most closely the activity-centered aspect of this model. The purpose of the Tasks component is to reveal the user activities, which are composed of such tasks. As part of this process, the interviewer aims to steer the interviewees into using examples rather than abstractions, and to reveal requirements that users may have difficulty recalling outside of their work environment. Real or made up examples capture both user tasks and the context in which the work will be performed.
How: A first question used to jumpstart the Tasks conversation is: "Give me the twenty most important questions you would like to ask from your data visualization system" [50] or, alternatively, "What are twenty example tasks that you would complete using this system?". In the design experience, twenty is about the right number: when asked for two to five examples, users tend to resort to abstract goals, which are inherently ill-defined: e.g., "Identify the link between gene mutations and molecular function" [29], or "Identify regions of error in turbulent flow" [32]. In contrast, requiring a higher number of examples has the effect of steering the users into more precise task formulations. These precise task formulations anchor the requirements in the target domain, and also reveal the actual data: "Locate an *ortholog cluster* in the 673 *genomes*", "Examine the *8-gene neighborhood* of the ortholog cluster" [2] etc. Because most users can spontaneously provide at most seven or eight example tasks, it is important that the agenda notification specifically ask for the twenty examples in advance (Sec. 3.1). A one week advance notification provides the experts with time to reflect on their needs, observe their workflow, and then expand, refine and organize their set of example tasks.

If a task description is too abstract or high-level, the interviewer asks "How would you go about doing that?", establishing a back and forth with the Flow component of the model, as appropriate. Users typically address the normal, most typical tasks first, in which no errors or variations occur. If not, some gentle steering may be required. Once a task list of reasonable length is compiled, the users are prompted to prioritize these tasks, in the order of importance and significance to their work. This prioritization can be later used when planning prototypes, as well as during evaluation.

As the users generate the list of tasks and the designers record it, the designers also generate a separate list of data entities which were mentioned in the tasks. Compiling the data list at this point helps steer the conversation into the Data component. Users may also describe simulated uses of the system by recalling or imagining stories, episodes, or scenarios that they have experienced. These stories and scenarios feed the Flow component.
Output: The output of this component is a prioritized list of tasks, as well as a preliminary list of data entities, and, potentially, a preliminary list of workflows and scenarios. The tasks may have been preliminary abstracted into visualization terms, with great care towards clarifying terminology and establishing a common vocabulary.
Model Perspective: The Tasks component reveals the user activities, at the right granularity level. Unlike the general activity-centered paradigm, in which focusing on tasks is considered too limiting [40], in this model of domain characterization the tasks (as opposed to activities) are used to reveal the underlying, precise requirements of the application domain. Each user task is a stereotypical description, written in text form, of the usage of the system to complete a task. Granular tasks can later be grouped in activities and goals during the requirements analysis step.

The Tasks component is a two-way process, in which designers and users together may refine the tasks using visualization specific vocabulary ("locate", "search", "compare" etc.) [7, 39, 43], as well as clarify terms. The conversation can thus be used to check the validity of preliminary abstractions and translations into visualization terminology.

Not all tasks associated with the domain problem can be explored in a single meeting; the set of tasks is typically iteratively refined over days and weeks. However, the prioritized set of tasks identified in this section can serve to guide the initial abstraction and encoding stages of design.

3.2.3 Data

Why: The Data component of the model is tightly linked to the Tasks component, and is explicitly included in this model because of its importance to visualization taxonomies. Visualization taxonomies, in turn, play an important role in the abstraction and encoding stages of visualization design. Last but not least, data influence the nonfunctional requirements of the activity-centered model, in the form of data constraints which affect the visualization solution.
How: During this section, the designer revisits the list of data entities compiled during the Tasks description, and attempts to clarify for each entity: 1) the nature of the entity—e.g., spatial (featuring intrinsic spatial coordinates) or nonspatial; 2) the entity type and attributes; 3) the size and amounts, and the streaming rates for streaming data; 4) the data source and potential sources of data error, and 5) the data format. Documenting counts, sizes, streaming rates, and validity of the data is, in particular, a crucial part of the process that is often overlooked. The interviewer also checks the number of data instances involved in a particular task—one, some, many, all; doing so further clarifies the translation of tasks into visualization taxonomies [39]. The conversation can also be used to check the validity of potential abstractions (for example, the validity of representing the data as a graph structure), while taking care to also clarify the meaning of those visualization terms that are unfamiliar to the users.
Output: The result of this component is a refined list of data entities, along with the characteristics of each data entity. The data may be already abstracted into a visualization taxonomy.
Model Perspective: Data is central to the visualization field, where it guides the taxonomy of visual representations. In the activity-centered model, data is nevertheless secondary to the user tasks. In the activity-centered approach, the inquiry does not start with a discussion of data instead of a discussion of tasks: the data inquiry follows the task discussion.

3.2.4 Flow

Why: The Flow component is an explicit request for existing workflows, or ideal workflows (how would the users go about solving a practical problem, ideally?), using the Tasks and Data previously identified. The Flow component often reveals the sequence of tasks, as well as the manner in which small tasks are sometimes combined in larger activities. Because user workflows go beyond task sequencing, the Flow component further provides an opportunity to identify: 1) cycles and frequency of tasks; 2) the data sources, and 3) the humans involved at each workflow step. In practice, the Flow component is an opportunity for reflection, which often reveals that the users skipped an essential preparatory, intermediate, or analysis step in their previous Tasks list, or an additional data entity.
How: The interviewer prompts the user for example workflows and scenarios that involve the previously identified tasks and data. Domain

experts may not engage in the topics of task frequency, data sources, and humans without guidance; it is the designer's task to make sure these topics are discussed. As part of this process, the designer explicitly maps the Humans, Tasks and Data information to the activity Flow.

Data sample elicitation. As part of the Flow component, the designers mandatorily request at this point access to the data, or a data sample of sufficient size and complexity—to ensure that real data exists, it is accessible, and it is available for visualization.

Ethnographic observation request. If appropriate, the designers may also request and schedule at this point a follow-up observation session. The request is particularly important when the users' work environment plays an important role in a workflow, or when it is plausible, despite the notification period of self-observation, that what the users say they do is not necessarily what they do in practice [13, 42].

Output: The result of this section is a set of user-specified workflows and scenarios. The interviewer needs to make sure that each previously identified task is associated with at least one flow or scenario. A secondary result is a set of updates to the Task and Data lists previously determined.

Model Perspective: Overall, the workflows and scenarios revealed through this component can provide a basis for designing prototypes, and for developing case studies for evaluation. The Flow component lays thus the grounds for the abstraction and encoding stages of design, and also can guide their evaluation.

In the activity-centered model, it is important that the Tasks and Data discussion precedes the Flow: compiling a list of tasks first ensures the workflows are described at the right granularity. Furthermore, in view of future evaluation, each task should be associated with at least one workflow or scenario.

This model instantiation also explicitly highlights two important aspects of visualization design: the source of data featured in the user activities, and the environment in which the activities take place. The benefits of incorporating real and interesting data into visualization prototypes, as well as the pitfalls of not using real data have been well documented in the literature [27, 44]. As shown in the literature, designers ignore the data access request and followup step at their own peril.

3.3 Establishing the Problem Context

The third chronological stage of the model, also instantiated here for visualization design, covers the problem context. The context of the problem is not the same as the "context of use" referenced in human-centered design [27]. In this domain characterization model, the context follows the activity-centered paradigm and has two components: 1) nonfunctional requirements, which describe the user expectations regarding the operational environment of the project; and 2) a set of three-pronged probes to test the validity of engaging in the project, as well as the potential impact of the project.

3.3.1 Nonfunctional Requirements

Why: Unlike functional requirements, which describe the functioning of the product, and which can be captured via scenarios, nonfunctional requirements capture the user's perception of the "being" of the product. Nonfunctional requirements are as important as functional requirements to the success of a project. Such requirements go beyond usability and user experience goals: 1) Scalability influences more than just visual scalability (would we be able to store and process on the fly a petascale dataset? [28, 33]); 2) Privacy can affect the publication of a project—several interdisciplinary projects nearly-failed publication because data was revealed to be private, sensitive, or proprietary [23, 51]; 3) Certain scientific user groups have a strong preference for desktop applications (as opposed to web-based) [58], and other groups expect cross-platform compatibility [46]. All the factors above influence the design and outcome of a project.

How: The questions asked in this component are informed by the earlier-covered topics of Humans, Data, Tasks and Flow. The minimum of issues to be clarified in this section of the discussion—because of the impact they can have on the project design—typically need to include: scalability, privacy, accessibility, operating environment, preferred tools, and learnability. The questions can be organized along the HCI typology of nonfunctional requirements [42], which includes: 1) environmental requirements related to the social, technical, or organizational environment of the product, including computing requirements and run-times; 2) data-related requirements regarding the volatility of data, its values, size/amount, persistence, accuracy etc.; 3) the user characteristics; 4) usability goals such as learnability; and 5) user experience goals such as enjoyment.

Output: The result of this section is an itemized list of nonfunctional requirements, organized according to their type.

Model Perspective: Nonfunctional requirements affect the user activities and the overall project, although they are in general not directly captured by tasks, workflows, and scenarios. Because nonfunctional requirements are not explicitly covered by activities and scenarios, special care must be taken that these requirements are clarified during the requirements session. Nonfunctional requirements need to be explicitly captured in the requirements documentation, and later discussed in the project evaluation plans.

3.3.2 Probes

Why: The context probes capture, from an activity perspective, the validity of engaging in the project, as well as its potential impact.

How: This work groups the probes into three categories: rationale of visualization being necessary to solve the domain problem; critique of existing tools; and exploration of possibilities.

Visualization argumentation. The goal of this set of probes is to clarify whether visualization is appropriate for solving the domain problem. In particular, the users and designers brainstorm about whether the activities and tasks could be addressed algorithmically, or through straight data integration.

Critique of existing tools. The second set of probes discusses the benefits and shortcomings of existing approaches to the problem. The probes also seek to establish whether the interviewees are aware of existing tools. In the design experience, most scientific users are well versed in visualization toolkits, and quite articulate and vocal in their likes and dislikes [32]. However, an existing visual tool or representation may still turn out to be the right solution.

Possibilities exploration. The third and last set of probes is the most open-ended one, and seeks to explore other desirable features that the user did not previously consider. Examples include journaling, integrating additional data, a new interaction paradigm or technology etc. For example, the users may not have realized that pairwise visual comparison could simplify their workflow [34]; if the comparison option is not explored at this point, the project may require later significant redesign. Note, however, that in order to preserve trust in the collaboration, the open-ended features do not take precedence over the user-specified requirements.

Output: The result is a crisp rationale for the visualization approach, in the context of existing tools. A secondary result is a set of updates to the Tasks, Data, and Flow lists.

Model Perspective: The probes describe a side of the overall context of user activities that is complementary to nonfunctional requirements. Like nonfunctional requirements, these aspects do not feature explicitly in the user activities and scenarios, and they must be explicitly accounted for in the requirements documentation.

The probes signal the end of the interactive requirements session. In closing, the interviewer thanks the participants, describes the next steps, reiterates the request for data access and potential observation session, and asks for permission to ask follow up questions. It also helps to agree on a single point of contact on both the domain side and the designer side, to help avoid potential communication breakdowns.

3.4 Analysis, Specification, Validation

The essential last stage of the activity-centered model is the analysis of requirements, specification, and validation of the designer's understanding of the problem. This step can take days to weeks to complete, and includes the activity-centered concept of functional specifications.

3.4.1 Analysis

Why: While the domain characterization defined so far is grounded in the application domain, proceeding at this point directly towards ideation and prototyping is a costly mistake. The problem stems from the fact that the information flow so far has been almost exclusively from the domain expert towards the designer. There is no guarantee, so far, that the *designer's* understanding of the domain problem and user activities is either accurate or complete. Analysis, followed by specs, is a first step towards validating this understanding.
How: Interviews generate large amounts of information. While the information is still "fresh" in mind, designers analyze the information in their notes to uncover patterns and conflicts, and contact the users for clarifications. Conflicting nonfunctional requirements may be weighed against each other and reprioritized. Similar tasks and their workflows are then grouped into logical sets of functional requirements.

The Flow information may help the designer group similar workflows or subflows into scenarios and use cases. In this respect, Springmeyer et al. [48] show an example characterization of the scientific data analysis process in terms of functional requirements. Their work describes the process of extracting an operation taxonomy which is grounded in interviewing and observation. Variations on grouping techniques like Hierarchical Task Analysis [49] and Card Sorting [31] can also be useful at this stage.
Output: The result of analysis is a non-conflicting description of the project requirements, as well as a hierarchical organization of the user activities and tasks.
Model Perspective: The activity model explicitly partitions requirements into activity-related capabilities (functional requirements), and nonfunctional requirements. The analysis step groups the previously determined granular tasks and workflows into activities and goals, and weighs in the nonfunctional requirements. The original set of prioritized tasks is reflected in the resulting description.

3.4.2 Functional Specifications

Why: Functional specifications reflect the process of organizing into documentation the requirements analyzed above, from the user's point of view. In this respect, a functional specification describes how a product will work, entirely from the user's perspective. A functional specification describes features, and it does not concern itself with how the product is implemented. Functional specifications allow the users to validate the designers' understanding of the domain problem.
How: The requirements captured and analyzed so far are translated into a written functional specification document, often in the form of a collection of scenarios. The specification also documents the data attributes that each scenario must access.

A functional specification spells out the activity-related design goals, as the designer understands them from the established requirements. A specification also specifies nongoals, or what—from the established requirements—the designed product will **not** do. A specification document further includes a prioritized list of nonfunctional requirements. The functional specifications also include scenarios, as discussed in the Validation section below.
Output: The result is a specification document, which describes the designer's understanding of the user activities. Like requirements, specifications evolve over time, in an iterative process. Spolsky [47] provides a complete example of a functional specification, and this work includes a sample functional specification in the supplemental materials. Functional specifications are evolving documents, which get updated after each interview and as the project evolves.
Model Perspective: Specifications and requirements are two separate concepts. Requirements can be established through interviewing and observation of the users. In contrast, specifications reflect the *designer's* understanding of the user requirements, and are the result of a reasonably lengthy requirements analysis process. Specifications take the form of a written document, which is the result of thoughtful analysis off line, and which the users can review in a process that is in no way an interview or an observation. In a nutshell, requirements come from the user towards the designer, while functional specifications go in the opposite direction, from the designer towards the user (Fig. 2).

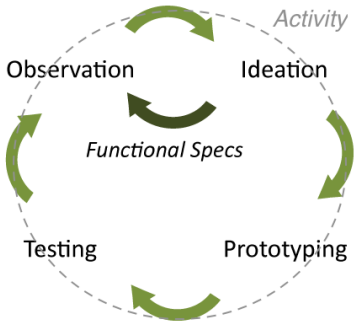

Fig. 2. Activity-Centered view of the HCD cycle (observation, ideation, prototyping, testing [40]). In HCD, users can confirm that the designer is solving the right problem after at least one full cycle iteration. The ACD functional specs are a backwards shortcut through the HCD loop, before the prototyping stage, that allows the user to confirm sooner the validity of the designer's problem characterization.

Specifications as a risk mitigation factor. Arguably, writing specifications constitutes the biggest risk mitigation factor available to visualization designers. Functional specifications can effectively ensure that the designers are not solving the wrong problem, as well as help the designers avoid situations where the way the data is shown does not fit correctly the user workflow—**before** the prototyping stage. In fact, in the absence of the activity paradigm and functional specs, prototyping is an early estimate: an operational prototype which can be tested (i.e., completing a full cycle through the human centered design loop) is typically necessary to answer such questions.

In the larger framework of software engineering, functional specs are ranked among the twelve steps essential to any product development process, right next to using source control [47]. Functional specs permit a rigorous assessment of requirements before design can begin, and reduce later redesign [6]. Designers can also use the specifications document as the basis for developing effective verification and validation plans. Similarly, functional specifications should be an essential component of problem-driven visualization design. In the activity-centered model, functional specifications capture the user activities determined during the requirements session, in the form of designer-written scenarios. Asking the user to review these scenarios is a unique opportunity to verify that the visualization designers are not solving the wrong problem.

3.4.3 Validation

Why: Specification validation ensures that the project is sufficiently specified to meet user needs, before ideation and prototyping commence. Validation further detects and corrects any unnecessary and incorrect requirements, and ensures that designers' understanding is consistent with the user needs.
How: Validation can be performed via peer review—a focused meeting in which a small number of stakeholders evaluates the requirements and specifications documentation to find errors and improve quality. Other validation approaches, such as acceptance tests, model walk-throughs, and operational prototypes are also possible (unlike exploratory prototypes, which clarify ambiguous requirements, operational prototypes typically implement functionality) [15].

As a rule of thumb, one can only inspect two to five pages of documentation in a few hours [15], which is part of the reason why interviewing and observation [21, 45] are not a good fit with specification validation. In the validation made possible by functional specifications, each inspector (including the author) prepares for the peer review by spending one to two hours examining the specifications. This is a critical part of the process; in software engineering, most of the errors are detected during individual preparation [15]. The functional specifications are then revised based on the user feedback.
Output: The result is a validated set of requirements and functional

specifications which characterize the domain problem.

Model Perspective: In theory, functional specifications should be easy to write and validate, in particular if the requirements collected earlier follow the activity-centered approach, and as such, are already described as scenarios. In practice, however, asking the users to validate specifications written in a formal format has a single, uniform effect [6]: users don't read the specifications.

To mitigate the risk of users not reading the specifications, this work recommends following the Spolsky advice for writing specifications: use interesting storytelling and entertaining language, while scrupulously preserving technical content. In this approach, a standard scenario starting with:

The user selects a biochemical model from the literature, and adds it as a new model entry with a single field "Model Name" into the visualization analysis system. The system is web-based.

becomes

Kermit the Frog, bored out of his mind, opens the latest issue of Nature Methods and spots a mouth-watering model of the fruit-fly response to allergens. Sticking his tongue out, Kermit runs to the browser, opens the visualization system, and types a new model entry with a single field called "Fruit-fly model".

The supplemental materials for this paper provide a full example of translating, for validation purposes, a few example use cases (resulting from the analysis of requirements for a biology visualization project [46]) into an example specification to be shared with the users.

From a practical standpoint, when following the approach above, the user response rate increases from approximately 10% to 90%; the user feedback also increases roughly by a factor of ten, not only in quantity but also in quality. Simply put: users read, pay attention to, and comment on entertaining specifications. In contrast, most users do not read formal specifications.

3.5 Model Output and Summary

When completed correctly, the activity-centered model provides: 1) a list of tasks and data entities; 2) a set of nonfunctional requirements; 3) an answer to the three context probes; 4) a set of scenarios—in the form of user-validated functional specifications—describing the activities possible using the visualization tool or technique; 5) a sample dataset of reasonable size and complexity.

The model is best described as a network, in that it is an interconnected set of components. Stage 1 (Notification) is a prerequisite for later stages; similarly, Stage 2 (Activity Inquiry) is a prerequisite for Stages 3 and 4, and Stage 4 (Analysis, Specification, and Validation) draws on the previous three stages. Stage 4 cannot be skipped. At the same time, the components within each stage depend on each other and feed each other.

Critical components (Tasks, Flow, Probes, Data Access etc.) can act as gates in this model, and are sufficient reason to abort or postpone the prototyping of a project. The Observation component, on the other hand, is begrudgingly optional: while user and task analysis methods are best carried out in the context of real work [52], it is not always possible to observe the users in their environment—for example, in highly secure or remote locations. In certain situations, unobtrusive contextual inquiry may also not be particularly helpful—for example, when the domain science requires years of training, laboratory experimental equipment, or proprietary packages.

The resulting set of tasks and data entities could potentially be, at the completion of the process, already abstracted into visualization terms. While it is advisable to explain and check with the users the validity-in-context of visual abstractions during the requirements session, this may not always be possible. In practice, certain abstractions keep getting refined through repeated iterations and the writing of the final report [44].

The preliminary set of validated requirements and functional specifications engineered through this model is now ready to be passed on, along with sample data, to the next visualization design stage. This concludes the description of the activity-centered model for domain characterization.

4 Evaluation

According to Karl Popper, a theory in the empirical sciences can never be proven, although it can be falsified [17]. With this observation in mind, in the visualization literature, a model or theory can be acceptably supported by as little as one to a few concrete examples coming from the experience of one to a few authors [27,38,41,44]. The present work takes supporting evidence a step further, by considering the impact of the activity-centered model on the success of 35 concrete short-term projects completed by young researchers, and by contrasting the findings against results on 40 short-term projects similarly completed, but under prior models. The evaluation is rounded by considering further evidence from reports in the literature.

4.1 Supporting Evidence

The two sets of projects considered here have been completed by young visualization researchers undergoing training in interdisciplinary visualization research. Arguably, these young researchers would be the first to benefit from a blueprint of the domain characterization process. Each visualization project required collaboration with domain experts from a variety of domains, from orthopedics to turbulent combustion (e.g. [1,18,30,33,34,51,61]), and was, in its first iteration, completed in under 3 months. For each of these projects (several of which have resulted in publications), this work considers success in terms of both novelty (defined as potential for publication) and, along Brooks's criteria [8], the domain experts' expressed interest in adopting the research result as a tool.

The first set of 40 of these projects operated under a generic agile software engineering process model, enhanced by the information visualization nested model [38] and pitfalls model [44]. Note that due to lengthy data learning, cleaning and preprocessing characteristic to scientific visualization, the agile process sometimes decayed into one iteration of the standard HCD loop. From these 40 projects, 10 had a successful outcome, or a 25% success rate. From the 10 successful projects, 2 were completed by dual-expertise researchers (with a degree in the problem domain). The remaining 8 benefited from collaboration with committed domain experts who agreed to weekly meetings with the designers. The remaining projects failed under a variety of factors, including data issues and miscommunication issues, despite several committed collaborators. The 25% success rate is significantly lower than the one reported for agile processes in software engineering [19].

The second set of 35 projects operated under the same agile process, enhanced by the nested model and by the Activity-Centered model described in this work. From this set, 22 projects had a successful outcome, or a 63% success rate. From these successful projects, 2 were completed by dual-expertise researchers, and 10 benefited from weekly meetings with committed domain experts. 10 other projects succeeded despite no weekly commitment from the experts. The 63% success rate over this second set is slightly higher than the one reported for agile processes in software engineering (58% for small projects) [19]. From the remaining unsuccessful projects, 9 were failed projects (due to: improperly executed Data Access component, skipped or poorly executed Functional Specs component, respectively client never reviewing the Specs), and 4 were partial failures (client success, but no novelty due to improperly executed Probes component). Notably, for each of the unsuccessful projects, their success could have been predicted right from the domain characterization stage. Note also that in both settings, a number of projects have failed despite committed collaborators.

These results lend support to the activity-based model. To a large extent, these results have helped motivate articulating the present work.

4.2 Fit with Existing Reports and Models

Agreement. Visualization models and reports [27,41,44,46,48,52] have previously noted, sometimes empirically, the benefits of including a task axis in the domain problem analysis, and paying attention to the user workflows.

In particular, Lloyd and Dykes [27] report how repeatedly drafting a scenario from the user requirements, and having it evaluated by the users, "showed this to be a fruitful exercise." The activity-centered

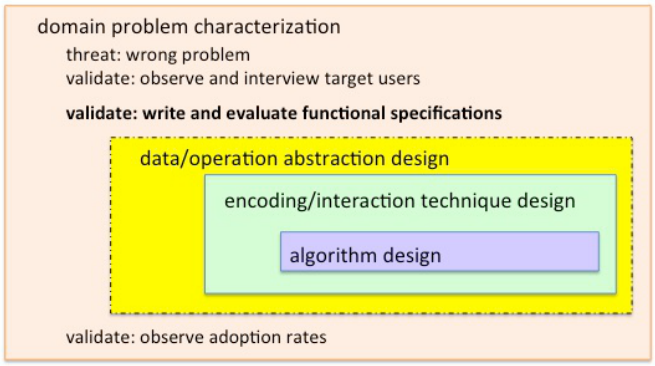

Fig. 3. Expanded Domain Characterization stage in the nested model. The updated stage model features a third validation scheme: validate: write and evaluate functional specifications. The nested stages of characterization and abstraction also cross over.

model offers an interpretation of why the scenario worked well in practice: in this case, although "too broadly-scoped", the scenario reflected the designers' understanding of the user activity. Furthermore, since the scenario was sent to the users for evaluation, it played the role of a partial functional specification, in its activity-centered meaning. Significantly, the report states that the users were able to make a total of over 300 suggestions after reading the scenario—a testament to the practical value of functional specs. The activity-centered model offers a theoretical framework for this reported case, and a roadmap for employing functional specifications systematically in the design process.

Aspects of the activity model are also reflected in the pitfalls model [44], although the latter does not necessarily provide actionable guidance on how to avoid those pitfalls. From a practical standpoint, the activity model combines well with their nine-level framework, and can be used to prevent pitfalls PF-4 ("no data"), PF-6, 7, and 8 ("not a visualization problem"), PF-9 ("existing tools"), and PF-10 ("users and tasks"). Alternatively, the activity-centered model could be used to analyze the first stages of the nine-level framework, as well as other different process approaches to domain characterization.

Certain aspects of the activity-centered model have also been captured elsewhere: for example, certain types of nonfunctional requirements and probes have been preliminarily considered by McKenna et al. [37], in the form of constraints, considerations, and opportunities. The activity-centered model offers a unified framework for the consistent and systematic treatment of these loose aspects of the domain characterization process.

The results reported here also support similar findings in the field regarding the role of active collaborators [22]. Furthermore, these collaborators may indeed have different roles, expertise and knowledge that apply during the analysis process [59], and may influence the design process repeatedly through repeated small exchanges of information [36].

Agreement and partial disagreement. Visualization models typically describe high-level frameworks; while other models focus on the design or evaluation stages. The activity-centered model focuses on, and goes deeper, into the domain problem and data characterization level briefly described by Munzner as part of a nested visualization design process [38]. However, the nested model's recommendation to the *wrong problem* threat is *validate: observe and interview target users*, and *validate: observe adoption rates*. This work adds to the nested model a third, essential validation scheme: *validate: write and evaluate functional specifications* (Fig. 3).

Furthermore, while the nested visualization design process separates the domain characterization from the abstraction stage, the activity model attempts abstracting the data and tasks as part of the requirements establishing process, and thus crosses, to some extent, the two levels. Munzner also suggests that perhaps collapsing the two stages into one level would be more apt, since "a characterization of a domain with no attempt at abstraction may not be very useful" [38]. Nevertheless, there is value in separating the abstraction stage, if only to make sure this stage is given adequate consideration during the design process.

Other existing design models [14, 24, 37, 45, 52, 57] are also derived from the HCI human-centered paradigm and miss the opportunities given by the activity paradigm.

5 Discussion

Visualization, HCI and software engineering research all report [6, 38, 40, 42, 44] that the hardest part of design is getting the requirements right, which means ensuring that the right problem is solved, as well as that the solution is appropriate. Defective requirements propagate into designs and technical solutions, into implementations, and into validation, with amplified effects. Moreover, domain characterization is particularly challenging in interdisciplinary visualization settings [27, 44, 55]: domain experts may use domain-specific jargon; may not be easily accessible for meetings, observation, or feedback; may not be clear about constraints surrounding the data; or may not have a clear picture of their needs. This section examines, in context, the merits and limitations of the activity-centered model.

5.1 Model Merits

Domain characterization model. The activity-centered frame and model provide a practical way of dealing with the complexity of domain characterizations and interdisciplinary communication, while reconciling several idiosyncrasies of the spatial and nonspatial design processes. Beyond problem-driven design, understanding the data and formulating correctly the tasks of interest on these data is relevant to domain-independent technique-driven approaches as well. In the long run, the activity-centered abstraction may help derive potential design patterns for abstracted tasks (e.g., [11]), and to create potentially reusable techniques in an extensible fashion.

Activity-centered framework. The model introduced here follows an activity-centered design paradigm. The emphasis on tasks and activities allows an explicit link between the requirements process and the abstraction stage, and through it, to the encoding stage of design. However, general activity-centered design recommends designing for activities: designing for tasks is deemed too restrictive [40]. In contrast, the work presented here uses tasks to reveal the underlying precise requirements of the application domain. This approach follows in this respect advice from the same design sources: "Requirements made in the abstract are invariably wrong" [40]. Focusing on detailed, rich, granular example tasks at this stage helps anchor the requirements into the target domain. The approach presented here then groups granular tasks in activities during the requirements analysis step. In a certain sense, this approach replicates, at the requirements level, the first component of the diamond design pattern advocated by the HCI community [40].

Functional specifications. The activity-centered framework further enables the validation of requirements from a functional perspective, at the task level, through functional specifications validated by the user. This intermediate checkpoint, before prototyping, can help ensure that the right problem is being solved. Functional specifications have the further potential to reduce the gap of understanding between designers and users [55], and thus can massively reduce the later redesign of visualization systems. If nothing else, analyzing the user requirements and writing the functional specifications should force both the designer and the technique-driven researcher to actually design their solution, and realize that the tasks and data characterization and their correct abstraction need to be properly addressed. Last but not least, the specifications document can be used as a basis for developing effective verification and validation plans.

General enumerations of domain characterization methods exist in both the HCI literature and in the software engineering literature. Yet generic approaches fail to yield adequate requirements for application visualization design [27], in addition to being biased towards user experience practitioners. Notably, while works in other graphics areas (e.g., sensing) occasionally employ functional specs within an HCD

process [5], in the style of software engineering, these works use the specs strictly as a preliminary blueprint for prototyping. Their spec is not used as a communication channel to the user, as recommended in the activity model, nor is it explicitly integrated with the HCD approach.

Roadmap. Engineering requirements does not last for a set number of hours or months and then finish [40, 42]: it is an iterative process in which activities inform and refine one another. In practice, requirements and specifications evolve and develop as the domain experts interact with visualization designs, and see what is possible and how certain features can help them. The requirements activity itself is repeatedly revisited. The model described here guides this iterative approach, and has the potential to help speed up and tighten up each iteration of the design loop.

Implementing the first iteration of the activity-centered model requires typically 60 minutes of face-to-face time with the domain experts, optionally followed by one or several observation sessions. Since initial meetings reported in the visualization literature last only a few hours, and occur in parallel with other projects [44], this model can provide a valuable roadmap for such meetings. Activity-Centered designers spend on average 10 minutes on the warm-up and Humans component, 25-30 minutes covering the Tasks, Data, and Flow components, and 10-15 minutes on Nonfunctional Requirements and Probes. Some collaborations require multiple interview and observation sessions; some interview sessions may be dedicated entirely to a Probes discussion about relevant tools etc.

Although presented in sequence in this paper, the model components are not crisply delineated in time: as indicated by the model figure, the conversation flows from one topic to another, and the different components reinforce each other. Nevertheless, the conversation notes are organized, for clarity and to enable later analysis, along the network model components. The supplemental materials include a typical set of requirements session notes, organized along these lines.

5.2 Assumptions and Limitations

This work assumes that appropriate stakeholders have been identified before an agenda is issued and requirements meetings take place. There are several materials available to guide the selection of relevant stakeholders, in both the HCI and software engineering literature, most notably Gottesdiener [15] and Rogers et al. [42]; as well as a discussion of the several types of stakeholders from a data visualization perspective [44, 59]. This work also assumes, based on experience, that the availability of stakeholders will become clear by the end of the first iteration of the domain characterization process; how long they take to respond to the functional specifications could be an indicator. In particular, busy experts make time for a collaboration they consider relevant to their work, even when their only availability turns out to be a weekly pre-surgery meeting at 5am.

This work also uses an "us" (visualization researchers) and "them" (users) discourse. This assumption builds on the depth and richness of modern science, in which expertise in both visualization and the domain science is unlikely, though not impossible. Other reports comment on the evolving and overlapping roles played by those who both consume and produce visualization [60] in certain domains. However, the activity-centered model proposed here bears relevance to at least one channel of discourse, even in such overlapping collaborations.

The activity-centered model builds on a combination of techniques, including interviewing, contextual inquiry, and researching and critiquing similar products. This is by no means the only way to approach the domain characterization and abstraction process. Other techniques for domain characterization exist, including focus groups, questionnaires, and studying documentation. In a wider sense, the activity-centered framework and model instantiation presented in this work are not the only possible approach to domain characterization.

In the spirit of functional specifications: the activity-centered model of domain characterization "is complete, to the best of my knowledge; but if I forgot something, please let me know" [47]. In particular, the assumption—based on practical experience and the software engineering literature—that all goals can be broken down into lower-level tasks may not always hold true.

6 Conclusion

This paper introduces and evaluates a novel, activity-centered framework to domain characterization for visualization design. The activity-centered frame enables a tight link between the domain characterization level and the abstraction level of the design process, as well as with its evaluation. This work provides a basic roadmap and agenda, in the form of a network model, for the domain characterization step. The activity-centered model assigns value to a visualization based on user activities; ranks user tasks before the user data; partitions requirements in activity-related capabilities and nonfunctional characteristics and constraints; and explicitly incorporates the user workflows into the requirements process. A further merit of this model is its explicit integration of functional specifications, a concept adapted from the software engineering literature, into the visualization design nested model. Functional specifications have the further potential to reduce the gap of understanding between designers and users, and thus reduce the later redesign of visualization systems. Using this model systematically can help remove a number of pitfalls which have been identified multiple times in the visualization design literature, including lack of real data and solving the wrong problem.

A potential major benefit of the activity-centered framework and model is that if the design requirements are consistent with their activities, users may tolerate complexity and the requirements to learn something new: "as long as the complexity and the new things to be learned feel appropriate to the task, they will feel natural and be viewed as reasonable" [40]. Designing for activity may improve the openness of users to novel, powerful visual encodings and interaction paradigms.

Acknowledgments

This work was supported in part by awards from the National Science Foundation (NSF CAREER IIS-1541277, CBET-1250171, DMS-1557559 and CNS-1625941) and from the National Institutes of Health (NCI-R01-CA214825). I thank the anonymous reviewers for their generous and valuable feedback, Torsten Moller, Andy Johnson, and David Laidlaw for their help while handling various aspects of the review process, the Electronic Visualization Laboratory for its support, and my collaborators and students for the awesome work that motivated the present work.

References

[1] J. Albrecht, R. Hwa, and G. E. Marai. The Chinese room: visualization and interaction to understand and correct ambiguous machine translation. In *Comput. Graph. Forum*, vol. 28, pp. 1047–1054, 2009.

[2] J. Aurisano, K. Reda, A. Johnson, G. E. Marai, and J. Leigh. BactoGeNIE: a large-scale comparative genome visualization for big displays. *BMC bioinformatics*, 16(11):S6, 2015.

[3] H. Beyer and K. Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc., 1998.

[4] R. Borgo, D. Duke, M. Wallace, and C. Runciman. Multi-cultural Visualization: How functional programming can enrich visualization (and vice versa). In *Proc. Vision, Modeling, and Vis.*, pp. 245–252, 2006.

[5] C. Bouras, V. Triantafillou, and T. Tsiatsos. A Framework for Intelligent Virtual Training Environment: The steps from specification to design. *Educational Technology & Society*, 5(4):11–26, 2002.

[6] P. Bourque and R. Fairley. *Guide to the Software Engineering Body of Knowledge: Version 3.0, ISO Technical Report 19759*. IEEE Comput. Society, 2014.

[7] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Trans. on Vis. and Comp. Graphics*, 19(12):2376–2385, 2013.

[8] F. P. Brooks, Jr. The Computer Scientist As Toolsmith II. *Commun. ACM*, 39(3):61–68, 1996.

[9] S. K. Card, J. D. Mackinlay, and B. Shneiderman, eds. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., 1999.

[10] J. R. Cordy and T. N. Graham. GVL: Visual specification of graphical output. *Journal of Visual Languages & Computing*, 3(1):25–47, 1992.

[11] B. Craft and P. Cairns. Beyond guidelines: what can we learn from the visual information seeking mantra? In *Information Vis. 2005. Proceedings. Ninth International Conference on*, pp. 110–118, 2005.

[12] V. Doshi, S. Tuteja, K. Bharadwaj, D. Tantillo, T. Marrinan, J. Patton, and G. E. Marai. StickySchedule: an interactive multi-user application for conference scheduling on large-scale shared displays. In *Proc. of 6th ACM International Symp. Pervasive Displays*, 2017.

[13] K. A. Ericsson and H. A. Simon. *Protocol Analysis: Verbal Reports as Data*. Cambridge, Mass. MIT Press, 1993.

[14] S. Goodwin, J. Dykes, S. Jones, I. Dillingham, G. Dove, A. Duffy, A. Kachkaev, A. Slingsby, and J. Wood. Creative User-Centered Visualization Design for Energy Analysts and Modelers. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2516–2525, 2013.

[15] E. Gottesdiener. *The Software Requirements Memory Jogger: A Desktop Guide to Help Software and Business Teams Develop and Manage Requirements*. GOAL/QPC (Growth Opportunity Alliance of Lawrence), 2009.

[16] D. Gotz and M. X. Zhou. Characterizing Users' Visual Analytic Activity for Insight Provenance. *Information Vis.*, 8(1):42–55, 2009.

[17] B. Gower. *Scientific Method: An Historical and Philosophical Introduction*. Psychology Press, 1997.

[18] M. A. Haque, W. Anderst, S. Tashman, and G. E. Marai. Hierarchical model-based tracking of cervical vertebrae from dynamic biplane radiographs. *Medical engineering & physics*, 35(7):994–1004, 2013.

[19] S. Hastie and S. Wojewoda. Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch. *Retrieved*, 1(15):2016, 2015.

[20] K. Holtzblatt and S. Jones. *Contextual Inquiry: A Participatory Technique for System Design*. Lawrence Erlbaum Associates, 1993.

[21] P. Isenberg, T. Zuk, C. Collins, and S. Carpendale. Grounded Evaluation of Information Visualization. In *Proc. 2008 Workshop BEyond Time and Errors: Novel evaLuation Methods for Information Visualization*, BELIV, pp. 6:1–6:8, 2008.

[22] R. M. Kirby and M. Meyer. Visualization collaborations: What works and why. *IEEE Computer Graphics and Applications*, 33(6):82–88, 2013.

[23] S. Kitsiou, M. Thomas, G. E. Marai, N. Maglaveras, G. Kondos, R. Arena, and B. Gerber. Development of an innovative mhealth platform for remote physical activity monitoring and health coaching of cardiac rehabilitation patients. In *Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on*, pp. 133–136, 2017.

[24] L. C. Koh, A. Slingsby, J. Dykes, and T. S. Kam. Developing and applying a user-centered model for the design and implementation of information visualization tools. In *Information Visualisation (IV), 2011 15th International Conf.*, pp. 90 – 95, 2011.

[25] A. Leontiev. *Activity, Consciousness and Personality*. Prentice Hall, 1978.

[26] S. Li, S. Dragicevic, F. A. Castro, M. Sester, S. Winter, A. Coltekin, C. Pettit, B. Jiang, J. Haworth, A. Stein, et al. Geospatial big data handling theory and methods: A review and research challenges. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:119–133, 2016.

[27] D. Lloyd and J. Dykes. Human-Centered Approaches in Geovisualization Design: Investigating Multiple Methods Through a Long-Term Case Study. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2498–2507, 2011.

[28] T. Luciani, B. Cherinka, D. Oliphant, S. Myers, W. M. Wood-Vasey, A. Labrinidis, and G. E. Marai. Large-Scale Overlays and Trends: Visually Mining, Panning and Zooming the Observable Universe. *IEEE Trans. Vis. Comput. Graph.*, 20(7):1048–1061, 2014.

[29] T. Luciani, J. Wenskovitch, K. Chen, D. Koes, T. Travers, and G. E. Marai. FixingTIM: interactive exploration of sequence and structural data to identify functional mutations in protein families. In *BMC proceedings*, vol. 8, 2014.

[30] C. Ma, T. Luciani, A. Terebus, J. Liang, and G. E. Marai. PRODIGEN: visualizing the probability landscape of stochastic gene regulatory networks in state and time space. *BMC bioinformatics*, 18(2):24, 2017.

[31] N. Maiden. Card Sorts to Acquire Requirements. *IEEE Software*, 26(3):85–86, 2009.

[32] G. E. Marai, T. Luciani, A. Maries, S. L. Yilmaz, and M. B. Nik. Visual Descriptors for Dense Tensor Fields in Computational Turbulent Combustion: A Case Study. *J. Imaging Sci. and Tech.*, 2016(1):1–11, 2016.

[33] A. Maries, T. Luciani, P. H. Pisciuneri, M. B. Nik, S. L. Yilmaz, P. Givi, and G. E. Marai. A clustering method for identifying regions of interest in turbulent combustion tensor fields. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*, pp. 323–338. Springer, 2015.

[34] A. Maries, N. Mays, M. Hunt, K. F. Wong, W. Layton, R. Boudreau, C. Rosano, and G. E. Marai. Grace: A visual comparison framework for integrated spatial and non-spatial geriatric data. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2916–2925, 2013.

[35] B. H. McCormick. Visualization in scientific computing. *Comput. Graph.*, 21(6):1–14, 1987.

[36] N. McCurdy, J. Dykes, and M. Meyer. Action Design Research and Visualization Design. pp. 10–18, 2016.

[37] S. McKenna, D. Mazur, J. Agutter, and M. Meyer. Design Activity Framework for Visualization Design. *IEEE Trans. Vis. Comput. Graph.*, 2014.

[38] T. Munzner. A Nested Model for Visualization Design and Validation. *IEEE Trans. Vis. Comput. Graph.*, 15(6):921–928, 2009.

[39] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2014.

[40] D. A. Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, Inc., 2013.

[41] A. J. Pretorius and J. J. Van Wijk. What Does the User Want to See?: What Do the Data Want to Be? *Information Vis.*, 8(3):153–166, 2009.

[42] Y. Rogers, H. Sharp, and J. Preece. *Interaction Design: Beyond Human - Comput. Interaction*. Interaction Design: Beyond Human - Comput. Interaction. Wiley, 2011.

[43] H.-J. Schulz, T. Nocke, M. Heitzler, and H. Schumann. A design space of visualization tasks. *IEEE Trans. on Vis. and Comp. Graphics*, 19(12):2366–2375, 2013.

[44] M. Sedlmair, M. Meyer, and T. Munzner. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2431–2440, 2012.

[45] B. Shneiderman and C. Plaisant. Strategies for Evaluating Information Visualization Tools: Multi-dimensional In-depth Long-term Case Studies. In *Proc. 2006 AVI Workshop BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV, pp. 1–7, 2006.

[46] A. M. Smith, W. Xu, Y. Sun, J. R. Faeder, and G. E. Marai. RuleBender: Integrated modeling, simulation and Visualization for rule-based intracellular biochemistry. *BMC Bioinformatics*, 13(S-8):S3, 2012.

[47] J. Spolsky. The Joel Test: 12 Steps to Better Code. Joel on Software, https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/, 2000.

[48] R. R. Springmeyer, M. M. Blattner, and N. L. Max. A Characterization of the Scientific Data Analysis Process. In *Proc. 3rd Conf. Vis. '92*, VIS, pp. 235–242, 1992.

[49] N. A. Stanton, P. M. Salmon, G. H. Walker, C. Baber, and D. P. Jenkins. *Human Factors Methods: A Practical Guide for Engineering And Design*. Ashgate Publishing Company, 2006.

[50] A. S. Szalay. The Sloan Digital Sky Survey and Beyond. *SIGMOD Rec., Tribute to Jim Gray*, 37(2):61–66, 2008.

[51] M. Thomas, T. Kanampallil, J. Abraham, and G. E. Marai. Echo: A large display interactive visualization of icu data for effective care handoffs. In *The 8th IEEE Workshop on Visual Analytics in Healthcare VAHC'17*, pp. 1–8, 2017.

[52] M. Tory and T. Möller. Human Factors in Visualization Research. *IEEE Trans. Vis. Comput. Graph.*, 10(1):72–84, 2004.

[53] M. Tory and T. Moller. Rethinking Visualization: A High-Level Taxonomy. In *Proc. IEEE Symp. Information Vis.*, InfoVis, pp. 151–158, 2004.

[54] J. J. Van Wijk. The value of visualization. In *Visualization, 2005. VIS 05. IEEE*, pp. 79–86, 2005.

[55] J. J. Van Wijk. Bridging the gaps. *IEEE Computer Graphics and Applications*, 26(6), 2006.

[56] L. Vygotsky. *Thought and Language*. MIT Press, 1926.

[57] I. Wassink, O. Kulyk, D. E. van Dijk, G. van der Veer, and D. P. van der Vet. Applying a user-centred approach to interactive visualization design. In *Trends in Interactive Visualization*, Advanced Information and Knowledge Processing. Springer Verlag, 2008.

[58] J. E. Wenskovitch, L. A. Harris, J.-J. Tapia, J. R. Faeder, and G. E. Marai. MOSBIE: a tool for comparison and analysis of rule-based biochemical models. *BMC bioinformatics*, 15(1):316, 2014.

[59] K. M. Winters, D. Lach, and J. B. Cushing. Considerations for characterizing domain problems. In *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, pp. 16–22, 2014.

[60] J. Wood, R. Beecham, and J. Dykes. Moving beyond sequential design: Reflections on a rich multi-channel approach to data visualization. *IEEE Trans. Vis. Comput. Graph.*, 20(12):2171–2180, 2014.

[61] W. Xu, A. M. Smith, J. R. Faeder, and G. E. Marai. RuleBender: a visual interface for rule-based modeling. *Bioinformatics*, 27(12):1721–1722, 2011.