UNIVERSITY OF CALIFORNIA
SANTA CRUZ

SOFTWARE INSTRUMENTS

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

DOCTOR OF PHILOSOPHY

in

COMPUTATIONAL MEDIA

by

Jasmine Tan Otto

December 2023

The Dissertation of Jasmine Tan Otto is approved:

Professor Noah Wardrip-Fruin, Chair

Assistant Professor Adam M. Smith

Doctor Scott Davidoff

Dean Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Software Instruments

by

Jasmine Tan Otto

Software instruments are intelligent interfaces used by domain experts to manipulate complex artifacts. By contributing the concept of software instruments, this thesis connects the field of data visualization to its relations in both artificial intelligence and computer-supported collaborative work. Software instruments arise from projects carried out by teams with complex datasets in aerospace and narrative design, whom I have worked closely with, and which are developed and documented in this thesis. They are a lens for other computational media practitioners to introspect about their own design processes and artifacts. I argue that the evaluation of software instruments occurs in discussion with expert stakeholders, adopting the lens of critical technical practice to avoid disciplinary silos and narrow strategies of evaluation.

For the shardfolk

and the ice-cousins.

Acknowledgments

I could not have done this work without my collaborators and labmates: the Design Reasoning Lab (DRL) of Adam Smith; the Data to Discovery program of Santiago Lombeyda, Hillary Mushkin, Maggie Hendrie, and Scott Davidoff; the Creative Coding Lab of Angus Forbes and Oskar Elek; our neighbors in the Expressive Intelligence Studio (EIS, read as 'ice'), the Interaction Dynamics lab, the Human-Robot Interaction Lab, the Social Emotional Technology Lab (SET), and the Misfit Lab; and certainly not without the Seabright Camerata, my home shard during the COVID-19 pandemic. My thanks to Isaac Karth, Melanie Dickinson, Max Kreminski, Nic Junius, Jason Grinblat, Cat Manning, Jacob Garbe, Stacey Mason, Kavi Duvvoori, Kyle Gonzalez, and Kate Compton for their participation in the reading group.

My gratitude to my parents, Helen Tan and Jim Otto, and to many more teachers over the years: a partial list includes Jan Verschelde, David Dumas, Lou Kauffman, Jerry Bona, Ramin Takloo-Bighash, Paul Malchow, Joel Brown, Barry Sinervo, and Ralph Cintron. My gratitude to the extended academic network of computational media (CM)-aligned researchers; this includes you who have found your way here, dear reader. My support to the wildcat strikers, who won a raise from the University of California twice in three years. My research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

I thank Barrett Anderson for his unwavering and unconditional support, despite that words are finite and insufficient.

Chapter 1

Introduction

The workplace is full of data, and there is more of it the more experts you have. To be precise, in any interdisciplinary context, there are more *domain logics* the more domain experts you have — and each domain has its own language that takes years to learn. Therefore I ask, how do these experts all work together in the complex sociotechnical context of the workplace? And how do intelligent user interfaces help them to do this?

I will connect disparate literatures by troubling the division between researcher and performer, designer and engineer, humanist and scientist. Software instruments can be played like a guitar but also examined like an oscilloscope. My main contribution in this thesis is the concept of *software instruments*: intelligent interfaces used by *domain experts* to manipulate complex datasets, yes, and spaces of possible artifacts alike. I will frame my own work with multiple teams in terms of software instruments, and demonstrate that all computational media practitioners can productively employ this terminology.

I will argue against narrow strategies of evaluation peculiar to engineering-specific disciplinary silos, and advocate an approach rooted in design theory instead. I will adopt instead the stance of critical technical practice, as realized through methodologies including data-first design studies. I will reflect on design study practices through the lens of boundary objects, allowing me to connect exploratory visualization practices with rigorous qualitative evaluations, achieved through deep discussions with expert stakeholders at theoretical sites accessible through the visualization artifact.

This thesis follows two threads: a theory thread where we define software instruments, its background in multiple literatures, and which includes a short paper about the nested design study method and boundary objects; and a practice thread containing two application papers. One describes the MarsIPAN schedule prototype being developed at the Jet Propulsion Lab (JPL) to support operations planning for NASA's upcoming Mars Sample Return mission. The other is an application paper describing the DendryScope playtesting tool, which combines a visual interface with a static analysis approach to support narrative designers creating interactive digital narratives.

1.1 Motivation

Nobody knows how data visualization works. Some is good and some is bad, but what perceptual metric makes it work? I will argue that they work because blackboards work: embodied enactive learning can occur between experts and non-experts only when the system of marks on the board corresponds to a domain logic and operations therein. Take a contrived example: when a geologist and an urban planner look at the same map, we may imagine that one will describe boundaries in terms of chert and karst, while the other will describe boundaries in terms of urban and industrial.

Although these experts refer in either case to the same plots of land, they do not necessarily share a system of references in which to discuss those plots. If a map seeks to encode all the features of a territory, then which set of features is correct and congruent with itself? Susan Leigh Star's theory of **boundary objects** gets us out of this jam [124, 72]: in practice, both experts can read a map, so they need not agree perfectly on what 'ought to be' in order to talk about *what is there.*

Susan Leigh Star described both ontologies and artifacts encoding them as *boundary objects.* Early on, she situated her work in the terms of symbolic AI [123], but the concept was taken up primarily by science and technology studies (STS), information systems [75], and design theory [126].

When experts draw upon their 'domain knowledge' [102], they perform reflective design [111], interpreting the referent of the boundary object (or other plot) in order to produce a claim, a question, or another well-reasoned utterance [96]. Boden identifies this creative process as *reflection-in-action* [13], which may take place on whiteboards, in code bases, or in other contexts of sketching, writing, and conversation.

1.2 Research questions and claimed contributions

This thesis straddles three widely divergent fields of research, and as such asks a question in three parts - taking on the perspectives of human-computer interfaces (HCI), symbolic methods in artificial intelligence (AI), and computer-supported collaborative work (CSCW) respectively:

1. **HCI**: What are the properties of sociotechnical systems, paper tools, and creativity support tools (CST) that make them useful to domain experts who need to share out data?

2. **AI**: What are the properties of programming systems, domain-specific languages, and knowledge representations that make them useful to domain experts who need to reason over data?

3. **CSCW**: How are experts' tools and systems shaped by power, resources, and agendas within an organization? Where do language-boundaries become reified into tool-boundaries?

I have sought to address these questions using the *design study methodology* [113, 82]. I write as a practitioner engaged in critical reflection, and as a scholar who intends to make these fields engage with each other more deeply. As such, this thesis presents three research contributions: *a definition of software instruments* (Ch.2), with specific application to data visualization theory (Ch.3), and *two visualization systems* produced in conversation with aerospace mission operators (Ch.4) and with narrative designers (Ch.5).

1.2.1 Definition of software instruments

In chapter 2, I will define *software instruments*: they are boundary objects used by domain experts to create or analyze domain artifacts, and to communicate their insights with non-experts. Software instruments have been identified in different ways by different domains: existing theories range from expert systems [37] and visualization transforms [24] to exploratory notebooks [66] and generative pipelines [28]. Therefore, it is necessary for a scholarly work to bridge between the rich AI and HCI literatures. Advances in thinking about interfaces must be brought to bear on intelligent reasoning systems, and vice versa.

1.2.2 Using boundary objects to understand visualization methods

In chapter 3, I will sketch an emerging theory of data visualization that encompasses all four roles performed by visualization practitioners: stakeholders, domain experts, designers, and developers. Some visualization theories seek to maximize perceptual goodness (given a chart type), using constrained optimization. Others emphasize building up a picture of stakeholders' ontologies (given a community), using constructive grounded theory. I will argue that what each visualization method does, by producing intermediate boundary objects (such as exploratory prototypes, or tasks gathered from interviews), is mitigate threats to the visualization system at multiple nested levels identified by Munzner [89]. As such, visualization practitioners' expertise extends beyond their own programming systems and their ability to solve data problems; broadly construed, it consists of establishing lines of communication in the existing sociotechnical ecosystem.

1.2.3 Software instruments contributed to practitioner communities

In chapters 4 and 5, I will present two data-driven design studies performed with expert stakeholders in widely separated domains: relay pass allocation (Ch.4) and interactive narrative communities (Ch.5). I have produced software instruments that other people use as media artifacts, grounded in sophisticated symbolic reasoning systems building. The MarsIPAN schedule is used by operations planners on Mars Sample Return to diagnose the root causes of lost operational efficiency, whilst the DendryScope skein is used by narrative designers to playtest choice-based narratives using a static analysis tool.

1.3 Sociotechnical context of this research

I am a researcher in human-centered computing, which has recently enjoyed an explosion of intelligent systems interface design, as the result of a hype cycle around creativity and AI. Let us not dwell on who funds this hype cycle; it is the same suspects as when cybernetics exploded in the ’70s out of Berkeley, CA anyway.

In this work, I want to chart a path forward for intelligent interface design by establishing firm correspondences between design theory and software engineering theory, i.e. producing a domain of critical technical study, as proposed by Phil Agre [2]. Like Susan Leigh Star in her life-long study of boundary objects [72], I am ultimately interested in how interfaces are used by experts to communicate across differences in domain terminology and lenses of interpretation.

Part I

Theory

Chapter 2

Background and Ecology of Software Instruments

This chapter outlines concepts from three fields that I have drawn upon to define software instruments. From the perspective of a computational media practitioner, each field (for the purposes of caricature — these are not comprehensive takes!) has distinctly different priorities:

1. Computer-supported collaborative work (CSCW) [124, 65, 102] values novel theories of work produced in concert with expert stakeholders.

2. Human-computer interaction (HCI) [111, 134, 113] values novel interfaces which facilitate tasks performed by the system's users.

3. Artificial intelligence (AI) [125, 114, 2] values novel algorithms for gleaning important information from messy inputs or datasets.

As such, each concept from one field is typically in concordance with the other two, recognizing the same essential quality of a computational media artifact or system, but from a different point of view. In this chapter, I will construct the correspondence explicitly. After all, some adroitness with terminology will be very useful in describing software instruments. Organizations will have difficulty deploying this genre of media artifact without successfully describing both methods of engaging stakeholders in their creation, and methods for evaluating the results.

As researchers, we may for example consider systems and interfaces that act as **ready-to-hand tools**: these include paper tools (CSCW), creativity support tools (HCI), and interactive development environments (AI). Or, we may choose to consider systems and interfaces that **mediate communication** within a team of experts or any sociotechnical ecology: these are understood variously as boundary objects (CSCW), visualization systems (HCI), and generative pipelines (AI), within their respective fields. While each perspective carries with it a different set of methods, each set of these tools is another way to address the same critical need.

2.1 Definitions of software instruments

Software instruments are a *lens of interpretation* through which to read software systems that *produce media artifacts*. For instance, visualization systems transform datasets into graphical plots and interactive views. Likewise, programming systems transform codebases into executable scripts and applications. Software instruments are particularly interesting to study because, like all computational media, they are intermediate between intangible expert practices and objects in the world. As written programs made up of libraries and data structures, software instruments are more tangible than the cultural practices that make up *paper tools.* Yet, since their structure is independent of their physical hardware, the practical operation and existence of software instruments is harder to observe than that of either laboratory instruments or musical instruments.

In this work, I have synthesized insights from across multiple literatures with my own experiences as a computational media practitioner, and with those of expert stakeholders across disparate domains. As such, my description of software instruments draws upon research in STS to articulate how software instruments in particular (and data visualization methods in general) are a vital and already-existing technology of communication.

2.1.1 CSCW: Lens of collaboration

Design by experts is defined by Schön [111] as a process of *reflection-in-action*, meaning that it emerges through interactions with the object of study, with other experts, and with other stakeholders. In the case of experts in the workplace with data analysis needs, design processes are necessary to allow experts from diverse disciplinary backgrounds to successfully talk about the methods and results of these analyses at various levels of abstraction.

Domains are described by David Ribes [101] as theoretical sites and collections of richly-structured terminology, to which individual experts contribute by working together on problems. The domain provides a terminology which enables meaningful utterances to be made concisely, and serves as a lens of interpretation on other specific kinds of non-text artifacts. Ultimately, the logic of the domain is what enables collaborative practice between experts, but only if they all become sufficiently versed in the same domain.

Paper tools are figures used to perform computation, including Feynmann diagrams [25], box-and-stick diagrams of conceptual graphs [114], and arithmetic reckoning itself [65]. Klein describes paper tools as concordances between a symbolic algebra and its geometric representation, yielding a sign system in which constructive proofs can be worked on and shared with other experts.

2.1.2 HCI: Lens of visualization

Decision-making with data visualizations takes place, according to the work of Padilla et al., in a dual-process model [96]: the decision-maker first pattern-matches the visual to other graphics, and then grounds that understanding in a schematic description of the data represented by the visual. While pattern-matching is less accurate than symbolic reasoning, it is also much faster, which serves to bridge the *gap of evaluation* described by Schneiderman et al. [116].

Visualization prototypes are not only static mock-ups, and not only working examples of an interface, but typically part of a process leading from one step to the next [47]. In our terms, these artifacts represent an attempt to create a novel paper tool based on a certain domain-specific ontology. As boundary objects, these prototypes enable iterative collaboration with stakeholders and experts.

The **design study** is a method deeply based on prototype development [93], aimed at getting visualization developers to engage deeply with domain experts whilst avoiding common pitfalls of interdisciplinary collaboration [3]. The design study practitioner speaks regularly with expert stakeholders to solicit datasets, understand requirements, and ultimately build a system which supports a reasonable set of visualization tasks. By engaging with experts early and often, and rapidly creating prototypes to drive future conversations, these designers seek to bring their representation of the dataset in line with existing (paper) tools that experts understand, and tasks they deeply care about.

2.1.3 AI: Lens of intelligent system interfaces

Direct manipulation refers to a user interface where application state is 'bound to' graphical elements. Hutchins et al., writing in the 1980s, originally identified file systems and plots constructed in spreadsheets as examples [54]. Just as data visualizations bridge the gap of evaluation, so does direct manipulation bridge the *gap of execution* in the enactive loop between a software instrument and the expert. For example, a canvas in Photoshop admits many kinds of image manipulation by illustrators who skillfully and directly select areas on a raster grid.

Musical metacreation refers to the application of expert systems to composition and performance. Combining score generation with audio synthesis yields a genre of expressive software instruments, ranging from turnkey song generation [32] to livecoding performances [79] and any level of human-in-the-loop involvement in between.

Programming systems, as described by Jakubovic et al. [55], are a grounded description of software development practices and their remarkable flexibility. Programming systems encompass both the code editor per se, the package manager or development framework being employed, and other process-components which may or may not live inside the integrated development environment (IDE).

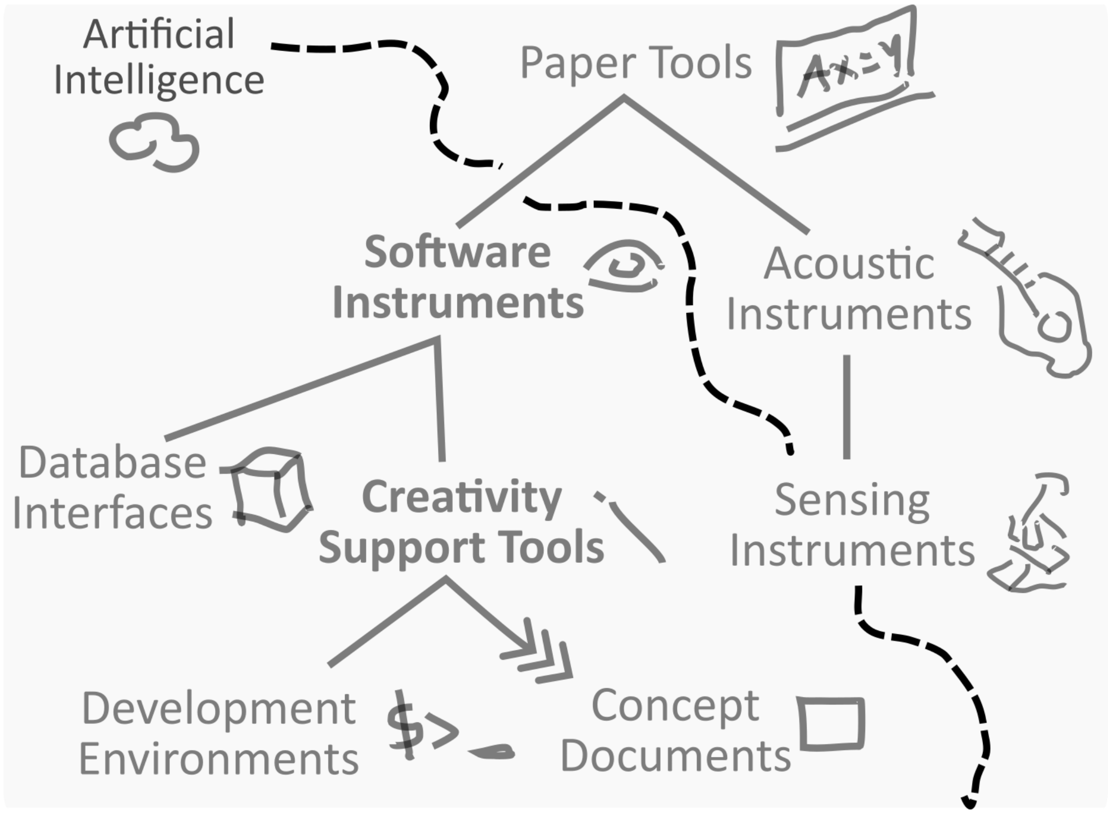


Figure 2.1: A graphical phylogeny of software instruments. AI divides visualization systems, IDEs, and other genres of software instrument (inside the nebulous cloud at the figure's center) from computational instruments like blackboards, guitars, and microscopes: that is, from hardware and from chalk-based representations.

2.2 A landscape of instruments and practitioner roles

We have seen that the academic fields of CSCW, HCI, and AI all seek to describe software instruments, using different terms. This is because software instruments are ubiquitous in expert practices, but are difficult to identify because they take on the shape of each field's respective theory. In this section, we describe many examples of software instruments 'in the wild' as a simple intuition pump.

In addition, we map out 2.1 a conceptual landscape containing all these various software instruments. Different computational media practitioners stand in different places on this landscape.

Physicists, engineers, and mathematicians participate in a blackboard culture [132], making extensive use of paper tools (such as Feynmann diagrams, or more conventional algebraic notations) to teach, to theorize, and to work out problems. 'Paper tools' give rise to all software tools, and therefore exist at the root: both predating them and serving often as prototypes.

Clinical cardiologists [94] use ultrasound machines to perform echocardiography, which (like other areas of medicine) requires precise interpretation of imprecise visualizations. 'Sensing instruments' therefore occupy the right-hand branch of our phylogeny, as they are no longer purely 'acoustic instruments', but typically hybridized with digital measurement and recording.

Business operators use graphical file systems and spreadsheets, which are interfaces refined since the days of Xerox PARC to databases of ordinary scale (not more than hundreds of files in a folder, or thousands of rows in a spreadsheet). The most sophisticated engineering practices in this regime are called data science, which has given rise to the exploratory notebook IDE (e.g. Jupyter, RStudio, Observable), vectorized libraries for simulations and analytics (e.g. Matlab, R, pandas), and tensor-based libraries for designing neural networks (go figure!). These together occupy the left-hand branch of our phylogeny as 'database interfaces' (when the ecosystem of extensions or libraries is not accessible to the end-user), or else the bottom-left branch.

Creative professionals, whose practices are less standardized, often require the most complex and bespoke software instruments. Digital illustrators, for example, have long since outgrown early applications of raster graphics to windowing systems [122], to such an extent that Adobe Creative Suite and many other commercial applications cater to them [73]. These *creativity support tools* are commonly employed in prototyping work (for example) to produce boundary objects such as GUI mockups, wireframes, and other conceptual artwork.

Live-coding practitioners turn signal processing into performances, constructing novel software instruments on the spot by using special-purpose programming systems which operate continuously in a read-evaluate-print loop (REPL) with real-time outputs. We position these as prototypical 'software instruments', the sibling of acoustic instruments.

From these five widely-spread examples, I invite the reader to envision their own favorite applications or videogames as software instruments in this landscape; and to consider what domain knowledge is encoded by each one, as well.

2.3 Vocabulary of software instruments

If design theory is at the root of HCI, then engineering methods are at the root of AI. But in order to understand the sociotechnical practices and impacts of software design, we will need a correspondence between the domain languages of design and engineering, such as CSCW offers in the form of *boundary objects*.

Of the four kinds of boundary object, observe that an artifact and a file are related (as a *coincident boundary* resembles a *standardized form*), just as an ontology and a database are isomorphic (as an *ideal type* resembles a *repository*). In Ch.3, we will construct further correspondences between design theory and software engineering via boundary objects, in the visualization domain. In the meantime, this section addresses non-vis methods by offering a domain-agnostic vocabulary of software instruments.

2.3.1 Artifacts and brushes

Software instruments can save and load *artifacts*. They allow users to act on these artifacts using *brushes*. In this section, we present a concordance between the terminology of artifacts and brushes, and the terminology of multiple scholarly disciplines.

An *artifact* is a structured object in the *expressive range* [121] of a certain software instrument. Mathematically speaking, the trivial software instrument is a structured editor, constrained to fields belonging to the appropriate data structure. In practice, editing fields in a structure is rarely the most intuitive and ergonomic way to manipulate a domain-specific language; therefore, other software instruments have come to exist.

If each noun refers to an artifact, then each verb refers to a *brush*. Similarly, a mathematician might describe elements equipped with operations. A programmer might describe objects equipped with methods. And an Inform 7 programmer might describe things equipped with verbs. (Haskell programmers might describe describe categories equipped with monads; they value compositionality and type-safety more than avoiding jargon.)

Hence, a *brush* is any operation that translates one artifact into a slightly different artifact. In parser fiction, player affordances are represented by 'verbs' which act on the state of the world, and may be parameterized by arguments (e.g. '[talk to galatea about] the gods'). In a text editor, the text-insertion 'brush' is parameterized by the cursor position, represented by a caret. In Paint or Photoshop, the ‘brushes’ which make selections and apply colors are parameterized by the pointer position, pointer pressure, and a great number of color pickers and sliders.

Let us return now to the expressive range, which models a design space as a high-dimensional vector space. To apply a brush once is to ‘travel to’ a nearby point in design space, defined relative to that brush’s ‘level of abstraction’. It turns out that experts can modify artifacts rapidly and fluently using appropriate brushes, as provided by creativity support tools and observed by Jingyi Li et al. [73]. Within this work, we will argue that both MarsIPAN schedule navigation (Ch.4) and DendryScope queries (Ch.5) constitute novel brushes, within their respective domains.

2.3.2 Decision support tools

In the context of data visualization, screenshots of visualization systems are artifacts. (After all, they are what goes in the presentation about the analysis results.) Therefore, analysis tools like selections, filters, colormaps, and other ways of manipulating view state all function as brushes.

As contributions to conferences on visualization or computer-human interaction, visualization systems can be evaluated in terms of how well the designers’ understanding of what space their users are exploring ‘matches up with’ the expressive range their system demonstrates.

2.3.3 Creativity support tools

Shneiderman et al. [116] introduced creativity support tools (CSTs), which Compton [28] extended by identifying design patterns in certain examples this genre. Compton's principle of **no wrong moves**, for example, may be understood in terms of brushes; it is the result of designing brushes that align with both the structural and semantic types of their underlying artifact. Compton identifies effective CSTs as tools which coordinate communities of learners and domain experts, by empowering them to share both artifacts and techniques with each other.

The evaluation of creativity support tool systems varies widely depending on what academic audience the system's designer wishes to address. Some venues value the usage and impacts of CSTs on communities of practice, emphasizing grounded theory methodologies; while others value the systemic description of creative practices that a CST represents, and identify this as research into the capabilities of certain kinds of AI systems.

2.3.4 Software instruments as tools of practice

In summary, software instruments serve the domain expert who wishes to integrate their practice, as mediated by paper tools and boundary objects, with existing databases and algorithms.

In Ch.3, we will elaborate upon those boundary objects which mediate the representation of a *database* and are produced by a visualization design study, including both the visualization system as such and its low-fidelity prototypes. Do not think of this 'database' strictly as an archive. For example, James Ryan [107] describes the *story* *volume* of a certain social simulation (or other narrative game [49]), as though it were a boundary object of the underlying model; so the story volume itself is now subject to strategies of representation (e.g. in Ch.5).

In Ch.4, we will take on the standpoint of operations experts on a joint aerospace mission. While it has been and remains possible to design operations schedules on paper, there are many upstream stakeholders and data sources (including orbital geometry, the condition of individual relay orbiters, various threat scenarios, and the status of the surface rover itself) which yield a complex set of hard and soft constraints. Teams are known to 'trade off' soft constraints — for example, by giving up some data volume on one day in exchange for getting it back at an opportune time. This leads to a combinatorial explosion of possible schedule designs, which in turn leads to data-driven schedule design systems such as MarsIPAN.

In Ch.5, we will take on the standpoint of narrative designers working in interactive fiction. While its consequence is apparently limited to the realm of art, their domain is simultaneously rich in software instruments (as the practice of videogame development) and rich in paper tools (as the practice of storytelling). I have built DendryScope, a static analysis system based on quality-based narrative (QBN) [67], a narrative design formalism, in order to reveal the context-dependent design intent underlying the straightforward database representation of QBN storylets.

Chapter 3

Visualization Artifacts are Boundary Objects

This chapter is based on a paper coauthored by myself and Scott Davidoff, currently under revision. In this chapter, "we" refers to the team of authors.

Despite 30+ years of academic practice, visualization still lacks an explanation of how and why it functions as a sense-making tool. We reflect on the notion of a *boundary object* as a potential theory to explain visualization practices. A boundary object is a dataset, visualization artifact, or other representation that bridges multiple domains. Using this theory, we integrate recent reflections on design work and collaboration with Munzner's nested model of visualization. Layers of the nested model are processes mediated by boundary objects, which in practice range widely: from expert interviews, to paper prototypes, to visualization transforms. Data visualization works because each layer of the nested model represents a collaborative meaning-making activity. If visualization needs a scientific theory, then it must be a social theory, as datasets are tools of communication.

3.1 Introduction

From color theory[17] to visual grammar[11] and graphical perception[27], much of the theoretical work in visualization has generally served in an instrumental capacity[99, 133], helping visualization designers to predict the effectiveness of a given visualization [74] to encode specific aspects of data. Yet this theoretical foundation seems to be absent of some basic explanatory power that accompanies theories in other disciplines: we don't have a theory that can adequately explain *why visualization works at all.* In this chapter, we seek to bridge the theoretical gap, bringing well-established social-scientific work into correspondence with visualization design study approaches (Section 2).

For its explanatory power, we introduce the concept of the *boundary object* to visualization. This terminology originates from the organizational ethnologists Star and Griesemer, who characterized differences in language between individuals, groups, and institutions that caused failures of communication. Star characterizes some failures as 'double-binds', where the same utterance had two different meanings as understood at two different *levels of abstraction* by two different groups [72]. They defined boundary objects as "representational artifacts and associated ideas that enable design knowledge to be transferred between social worlds and that facilitate the alignment of their interests [124]".

Boundary objects have proven to be a useful theoretical construct to describe the functioning of critical design artifacts across disciplines in disciplines that include

Human-Computer Interaction [75] and Organizational Management [20]. In this chapter, we look to integrate the idea of the boundary object into data visualization research. In particular, we argue that data visualization works because it functions in effect as a kind boundary object and that the artifacts produced by visualization work are successful in so far as they satisfy the principles that characterize *boundary objects* (Section 3).

Although visualization theory already seeks to describe how visualization systems empower users in different roles, it does not adequately address *communication between different roles*, and the privileged role of visualization system developers as mediators of this process. This chapter represents an effort to describe practices reported in the visualization literature, so that their successes can be transmitted through the theory, and connected with related efforts in computer-supported collaborative work (CSCW). To this end:

- We present a theory of visualization connected with literature from design, science and technology (STS), and organizational studies.
- We consider examples where contemporary practitioners used boundary objects to explain their visualization impacts.

3.2 Background

The background of this chapter is concerned with prior developments in visualization theory, driven by advances in visualization practice. Despite a turn to *second-order thinking* about visualization systems within specific design studies, visualization theory yet remains limited by an atomized view of the visualization system, neglecting its function in an organizational context.

In 2017, Chen et al. [25] wrote that “theoretical research activities in visualization are sparse.” This observation comes in the wake of instrumental theories of visualization, such as Cleveland and McGill’s 1984 work on perceptual tasks [27] (which advocated for the discontinuation of pie charts); MacKinlay’s metrics for expressiveness and effectiveness in visualization grammars [74]; and Brewer’s guidelines for the use of color in maps [17].

Purchase et al. in 2008 [99] seek to root information visualization in linguistic theory rather than design theory, arriving at similar insights: meaning in visualization is not fixed (“language is a process”), and an internal data representation is manipulated through an external form chosen by the designer. Their conclusions regarding visualization transforms are especially insightful. However, they do not discuss stakeholder interviews in any great depth, as Diehl et al. seek to do in adapting grounded theory methods to visualization [34].

Six years after Chen et al., visualization theory still lacks a shared framework across visualization’s incredibly broad community of research and their respective workflows. This state of affairs is not a failure; it points at the field’s richness and depth of interdisciplinary collaboration. Correll’s 2022 survey [30], in discussing the growing pains of visualization theory, argues that its focus is expanding beyond individual visualization artifacts to consider the entire workflow.

A report on the 2016 and 2017 Vis4DH workshops [16] invokes the possibility of a visualization theory beyond scientific theory: “humanities work is written not to be reproduced but to be followed, understood, and questioned”. In this spirit, we seek to recast design studies in terms of boundary objects, in order to draw out the mutually supportive nature of design practice and technical research.

3.2.1 Visualization Practitioners Hold Many Roles

Currently, visualization theory speaks to problems faced within specific roles held by probelm-domain experts. Like Hinrichs et al. [51], we see the diversity of visualization practices as a great strength; we also see explicit coordination between *different domain logics within visualization* (e.g. interface design, data science, cartography, language design, etc.) as a necessity.

Following Meyer et al., we observe that the complexity of combining these various practices is either mediated by *blocks and guidelines* [83], where blocks are design choices, and guidelines are domain knowledge of visualization. In larger collaborations, where these roles are taken on by different practitioners, design choices must be externalized, and thereby take the form of boundary artifacts.

Visualization practitioner roles

Although other taxonomies of roles in visualization practice are possible, we will use these four throughout the chapter. In prior visualization theory, we usually see one or two of these roles is the primary audience being addressed (Table 1):

- Design practitioners: Recruiting, interviewing, and understanding the needs of organizational stakeholders who own large or complex data.
- Data analysts: Understanding how to prepare the data, what capabilities exist in current data structures and algorithms, and communicating with visualization

designers how their goals can be achieved.

- Domain stakeholders: Performing visualization tasks on data, which can include justifying their choice of abstraction, acquiring decisionable insights, and coordinating cross-disciplinary teams.
- Visualization researchers: Validating visualization systems produced by the team. Was the system successful? What changes does it suggest to visualization theory?

	Interviews	Prototypes	Conceptual Models	Research Agendas
Designers	[3, 105]	[61, 130]	[105]	[126, 1, 134]
Analysts	[105]	[61, 24, 63]		
Stakeholders	[3]	[24]	[35, 75]	[41]
Researchers			[51, 130]	[126, 14]

Table 3.1: Survey grid of papers according to (rows) visualization practitioner addressed and (columns) theoretical contribution type. Each paper makes many kinds of contribution, but we have tried to highlight readings that are more salient to us.

Prior theory about design studies

The relationship between visualization practitioners and their intermediate artifacts is not fixed, because subsequent interactions with stakeholders can drastically change their reading frame. In this way, practitioners are in close contact with boundary objects nearly all of the time, and not only when they are collaborating with outside stakeholders. Therefore, the roles held by each stakeholder are fixed points that the boundary object serves to ‘refract’, and we can see this phenomenon occurring recently in design studies through the next two examples.

Vuillemot et al. proposed to use boundary objects as a core concept recently [130], describing their collaboration between cartographers, and urban planners, and academic visualization researchers in these terms. Their paper provides extensive and practical detail about the roles of each contributor, and strategies employed by subgroups of contributors to form a competent collective.

Rogers et al. contributed a set of situated, interpretivist criteria for rigor in design studies [105], based on their own collaboration with evolutionary biologists. In addition to introducing new chart types to the literature, tailored to the specific analysis needs of their domain expert collaborators, they have shared detailed accounts of a visualization designer reflecting upon and revising design sketches that appeared in their earlier notes.

Disentangling visualization theory

In this chapter, we seek to describe visualization practices as *meaning making across multiple roles*. Although visualization theory has been been largely successful in defining these roles, and producing valuable guidelines in each discipline, that is not enough to weave them together. The nested model of visualization practice lets us pull the camera back and observe all the roles working together, just as Vuillemot et al. and Rogers et al. have done.

Each group chose to combine a theory contribution with a design study contribution; as practitioners, they actively seek to advance the theory. Meyer et al. [83]

sought in 2012 to put the nested model on firmer theoretical ground, building on the visualization literature at that time. This chapter extends the nested model with insights from design theory, which adapt and are in conversation with science and technology studies.

We read existing visualization practices through this lens, understanding that visualization systems research meaningfully carries on the legacy of cybernetics as a 'universal science'. In their recent survey "Are we making progress in visualization?," Correll highlights obstacles to this ideal, and thoughtfully questions whether it is a worthwhile framing [30]. Visualization isn't another psychology, or another sociology. To some practitioners, it's not even a science. What is it?

3.2.2 Visualization Practitioners Work Across Domains

The intervention of visualization practitioners must serve to meaningfully support stakeholders, even when the practitioner does not share the stakeholder's specific domain knowledge. Artifacts or ontologies which coordinate actions without requiring consensus are *boundary objects* [126, 124]. To the extent that experts from different domains speak different languages, by this definition, they have not reached consensus.

We identify the visualization system as a *coincident* boundary object: a product which has the same form across multiple teams, yet is used in multiple ways. Being a product, the visualization system is more portable across stakeholders than any intermediate artifact. The utility and meaning of a visualization is not preserved exactly across different groups of stakeholders, except in special cases like scientific plotting. Hinrichs et al. defend this property of 'sandcastle'-like visualization artifacts as a great strength [51].

For example, Akbaba et al. point out that stakeholders don't want to be understood only as data sources; they collaborate with visualization practitioners in order to produce practical tools for their own processes of analysis [3]. Visualization practitioners must create correspondences between their stakeholders' domain(s) of expertise and their own; if the visualization artifact does not 'translate' in both directions, its value as a boundary object is lost.

Ribes proposes that data science (which we interpret as a guise of visualization practice) inherits the remit of cybernetics, complex systems, and computer science: a 'universal language' through which any discipline may be addressed [101]. This language operates effectively across various disciplinary languages, bridging gaps and facilitating communication and understanding between experts from different domains.

3.2.3 Gaps Between Domains

In the context of interdisciplinary problem-solving, a visualization is a boundary object. Visualization has the unique ability to translate complex jargon into shared understanding, making it a crucial tool in bridging gaps between different domains of knowledge.

Caccamo et al. [20] identify three senses in which the term 'boundary object' functions in theory of management. It can describe (1) the visualization prototype itself, or (2) the visualization as a site of knowledge sharing between stakeholders, or (3) the data representation underlying the visualization as an infrastructure for knowledge sharing in its own right. These are all appropriate descriptions of different levels of the

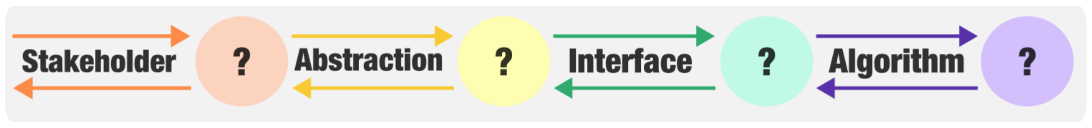


Figure 3.1: Representation of the nested model [89] as a graph. Edges correspond to layers, while nodes correspond to intermediate visualization artifacts. Layers are 'comprised of' design decisions [83], which become encoded into the intermediate artifacts, but may not be visible as such to more distant practitioners. In this way, every visualization artifact serves as a boundary object.

same stack, including the site of knowledge sharing: an ideal boundary object, whose existence is made possible by the visualization.

Data visualization practices turn data into stable forms of documentation - like screenshots of visualization systems, meetings to discuss visualization systems, and design specifications for visualization systems. To be precise, visualization practitioners translate knowledge from domain-specific terms (i.e. piles of numbers, jargon, and other opaque representations) into deep visual representations (which are hopefully not too dense). It is these representations which serve as common grounds for coordination in the hands of decision-makers and funding sources [35].

Mark et al. describe NASA's project selection process as operating on boundary objects [75], some of which are visualization systems. Project proposals are ideal-type boundary objects, containing documents and visualizations used inform decision-making, coordinate large teams across multiple disciplines, and reveal critical threats (say, technical capabilities that aren't adequately resourced within the proposed timeline). Mark et al. argue that these proposals ultimately promote shared representation, transform design knowledge, mobilize for design action, and legitimize design knowledge.

3.3 Visualization Produces Boundary Objects

In this section, we use the popular *nested model of visualization design studies*[89] to identify intermediate visualization artifacts, and explain their functioning with the language of boundary objects. The nested model (see Figure 3.1) describes process that visualization practitioners, deeply engaged with domain expert stakeholders, can effectively use to mediate their abstraction by providing appropriate visual interfaces. Each layer in the nested model is characterized as a process which faces certain threats to validity. In this section, we argue that these processes are linked by specific boundary objects, such as interviews with stakeholders and prototypes shared across teams, which mitigate the corresponding threats.

The nested model of visualization design studies is a structured and multicursal approach. Initially, the visualization team engages a stakeholder in an **interview** to identify their problem, collect their dataset, and identify the domain-specific language used to interpret the dataset. The visualization team creates **prototypes** of visual representations and interactions that they believe will support the stakeholder's problem-solving work. Simultaneously, the team will clean the dataset and implement appropriate visualization transforms to produce a working **visualization system**. In a typical iterative process, stakeholders will use this early system and be interviewed about their experience, allowing the domain problem to be understood more deeply, and the cycle to restart.

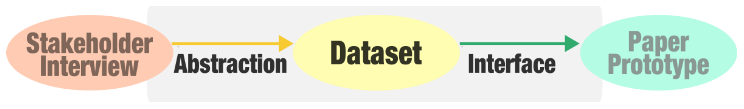


Figure 3.2: Cartoon of a conversation between a domain expert stakeholder (who owns the data and answers questions about it) and a visualization designer (who asks questions about the data so that they can make sense of it).

3.3.1 The Dataset is a Boundary Object

Every stakeholder who produces a dataset has in some way transcribed real-world objects or practices into *computational objects* that can be represented as rows, tables, and relations (suitable for transmission through computers) [14]. Fundamentally, these perspectives are related in different ways by different experts: Perhaps these are recordings produced by scientific or analytic instruments pointed at a certain physical or social phenomenon. Or they may be simulated traces of possible scenarios produced from a model of large-scale or slowly-progressing phenomena.

Because of this 'translation from world', each dataset is more than just a set of records; it is an expert's choice of representation, encoding some part of their perspective, and as such there is no universal mapping from the world to any given dataset. This is a design process as described by Tharchen et al. [126]. The dataset has been 'cooled down' to a non-interactive state, as both a repository-type and ideal-type boundary object.

In design studies, typically a visualization designer solicits a dataset from a stakeholder group represented by a domain expert (Fig. 3.2). Somehow, the designer must acquire 'the right' data - through understanding both what has been articulated in conversation, and which schema (types, tables, and relations) are present in the

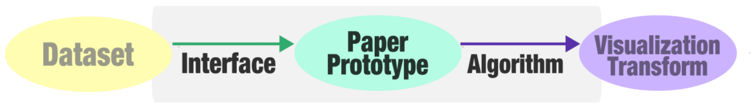


Figure 3.3: Cartoon of a conversation between a visualization designer (who wants to communicate a specification) and a data analyst (who wants to explain whether the specification is easy to implement, impossible, or something in between).

dataset at hand. In practice, the dataset often falls short in unpredictable ways, and the designer must collaborate with data analysts to perform a complex process of data wrangling [61].

Stakeholders who are not domain experts are likely to interact with datasets through visualization artifacts [41], including both paper and software prototypes. These visualizations serve an epistemic purpose, demonstrating the structure of the dataset to the broader group of stakeholders. This empowers the domain expert to 'speak' with other stakeholders in a language that makes concise more complex insights.

3.3.2 The Prototype is a Boundary Object

In the next step of their design study, visualization designers deliver paper prototypes, wireframes, or data sketches to the stakeholder (Fig. 3.2 again). These prototypes are used to identify whether the proposed interface elements make sense with the data types at hand, in the perspective of the domain expert. This collaborative move helps ensure that the prototype fits the domain's unique requirements.

Simultaneously, the visualization team's data analysts should deliver technical prototypes that explore these emerging requirements. Elements of the interface must be implemented, populated by visualization transforms (which may be novel and complex), then bound together with linked brushing, progressive disclosure, and other

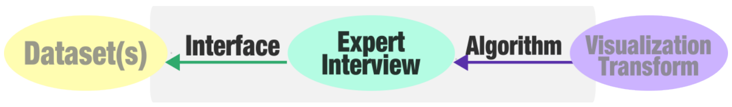


Figure 3.4: Cartoon of a conversation between a domain expert stakeholder (who wants to see their data) and a data analyst (who wants to know if the representation succeeded).

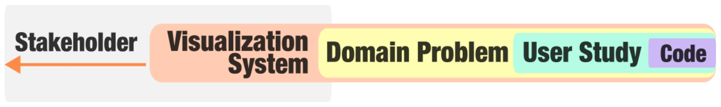


Figure 3.5: Cartoon of a share-out between the visualization team (who want to demonstrate some new insight), and a project lead to whom the domain expert stakeholder reports (who wants to know if their investment paid off).

navigational strategies. The data analyst strives to accurately represent each datum and its relationships within the dataset. An ongoing conversation between the analysts and designers (Fig. 3.3) is necessary to resolve messy real-world data and complex domain practices.

We argue that prototypes are boundary objects that facilitate communication between visualization practitioners and domain experts. Hopefully, the domain expert gains some insight by seeing their data represented in a new way. Simultaneously, the visualization practitioner learns more about the domain, enabling them to identify potential stakeholders and refine the visualization system itself.

The prototype itself acts as a *coincident boundary object*, ensuring that the final visualization system is both technically sound and tailored to the unique needs of the domain.

3.3.3 User Studies are Boundary Objects

User studies consist of watching stakeholders perform tasks with the visualization system or a prototype, and interviewing them about the process. During this phase, domain expert stakeholders have the opportunity to discuss their dataset in depth with the designers and analysts (3.4). This conversation helps bridge the gap between domain experts and visualization practitioners. It allows the practitioners to iterate on their grounded theory of the visualization tasks, and gain a deeper understanding of how their visualization system supports stakeholder needs.

Visualization systems effectively function as boundary objects *for their stakeholders, with respect to third-party stakeholders*. Ordinarily they are understood as *standardized boundary objects*, portable products expressing domain knowledge. In practice, they are also adopted into pre-existing workflows as tools for generating figures, or as sites of discussion.

Visualization exposes an object of study to multiple forms of domain knowledge, and allows discussion to proceed 'without consensus'. That is, these systems provide a common visual representation of data that allows experts from differing domains to engage in fruitful discussions and knowledge-sharing without full agreement on all details or terminology.

Diehl et al. position grounded theory as a tool shared with digital humanities by visualization practitioners [34]. (Where grounded theory refers to 'data', this means interview transcripts.) The process of coding an interview in the (constructive) grounded theory method is what produces transferable knowledge, in the form of models and shared ontologies.

3.3.4 Boundary Object-ness Validates Vis Systems

How should visualization research make claims to rigor? The design studies methodology argues that replicability is not an appropriate bar for visualization systems, since they are constructed through a contingent process. Instead, transferability must be assessed. This can occur in three kinds of ways:

Experimental validation occurs at both the algorithmic and interface levels. It assesses whether the tool is technically ready for use, ensuring it meets the needs of its stakeholders. It also examines the tool's readability, determining if it effectively communicates its intended information to users.

Grounded theory validation, following the approach of Rogers et al. [105] and as characterized by Diehl et al. [34], assesses the system's effectiveness. It focuses on the interface level, determining if the tool assists users in solving their specific problems and whether it enhances the legibility of the underlying issues.

Ecological validation refers to visualization systems that find their userbase. Who can effectively use the system and benefit from its insights? And who continues to work with the system, demonstrating its enduring utility and relevance? Visualization systems that function as boundary objects will foster collaboration and understanding among stakeholders with diverse expertise, which is ultimately our goal.

3.4 Discussion

In this section, we discuss various design methods and technical methods which are interlinked with visualization design studies. In particular, we identify the collection of datasets with the effort (and accompanying challenges) of knowledge production, especially in problems involving multiple stakeholders and contested ontologies.

3.4.1 Exploratory Visualization Methods

Visualization transforms are the computational pipeline that transforms a **dataset** and some visualization parameters into **interface elements** [99]. They are the infrastructure of a visualization interface, and often have unpredictable effects. Take bindings from data points to spatial coordinates as an example: visual clustering, gestalt effects, and navigable structure are all desirable and exploitable. Conversely, objects that stubbornly drift off the edge of the viewport, or hide behind each other, or evade selection otherwise are problems. All of these are hidden until the transform is combined with its dataset.

In response, one approach to *exploratory visualization* is to work in a reactive interactive development environment (IDE) with a notebook structure. These IDEs represent systems of multiple linked interactive visualizations (*widgets*) in one long scrolling page, divided into cells of code, which perform a mixture of data loading, visualization binding, and intermediate transformations.

Notebook IDEs in general contain both persistent data and a read-evaluate-print-loop (REPL); examples include the Jupyter notebook and the ObservableHQ notebook, as discussed by Vuillemot et al. [130]. Reactivity (such as in the ObservableHQ notebook) refers to the ability of data objects to automatically update themselves when objects upstream of them are changed. Capabilities like reactivity are especially powerful in exploratory visualization and interaction design, where precise design specifications are not knowable ahead of time [29].

3.4.2 Paper Prototyping Methods

As visualization practitioners in the field, we are reliant on stakeholders to describe their analyses to us, which is an intensive process. We are aided, however, by the pre-existing domain-specific practices of visualization, such as sketching on whiteboards or in Excel sheets.

For instance, Feynman diagrams encode both syntax and operations in quantum field theory calculations, as Chen et al. portray [25]. As Crease recounts [31], these diagrams are more than a record of the computation: in fact, an apparatus by which it is carried out. Similarly, computer scientists draw many graphs; these may describe anything from a parse tree (i.e. a paper tool for linguistics) to a database architecture (i.e. a paper tool for systems engineering).

Crease identifies Ursula Klein's notion of *paper tools* (originally described in her ethnographic study of organic chemists) as a broad class of domain-specific notations. Data visualization experts may already be familiar with the relevant paper tool if they have domain knowledge; but all visualization practitioners should be prepared to acquire paper tools from expert stakeholders, and to identify their relationship with whatever dataset is actually available.

Despite sharing some benefits (i.e. engaging active, intermodal perception), paper tools are not data physicalizations [56], which presume the existence of a particular dataset. Both describe a data schema equipped with certain operations; and like data physicalizations, paper tools can be used to ideate new operations more rapidly than implementation work would allow.

3.4.3 Knowledge Production is a Wicked Problem

Visualization practitioners are uniquely positioned to identify and address *wicked problems*, a term popularized in 1973 by Rittel and Weber [104]. For example, increasing the quality of visualization research is a wicked problem: the criteria for allocating publication slots (and other scarce resources) shapes the direction of the research in unpredictable ways, because these criteria are necessarily interpreted by different researchers engaged in different projects in different ways.

Zimmerman et al. [134] use wicked problems to highlight prior attempts to produce a 'science of design', which subsequently met "a problem that because of the conflicting perspectives of the stakeholders cannot be accurately modeled," leading Rittel and others to conclude that reductionism is not a theory of design. (Rittel's post at UC Berkley was title Professor of the Science of Design.) Correll's survey of visualization researchers about the problem of improving visualization research [30] shows us that practitioners may read visualization as a technology, or a design practice, and not as a science.

We are inspired by Rittel's argument that in planning, regret cannot be minimized [118], to offer the following provocation about interdisciplinary work: *communication cannot be maximized.*

We situate this claim in an observation about evaluation, paraphrased from Sedlmair et al. [113]: Designers, when confronted with wicked problems, cannot be expected to produce identical or even similar solutions. The goal of design studies is not reproducibility, but transferability - that the domain problem is well-articulated, and its solution is relevant to other domains.

3.4.4 The Contours of Datasets Trace Wicked Problems

Boundary objects become indispensable in confronting wicked problems because they emerge precisely where one domain cannot be easily translated into another. Star's earliest work on boundary objects [123] is set in a distributed planning context, where **objects** serve as shared representations of state between loosely coupled intelligent systems. In general, boundary objects enable different groups of stakeholders to recognize where one terminology ends and the other one begins, usually with some degree of overlap.

Ribes emphasizes that experts traversing different domains do not necessarily create shared ontologies; instead, they establish concordances that facilitate translation across domains [101] at multiple levels of abstraction. Project stakeholders who engage in conversation with visualization practitioners and data scientists are, through that process, bridging the gaps between their own disparate domains.

Given a situation where all stakeholders operate in the same domain, we would not expect to see outside visualization experts involved. Rather, someone will have already created a visualization system (such as packages for producing violin plots, or Gantt charts) that meet typical stakeholder needs in this domain. Design studies act to identify sites of visualization intervention into user workflows that cross domains, and aren't already worked out this way. Where multiple stakeholders are involved (e.g. domain scientists, engineers, and project leads), the intervention of visualization practitioners can facilitate communication across groups.

While datasets themselves can be considered boundary objects, their interaction with stakeholders is predominantly mediated through visualization artifacts. These artifacts transform domain-specific records into communicative visualizations and interfaces. Prior to visualization, datasets have limited utility as boundary objects because they are not well-suited for human perception. Their structural understanding remains confined to a select group of experts. Datasets may originate from scientific instrument recordings targeting specific phenomena or be crafted by experts as pragmatic representations of complex, challenging-to-instrument phenomena.

Visualization practitioners do not evaluate whether a dataset addresses a stakeholder's 'real problem', but they perform a more valuable role: stakeholders can gauge this better once they have a means of visual analysis. Domain expert stakeholders often benefit from the addition of visual modalities which enhance their existing processes of analysis [29, 3, 130, 105, 8, 51].

3.5 Conclusion

This chapter is intended to support ongoing conversations about boundary objects in data visualization, and the deep connections between datasets, interfaces, and abstractions. There are emerging methodological considerations at each level of visualization work: the interview as a site of grounded theory, the paper tool as an interface to domain expertise, and the visualization transform itself as a site of exploratory data analysis. The visualization practitioner must draw on all of these methods in order to understand stakeholders' diverse needs, and ultimately contribute a visualization artifact that functions as a boundary object.

While software instruments themselves are visualization artifacts, all computational media practitioners can use boundary object theory to characterize properties of their artifacts which support rich communication between stakeholders with different perspectives.

Part II

Practice

Chapter 4

MarsIPAN: A Schedule for Mars Sample Return Operations Planning

This chapter is based on the paper "MarsIPAN: A Schedule for Mars Sample Return Operations Planning" coauthored by myself, Malika Khurana, Noah Deutsch, Benjamin Donitz, Oskar Elek, and Scott Davidoff (citation forthcoming). In this chapter, "we" refers to the team of authors.

The upcoming NASA/ESA Mars Sample Return (MSR) mission under development will situate multiple spacecraft on the surface of Mars, each with their own operations team, who must collectively allocate a finite, scarce communication bandwidth between Earth and Mars. We conducted a data-first design study to develop the MarsIPAN schedule, which visualizes the relay passes allocated to each operations team.

Through this process, we engaged diverse MSR stakeholders to understand how they read pass allocations. In particular, we worked closely with the creators of the MARMLADE pass allocation design support tools, which empower stakeholders to explore a tradespace of allocations based on different possible assumptions. In this paper, we produce a taxonomy of tasks performed by MSR stakeholders, and an ecology of data artifacts that the stakeholders use to coordinate their analyses. We identify shifts over time in their critical visualization needs as these experts seek to develop an ideal mission plan together. Ultimately, we characterize the schedule as a visual representation of pass allocations that stakeholders with diverse expertise all play a role in reasoning over.

4.1 Introduction

Because communications bandwidth to Mars (and anywhere in deep space) is such a scarce resource [69], space agencies plan bandwidth allocation out for years into the future to ensure successful mission operations [110]. The under-development Mars Sample Return (MSR) mission [88] will put extreme stress on that infrastructure by situating multiple spacecraft, each with its own independent and varying uplink and downlink bandwidth requirements, into a single region on the Mars surface.

Space agencies often approach this bandwidth problem as a scheduling problem, optimizing the efficiency of the deep space antennas resource allocation in terms of fixed and flexible constraints. Previous mission analysts worked with x, y and z to model and understand communication bandwidth for single spacecraft. But MSR introduces the challenge to support balancing of priorities of multiple spacecraft, each with different operations teams and goals, and potentially different operations cadences and distinct uplink and downlink priorities.

To support the complex trade space of reasoning about the economy of communication bandwidth and its relationship to mission success and risks across multiple missions, we introduced MarsIPAN (the Mars Interactive Pass Allocation Navigator), a visual analytics tool to support coordinated planning of operations on Mars across multiple robotic surface assets and their respective teams. MarsIPAN serves as a visual and collaborative analysis layer for the MARMLADE multi-spacecraft scheduling simulation [36], developed previously to support MSR campaign goals by maximising operational efficiency under various possible scheduling constraints. The goal of the MarsIPAN schedule is to transform the MARMLADE allocations into shareable visualizations.

In this chapter, we report on a Design Study [113] of the MarsIPAN system, and report on lessons learned through the creation of the application, and their broader implications for the design of complex scheduling visual analytics. MarsIPAN has been designed in collaboration with domain experts in surface operations and relay operations at a large research laboratory participating in the mission (anonymized for review as "The Lab"). We characterize how domain experts participating in MSR from multiple disciplinary perspectives collaboratively analyze candidate schedule together using MarsIPAN application. These experts each incorporate their own knowledge, such as specific mission goals on a given Mars sol, to produce a more complete picture of the schedule.

We describe scheduling analysis data, analysis tasks and their design drivers, and how each manifests are particular features in the visual analytics systems developed. We describe a user study that observes the impact of how MarsIPAN users create a schedule supporting event coordination over 300 days across both Earthtime and Marstime.

This chapter provides the following visualization contributions:

1. A characterization of the analysis process and design considerations the for analysis of constraint optimization scheduling algorithms
2. The MarsIPAN application, a visual analytics system to address scheduling optimization analysis data and task abstractions
3. Themes, insights, and excerpts from a system evaluation that describe the fit of the application to the data and tasks. In particular we find that even expert users from different backgrounds coordinated their work across four levels of abstraction

In this chapter, we review prior work in schedule visualization. We then go on to describe the user and task abstractions for telecommunications analysis, and connect those to the design of the MarsIPAN schedule visualization tool. We then describe the findings from an evaluation of the tool, and connect those more broadly to the design of collaborative scheduling optimization analysis tools. We start by situating this chapter within the landscape of related research.

4.2 Background

Timelines are a ubiquitious class of event sequence visualizations which, in the context of scheduling, represent events that many agents need to coordinate over. In particular, events may not be fixed and past: rather, they are planned in the future, and the schedule is used to inform all agents so that they can participate in those events. In this section, we describe prior work in tools for the visual representation of schedules produced under certain constraints.

4.2.1 Scheduling Optimization Algorithms

Time series data form a useful part of computer science, appearing across a diversity of domains from knowledge retrieval [6] to biochemistry [68] and medicine [18]. Our interests lie in a subclass of time series data that include schedules, which are time series that define relationships amongst temporal events, often ordered around a constrained resource, whose use needs to be optimized [26, 57]. Schedule optimization problems are also found across a huge diversity of domains from robotics and manufacturing [58, 22] to logistics and operations research [71, 131].

Schedule optimization aims in general to optimize an efficiency metric within some domain-specific set of constraints [40]. This particular design study looks to operationalize schedule analysis of the MARMALADE scheduling algorithm [36], developed to support analysis of the Mars Sample Return mission [88], currently under development. Mars Sample Return looks to return to Earth the astrobiological samples currently being collected by the Mars Perseverance Rover [43]. The current MSR mission architecture includes a Sample Retrieval Lander (SRL), multiple Sample Retrieval Helicopters (SRH's), a Mars Ascent Vehicle (MAV), and an Earth Return Orbiter (ERO), radically increasing the local population of surface robots. The radical constraint that the mission needs to consider is that the data sent to command those spacecraft will need to fit within the current bandwidth constraints of Mars telecommunications. And because the spacecraft will rely on solar power, each spacecraft needs to operate every day, so that accumulated dust doesn't kill the spacecraft.

The MARMALADE algorithm [36] was developed to assess if the MSR mission was possible at all, given the tight communication constraints. On each Mars day (sol), a closed decisional cycle consists of an uplink pass (where instructions go to the surface asset on Mars), then an execution window, followed by a downlink pass (where the asset returns its data for that sol's operations). In each MARMLADE output allocation, *operational efficiency* is maximised by closing as many decisional cycles as possible for each surface asset.

Previously, communication scheduling simulation data was analyzed by each team independently using their own bespoke software, and brought into the live operations context by printing these schedules out on paper. But these existing tools and processes do not separate the concerns of individual teams from the business of coordinating operations using those scarce resources like communications bandwidth that are shared between teams. This unmet need led us to create the MarsIPAN visual analytics application.

A number of research projects have investigated ways to configure scheduling optimization algorithms, and connect their output to visual analytics systems. In this next section, we describe how MarsIPAN sits within the landscape of other scheduling optimization visual analytics tools.

4.2.2 Visual Analytics and Optimal Schedules

One key axis around which schedule visualization systems can be organized is how long a typical ‘event’ is – that is, the level(s) of **granularity** which they attempt to optimize over [103]. In practice, events are very heterogeneous between domains; within computer science, they might include computing processor function calls measured in milliseconds (with schedules covering ten-second blocks), or individual minute-long robotic task events and schedules covering hours long exploration plans [7].

In the project management literature, on the other hand, events may constitute an arrangement of shifts during one work-week, or processes taking multiple days across a months-long construction schedule [127]. MarsIPAN supports analysis of relay pass events of around an hour, and outputs an optimized schedule covering the multi-year campaign from launch to safe earth return of the samples.

Another organizing axis is that of **agential autonomy**: how ultimate schedule autonomy is apportioned between the scheduling algorithm and its users, and how that tradeoff is negotiated using the application. Outside of computer science and robotics, where systems are expected to operate unsupervised, most systems produce candidate solutions which analysts may then explore.

Air traffic controllers work, for example, with ‘suggested trajectories’ that allow controllers themselves to negotiate the eventual outcomes [45]. An academic conference scheduling system might provide a rough outline of a possible schedule, leaving conference organizers to reorder proposed solutions that depend on their intimate subject matter expertise, which is not encoded by the scheduling algorithm [64]. And in aircraft line maintenance staff scheduling, the system outputs options that managers could then edit, allowing them to apply their domain expertise around existing best practices and social mores[9]. MarsIPAN output serves more closely as suggestion, depending on the deep domain expertise of spacecraft operations teams to know when to make claims counter to the ouput of the scheduler's oversimplifications.

Finally, schedule optimization systems vary in their **explainability**, which reflects the ability of their accompanying visualization system to represent the scheduling algorithm's underlying abstractions and decision process. Schedules with higher explainability are those which engage multiple groups of stakeholders in due process regarding scheduling outcomes. This largely political definition reflects that schedules are not only a tool for performing analysis, but also for communicating those analyses to a broader team.

Experts typically adapt visual analyses into new explanations through the medium of written reports and presentations (white boxes in Fig.4.1), taking screenshots which represent curated subsets of data. Scheduling algorithm decisions might be presented as outcomes without context [109], which can help when dealing with overwhelming amounts of data, but can make their decisions somewhat opaque.

MarsIPAN is designed to support visual analysis of periodicity and the temporal relationships between events, including both the relay passes themselves, and the tactical execution window which yields the decisional data used to plan the next sol of execution. In this role, it facilitates existing processes of negotiation between teams (Section 6.2), used to reach consensus on a schedule, by helping teams to jointly understand when operations are not possible.

4.3 Methodology

The MarsIPAN schedule visualization system was developed according to a data-first design study methodology [113, 93], in which our visualization team was embedded with the Lab collaborators. Thanks to their availability and assistance scheduling interviews with other on-Lab domain experts, we were able to iterate on MarsIPAN prototypes while remaining sensitive and responsive to changing stakeholder needs. We developed our characterization of the experts' visualization tasks through iterative rounds of this process.

Our team bootstrapped the requirements for MarsIPAN by mocking up and building out different components of MarsIPAN, and sharing these with the algorithm team. Based on our initial interview of the allocation algorithm team, we began an iterative prototyping process [53], which allowed us to identify the following tasks.

We then used prototypes to visualize relay pass allocations (e.g. Table 4.1), which depend on both stakeholders' scheduling assumptions and on hard geometric constraints (i.e. which relay passes will be provided by Mars orbiters). These prototypes grounded our conversations with stakeholders (Fig. 4.1) in the real data they are familiar with (Fig. 4.2), and allowed us us to elicit tasks representing multiple levels of abstraction (Fig. 4.8).

We conducted formative user fieldwork with two allocation algorithm designers over ten weeks, and used those discussions to map the problem domain for subsequent iterations of design. We met with the allocation team 60 times (who used the tool for 10 min each time, on average), and with external users 10 times (who used the tool for 1 hour each time) across 18 months of the design study. In total, the MarsIPAN schedule received 20 hours of evaluation from domain experts throughout its development.

We conducted end-user evaluations of the software prototype every 6 months (twice during the study period), giving users tasks, asking them to think aloud, and capturing attitudinal self-reports. For the interviews conducted after 12 months of development, we transcribed these and extracted themes, which are described in Section 4.8.

The data ecology at JPL has been ideal for conducting a data-first design study [93] on pass allocations with their many expert users. The pass allocation algorithm is downstream of many contingent assumptions, which are captured in data stores and through interviews with other stakeholders. When our team approached the allocation data for the first time, the algorithm team initially talked us through their 'day in the life' schedule visualization (consisting of a manually created heatmap in Excel). As visualization developers, we learned to read this figure by algorithmically implementing it for the first time, creating MarsIPAN.

4.4 Design Study

This section describes the ontology which distinguishes MarsIPAN from other scheduling systems, by tailoring its use to investigating operational cycles in Mars-Earth communications. At this stage, we have conducted numerous informal interviews with the team responsible for producing optimal pass allocations for each scenario of MSR operations under consideration. Each interview was supported by iteration on a prototype visualization of allocation data, exploring different representation strategies, which we will elaborate upon in the following section.

4.4.1 Pass Allocation Dataset

MarsIPAN allows users to collaboratively analyze, and interact with communications scheduling data output by the MARMALADE algorithm [36]. The main data object analyzed and optimized by the scheduler is a *relay pass allocation*. Each relay pass is a window of time (see Figure 4.1) in which an orbiter flying around Mars can send and receive data from spacecraft on the Mars surface. The algorithm parameterizes Mars surface spacecraft operational window (because of energy and temperature considerations) and the Earth operational time window (because of human factors constraints), and computes potential communication windows. For each window, it estimates the bandwidth using the geometry of the Earth-Mars connection. Each allocation includes data from Earth, or *uplink data*, and data to Earth, or *downlink data*, and include a data volume and Earth-Mars transit time. Allocations are computed each operational day, or *sol*, on Mars, for each spacecraft. Mission planners vary their assumptions, and simulate an entire mission

Analysts then assess the efficacy of each pass allocation. On any given sol, according to the mission plan, a spacecraft has a number of activities to perform. The Bandwidth needs for each of these activities are modeled using historical data. For a sol to be considered active, each spacecraft must receive uplink data before the execution window on Mars (i.e. when the surface asset has enough sunlight to drive around and use its instruments), based on data returned to Earth after the previous sol's execution window. The first several hundred megabits of downlink after each execution window (always taking place in the sunny Mars afternoon) are called *decisional.* These downlinks include sensor data from the rover that must be interpreted by the operations team in order to plan the exact commands for next sol. As planning is itself time-consuming, it is not always completed in time for uplink on the next sol, in advance of the next execution window. Sols in which operations are not possible are referred to as *restricted sols*, or Mars days on which one or more surface assets cannot operate.

Across the entire optimized schedule, analysts are also looking to understand, for a particular configuration of the possible parameter space, what is its *operational efficiency*, which is calculated for each surface asset as the percentage of sols where that surface asset could operate based on data from the previous sol. And the principal task of the analyst, and the main motivator for the MarsIPAN application, is to provide traceability to the root causes of the operational efficiency of any particular pass allocation. We discuss how different users engage in that process in this next section.

4.4.2 Users and Task Abstractions

MarsIPAN presents allocations drawn from the tradespace of possible pass allocations, to a number of users, each of whom then conducts a series of analysis tasks using the data, both individually, and collaboratively, which range from the highly abstract to the more concrete. This section unpacks each of these user groups, and describes their individual and collective tasks.

Teams working on MSR surface operations work together to identify threats to the goals of the mission as a whole. A campaign lead coordinates the teams for all MSR surface assets, who individually develop a plan to use whatever data volume is

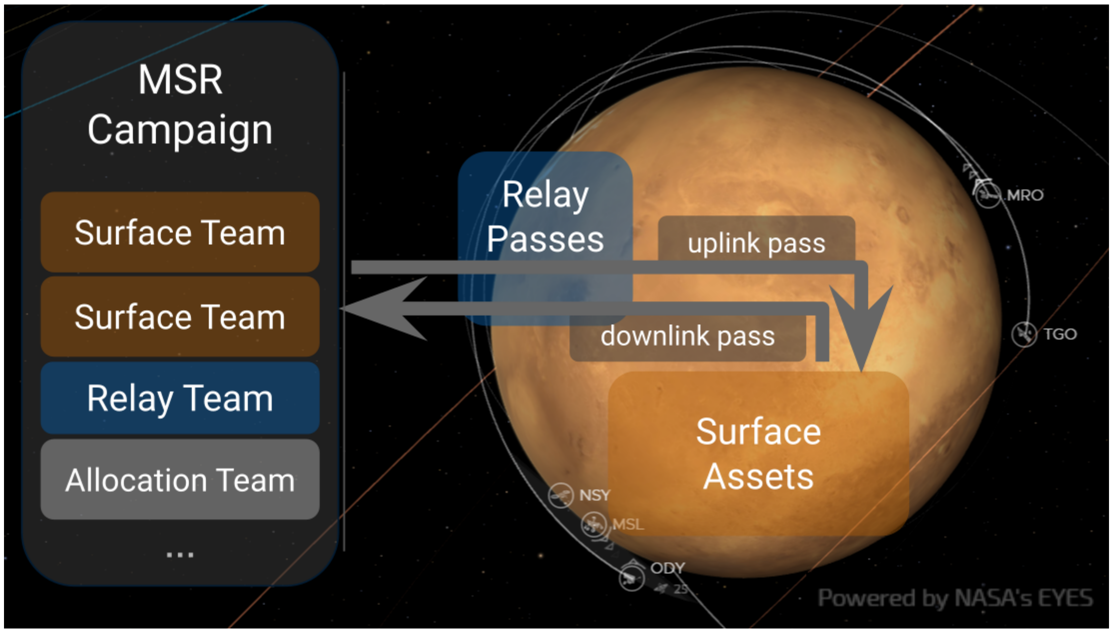


Figure 4.1: Conceptual map of Mars Sample Return Campaign stakeholders and their respective responsibilities. The relay team (blue) provides pass bandwidth to and from Mars, which the surface teams (orange) need to operate their respective assets. The allocation team (light gray) creates a recommendation for each relay pass, including whether it should be used for uplink or downlink, in order to meet all operators' planning needs

Orbital	**Surface**	**Sol**	**LMST (Hr)**	**Assignment**	**FWD (Mb)**	**RET (Mb)**
ERO	M20	125	8	RET	5	400
MVN	SFR	125	11	TOTAL	2	200
DFE	M20	125	22	FWD	1	0

Table 4.1: Simplified version of the data schema for each relay pass allocation. The Orbital Asset provides the communications pass; it is either one of four Mars orbiters, or direct-from-Earth (DFE) x-band communication. Each Surface Asset is a robot on Mars, commanded by its own operations team. Sol Number and LMST describe when the pass occurs in Marstime; in addition, the actual schema includes pass start/end times, and their UTC counterparts. Assignment is one of RET (decisional return), TOTAL (non-decisional return), or FWD (decisional forward): FWD and RET Volumes are the amount of data that could be, respectively, either uplinked or downlinked by this pass.

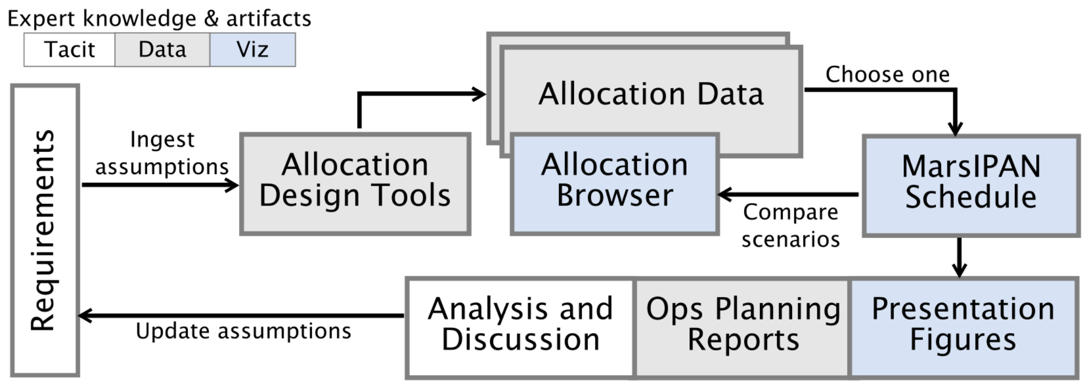


Figure 4.2: Map of the data ecology around the MarsIPAN schedule, which is used by experts involved in the MSR campaign to create figures, reports, and analysis from datasets based on contingent assumptions that are continuously updated by this activity. Tacit expert knowledge (white) is recorded in datasets (grey), which are communicated through visual representations (blue).

available to support their mission goals.

Relay Operations Team

The relay operations team wants to maintain the health of relay passes in adversarial scenarios. They ensure that the telecommunications requirements for the mission can be carried out on the available hardware. Their data products describe relay capabilities across the mission duration, accounting for orbital geometry (e.g. Earthrise time on Mars), and even factors like mountains blocking line-of-sight to rovers. These are ingested by the pass allocation algorithm. Like the algorithm team, they are concerned with orchestrating communication for all surface assets involved in the mission to maximize operational efficiency and prioritize critical data transfers.

The relay lead thinks ahead to tactical mission needs: because they expect the allocation does not reflect exactly how the mission will go, they use it as a starting point to anticipate actual behavior and to prepare alternate communication plans. Tasks for the Relay Operations Team include the following: (1) Prioritizing critical data transfers; (2) preparing alternate communication plans; (3) characterizing relay pass health and latency times in various possible scenarios, e.g. where legacy orbiters have survived until the mission timeline.

Surface Operations Team

On MSR, each surface asset has an operations lead. The surface operations team wants to know when their team will have data and be able to plan. They are concerned with coordinating each spacecraft's needs at different phases of the mission with planning team schedules, and identifying when the team on Earth must work odd hours. A high-quality staffing schedule will avoid Marstime staffing, in which the entire operations team adopts a 25-hour schedule on Earth for a strenuous period of time. The surface operations team also has a team lead, whose job is balance the at-times conflicting needs of each of the individual spacecraft, so that the overall mission can be successful.

Tasks for the Surface Operations Team include the following: (1) schedule their team to accomplish mission goals with the available bandwidth; (2) characterize the tradeoff to their team if a given assumption changes; and (3) identify threats to the mission goals for a particular span of time, given certain tasks (e.g. sample tube transfer) will be especially demanding of the operations team.

View	Model	Supported Task
Fig 4.3	Pass structure	See passes yesterday and tomorrow
Fig 4.3	Navigation track	Identify sols with missed operational cycless
Fig 4.4	Timeline	See if pass is in the morning
Table 4.1	Pass detail	See exact time + data volume of a pass
Fig 4.6	Selection quer	Highlight interesting passes

View	Component	Our Design
Fig 4.3	Faceted timeline	Decisional passes bracket operational cycle
Fig 4.3	Sequential timeline	Multiple fixed zoom levels
Fig 4.4	Temporal axis	Earthtime as secondary index
Table 4.1	Detail template	Linked selections on hover
Fig 4.6	Query builder	Incorporate Arquero queries

Table 4.2: Table of visual encodings used in MarsIPAN, as shown in wireframes and clickthroughs.

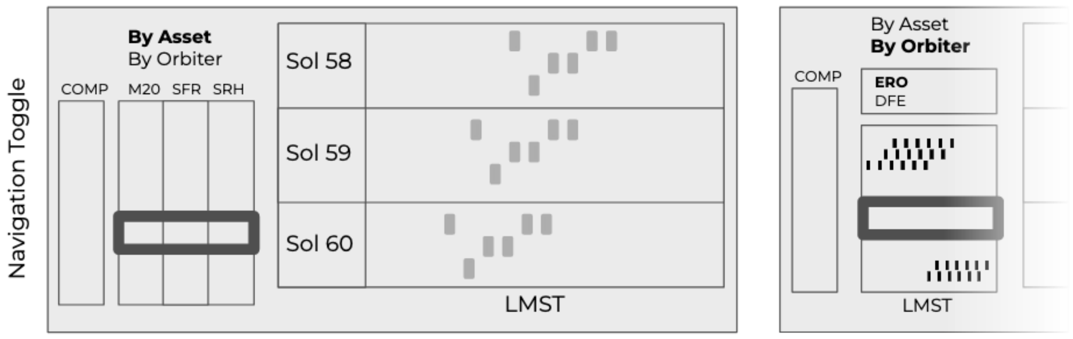


Figure 4.3: Wireframe of the two navigation views in MarsIPAN, by asset (left) and by orbiter (right). The highlighted rectangle is a 3-sol window that the user can drag along the 30-sol navigation track to scrub through the allocation. A second, 30-sol window selection on the 300-sol navigation track operates similarly.

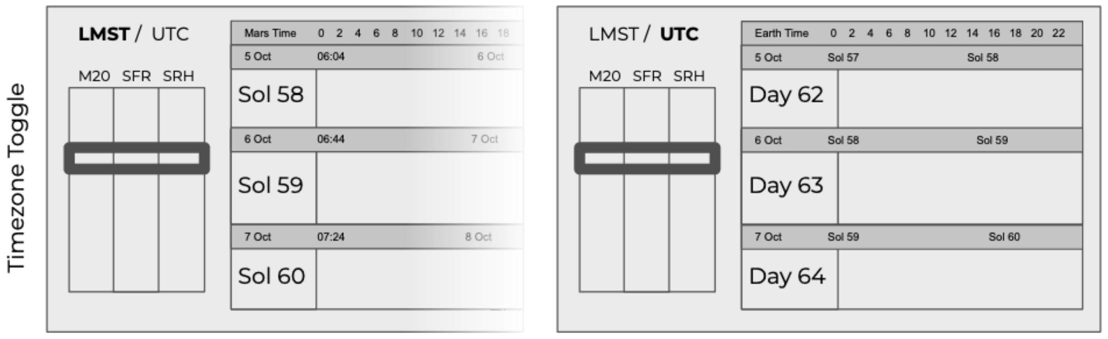


Figure 4.4: Wireframe of the two faceting modes in MarsIPAN, by Mars sol (left) and by Earth day (right).

4.5 Visualization System

Having established our experts' needs, we now describe the complete and interlinked set of interface widgets comprising the MarsIPAN schedule analysis application. Wherever the same 'event' abstraction - either a pass, an execution window, a decisional cycle, or a query - appears in multiple widgets, we link those selections across both widgets. Specifically, decisional cycles are represented both in the schedule itself and in the navigation sidebar, as are query selections (applied to the set of passes).

4.5.1 Navigation Track

The MarsIPAN sidebar contains side-by side views showing multiple zoom levels. At the leftmost side of the screen, a vertical 'navigation track' represents all 300 Mars sols in the schedule. Sols with larger numbers of missed decisional cycles are shaded more deeply to form a 'barcode'. Since there are three surface assets, this colormap has four stops, allowing subtle variations to be read in a tightly condensed format.

Between the 300-sol navigation track and the schedule, we have placed a detailed 30-day navigation track, which is horizontally faceted into the three surface assets. Redundantly, each asset is shaded with its own color. Sols with missed decisional cycles are 'struck' with a black cell and a white 'X', and any sol with no passes selected is faded out.

A sliding window on the 300-sol track controls the 30-sol track, and a sliding window on the 30-sol track controls the 3-sol schedule view. Because the 3-sol sliding window is constrained by the 30-sol sliding window, either one can be used to browse the schedule, depending on how quickly the expert wants to browse. This interaction pattern has been used previously in genome sequence browsers [108], but rarely in temporal schedules, which are not usually both long and detailed.

In addition to scrolling the schedule, the navigation tracks together act as a minimap, providing context to passes the expert is inspecting - i.e. whether the pass pattern is off-nominal, and if so, at what cadence. In the Earthtime view, both navigation tracks are re-grouped by Earth days instead of Mars sols, thus remaining 1-1 with the schedule.

4.5.2 Schedule Panel

The schedule panel is the most important view in MarsIPAN, containing all of the allocated and unallocated relay passes from a given scenario. Unlike in conventional schedules, Earthtime and Marstime (where each sol is about 25 Earth hours long) are on dual x-axes with each other. A toggle button allows users to switch back and forth, translating the schedule immediately between Earth hours and Mars hours (of which there are 24 each sol, making them slightly longer).

By default, passes are grouped into tracks and colored according to which surface asset they are allocated to: either the Mars 2020 rover (M20), the Sample Return Lander (SRL), or the Surface Fetch Rover (SFR). A second toggle allows passes the schedule to be colored by which orbiter provided the pass, rather than by which surface asset consumed it (Fig. 4.7). It also changes the navigation track into a 'pass pattern' overview, with each orbiter on a different layer.

Each pass is represented by a mark with three parts: the pass body, the brackets (to indicate 'opening' or 'closing' the period when data is 'on Mars'), and the tails (to indicate latency time as data travels through space and is relayed by the DSN). The pass body's width corresponds to the actual pass length.

4.5.3 Bookmarks and Query Builder

The selection query builder, pass info box, and bookmarked passes sit together on top of the rest of the schedule. When the expert hovers their mouse over any pass in the schedule, selected properties of that pass (which orbiter provides it, with how much bandwidth, compared to the surface asset's data volume requirements) are shown in the pass info box, including exact start times in UTC and LMST. Clicking on the pass bookmarks it to a table just underneath the pass info box, where the same information is represented in a vertically compressed format (and a 'remove bookmark' link is added). Clicking on a bookmarked pass re-centers the schedule view onto that pass.

The query builder itself is an Arquero query library demo[1] that we have integrated into MarsIPAN. In MarsIPAN, pass selections are subsets of passes suitable

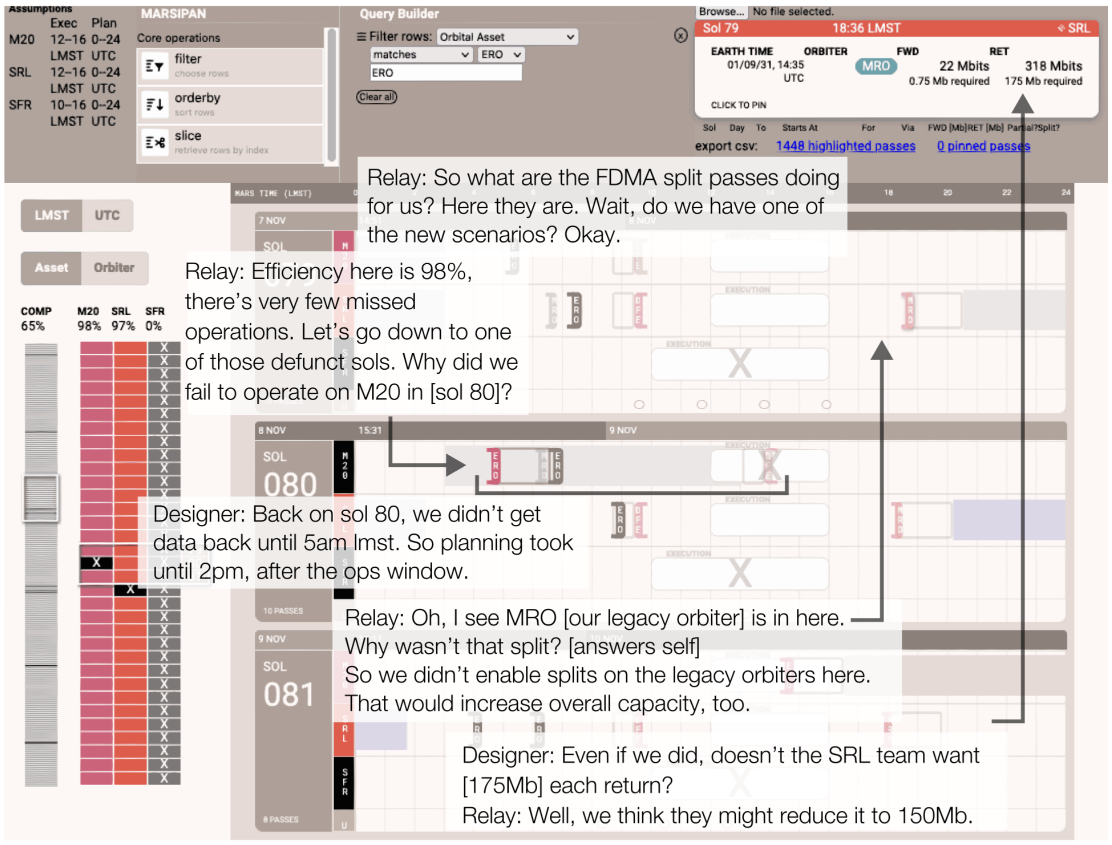
[1]Open source: observablehq.com/@observablehq/data-wrangler

for either direct exploration (only selected passes are shown in full opacity, on both the schedule and the navigation track), or export to a CSV file. All operations beyond filtering, sorting, and slicing are disabled, since we did not have scope to implement graphical feedback on more advanced operations, as these may produce relational tables which are not subsets of the table of passes.

We did extend the filter component to include (for each possible column in the table of passes) both a slider and a suitable textual input for multiple numeric and timestamp data types, or a dropdown selection for enumeration data types. By composing a sequence of filters (where order matters, because the extent of the selection is used by the filter component - rather than the extent of the dataset), experts were able to express a rich variety of Crossfilter-like queries without leaving the MarsIPAN schedule, and export the results of queries for further analysis.

4.6 Implementation

Our visualization stack is based on D3.js widgets [15], so that the SVG representations are created directly from data points, and expose them to user-generated pointer events. Due to unusual and heterogeneous representations - such as the y-facet's unallocated pass track (which is smaller than the other three schedule facets), the x-axis' dual Marstime/Earthtime representation, and the linked scrolling behavior across multiple navigation tracks - it was not convenient to implement our design with high-level plotting grammars.



Relay: So what are the FDMA split passes doing for us? Here they are. Wait, do we have one of the new scenarios? Okay.

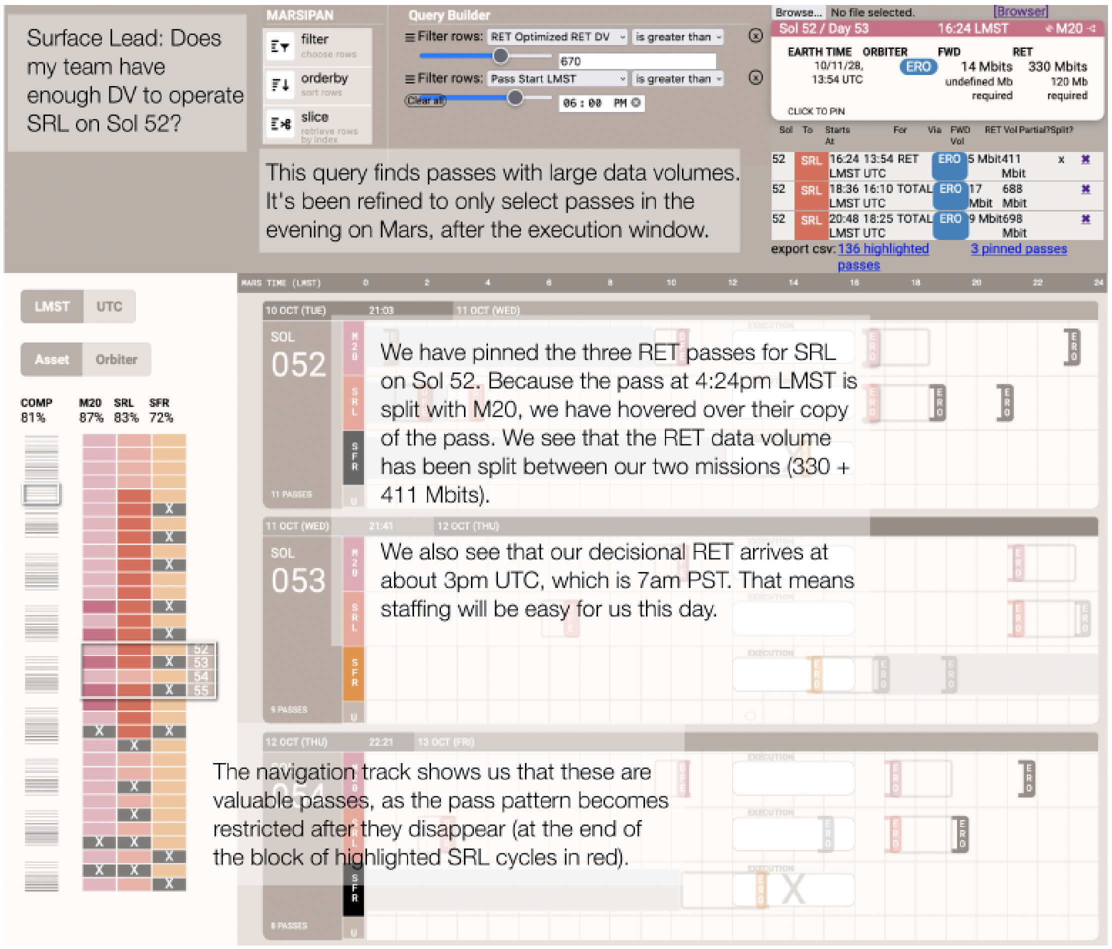
Relay: Efficiency here is 98%, there's very few missed operations. Let's go down to one of those defunct sols. Why did we fail to operate on M20 in [sol 80]?

Designer: Back on sol 80, we didn't get data back until 5am lmst. So planning took until 2pm, after the ops window.

Relay: Oh, I see MRO [our legacy orbiter] is in here. Why wasn't that split? [answers self]
So we didn't enable splits on the legacy orbiters here. That would increase overall capacity, too.

Designer: Even if we did, doesn't the SRL team want [175Mb] each return?
Relay: Well, we think they might reduce it to 150Mb.

Figure 4.5: Use case demonstrating how the relay lead reads a MarsIPAN schedule. They diagnose an operational efficiency loss that could be avoided if the MRO (Mars Reconnaissance Orbiter) pass were split. During simulation, the algorithm was configured to only split ERO (Earth Return Orbiter) passes, in a more conservative estimate of technical capability.


Surface Lead: Does my team have enough DV to operate SRL on Sol 52?

This query finds passes with large data volumes. It's been refined to only select passes in the evening on Mars, after the execution window.

We have pinned the three RET passes for SRL on Sol 52. Because the pass at 4:24pm LMST is split with M20, we have hovered over their copy of the pass. We see that the RET data volume has been split between our two missions (330 + 411 Mbits).

We also see that our decisional RET arrives at about 3pm UTC, which is 7am PST. That means staffing will be easy for us this day.

The navigation track shows us that these are valuable passes, as the pass pattern becomes restricted after they disappear (at the end of the block of highlighted SRL cycles in red).

Filter rows: RET Optimized RET DV · is greater than · 670

Filter rows: Pass Start LMST · is greater than · 06 : 00 PM

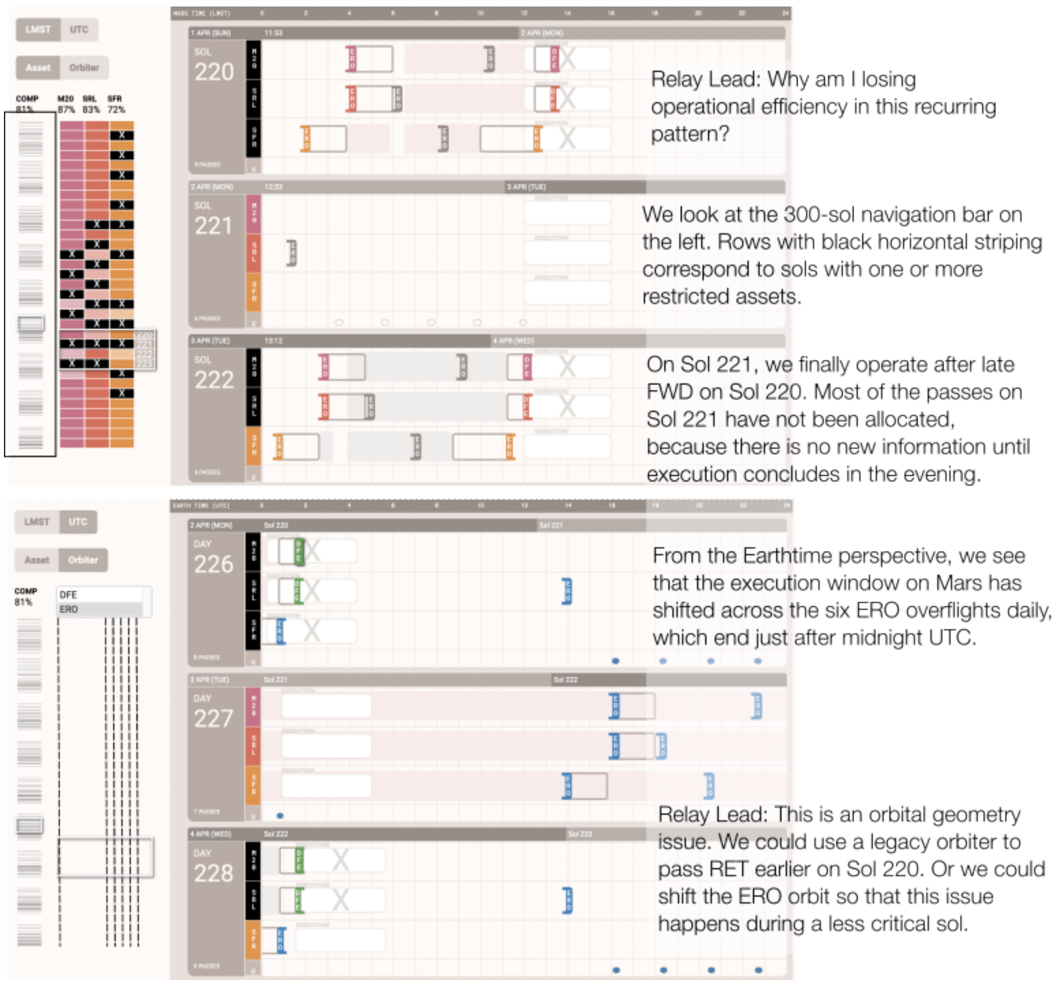
Sol 52 / Day 53 · 16:24 LMST · M20

EARTH TIME	ORBITER	FWD	RET
10/11/28, 13:54 UTC	ERO	14 Mbits undefined Mb required	330 Mbits 120 Mb required

Sol	To	Starts At	For	Via	FWD Vol	RET Vol	Partial?	Split?
52	SRL	16:24 LMST 13:54 UTC	RET	ERO	5 Mbit	411 Mbit		x
52	SRL	18:36 LMST 16:10 UTC	TOTAL	ERO	17 Mbit	688 Mbit		
52	SRL	20:48 LMST 18:25 UTC	TOTAL	ERO	9 Mbit	698 Mbit		

export csv: 136 highlighted passes · 3 pinned passes

Figure 4.6: Use case demonstrating the query builder. In this figure we consider a subset of passes in the evening on Mars, after the execution window.



Relay Lead: Why am I losing operational efficiency in this recurring pattern?

We look at the 300-sol navigation bar on the left. Rows with black horizontal striping correspond to sols with one or more restricted assets.

On Sol 221, we finally operate after late FWD on Sol 220. Most of the passes on Sol 221 have not been allocated, because there is no new information until execution concludes in the evening.

From the Earthtime perspective, we see that the execution window on Mars has shifted across the six ERO overflights daily, which end just after midnight UTC.

Relay Lead: This is an orbital geometry issue. We could use a legacy orbiter to pass RET earlier on Sol 220. Or we could shift the ERO orbit so that this issue happens during a less critical sol.

Figure 4.7: Use case demonstrating the ERO pass pattern. In the lower-left hand corner, the track demonstrates the alignment of ERO passes (in orbit around Mars) with Earth time.

4.7 Usage Scenarios

MarsIPAN will be used by MSR campaign stakeholders to perform the analysis tasks appropriate to their specific role. We provide three click-through scenarios in this section to illustrate how experts use the schedule interface to explain and understand the underlying scheduling algorithm.

4.7.1 Scenario: Relay lead diagnosing a split pass

The **relay lead** discusses the availability of split passes on legacy orbiters with a visualization designer on the **algorithm team** in order to *maintain relay pass health* (Figure 4.5). Upon loading the schedule, the relay lead immediately asks what split passes are doing for the mission health in general, and ask to see the latest scenarios (i.e. where SFR no longer receives pass allocations).

Looking at the operational efficiency figures, they see that M20's operational efficiency is very high at 98%, meaning only a few sols are defunct. They identify Sol 80 as one of these defunct sols by observing the strike mark in the bar code, and scroll to that point in the schedule. The designer explains the algorithm's decision by saying that decisional data did not arrive until 5am LMST, which is too close to the execution window for planning to be completed.

The relay lead sees on Sol 79 that a single MRO pass is available in the evening, and determines that it wasn't split (but assigned to SRL instead of M20) because pass splitting was only turned on for ERO (and not the other orbiters) in this allocation. The designer suggests that SRL will use too much decisional return anyway (since only 143 Mbits would be left over for M20), but the relay lead responds that SRL's needs will likely be reduced from 175 to 150 Mbits. Their implication is that the M20 decisional cycle on Sol 80 would be closed if this parameter to the scheduling algorithm were updated.

4.7.2 Scenario: Surface lead assessing a single decisional cycle

The **surface lead** for SRL wants to know how much data they will receive from operations on Sol 52, and when it will arrive (Figure 4.6). They create a query for passes with high return volumes ('RET Optimized RET DV') in the Mars evening ('Pass Start LMST'), as soon as the vehicle has finished executing its instructions.

They identify three return passes (closing brackets), all supplied by ERO. The first of these is the decisional return pass (marked in red and with a latency tail), which is split between SRL and M20. They inspect each pass by hovering over it, once in the SRL track (which has a return volume of 411 Mbits) and once in the M20 track (which carries 330 Mbits), verifying that both missions get enough return volume to begin operations planning (i.e. their decisional return needs are met), and that the total pass bandwidth is as expected.

Finally, the surface lead performs a time conversion from UTC to PST to determine that after latency, decisional return arrives around 7am PST, which will make staffing for Sol 52 easy. They look at the navigation track and observe that pass pattern quality will degrade in 5 – 10 sols, as the execution window shifts later across the six ERO passes, causing high-quality return passes to become scarce.

4.7.3 Scenario: Diagnosing operational efficiency loss

The **algorithm lead** identifies a *threat to the mission* to address a concern raised by the **mission lead**, basing their analysis on a recurrent pattern of operational efficiency loss (Figure 4.7). They begin in a section of the allocation with many missed decisional cycles, identified by a checkerboard pattern in the navigation panel barcode (since operations occur every other sol in the worst case).

The algorithm lead sees that the forward pass on Sol 220 is late and occurs during the execution window (as shown by the red planning window in the background, which together with latency has pushed all the passes into execution), causing operations to be delayed until Sol 221. This causes most of the passes on Sol 221 to be wasted, appearing in the bottom unallocated track, as no new information is available when they occur, prior to the execution window.

The algorithm lead switches to the Earthtime representation of the schedule and observes that the six ERO overflights on Sol 221 (Day 226) end just after midnight UTC, which is bad timing. They suggest that the missed decisional cycle can either be moved to another part of the mission (by shifting the orbit of ERO), or resolved by finding another orbiter which can supply a return pass in the evening on Sol 219.

4.8 Evaluation

This section describes the evaluation of the MarsIPAN system. We first unpack the methodology we applied, and then describe the results, reflecting on the expert knowledge shared by our stakeholders [81], and what that helps us generalize similar

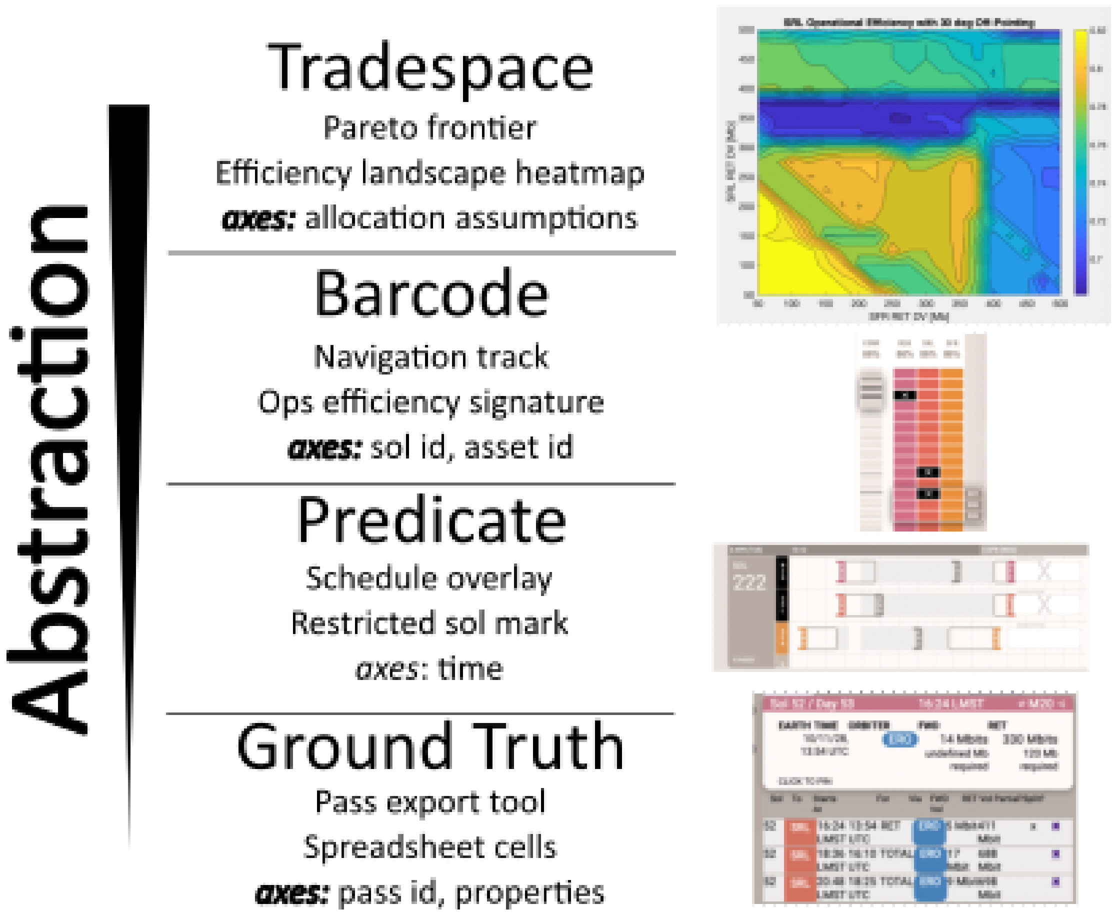


Figure 4.8: Decisional cycles in pass allocations participate in a ladder of abstraction, which we illustrate here with MarsIPAN widgets and other visual analyses. Our observations show that MarsIPAN users translate between levels of the hierarchy continuously, whether they are using MarsIPAN or otherwise accessing data sources. The ground truth (pass structure) implies the decisional cycles, which imply the tradespace (sensitivity analysis), that reveals how the users' chosen assumptions govern the efficiency of their pass allocations. Tradespace figure reproduced from Donitz et al. [36] with permission.

tools working on similar problems.

The MarsIPAN prototype has been released directly to stakeholders via internal GitHub Pages server. It is typical at the lab for prototypes of data analytics tools to be used in planning work. When MSR is being operated, years from now, we foresee tools based on insights from this prototype being used to coordinate teams.

4.8.1 Protocol

We applied a think-aloud protocol [92, 4, 5] that asked users to interact with the live MarsIPAN system using actual mission data. Users were given informed consent, and then walked through the basic application capabilities. Since many participated in earlier iterations of the work-in-progress application, they had basic familiarity with the capabilities, and so the tutorial focused on deltas since previous versions. We then asked users to think aloud while they interacted with the system, allowing researchers to observe their behavior, and to understand their underlying motivation for choosing specific actions. After the task, we conducted retrospective semi-structured interviews [12] to unpack our observations, and to ensure that we could dig below surface behaviors to make sure that we understood their root causes. Sessions were recorded, and later analyzed using emergent theme analysis [52, 23].

We conducted the full semi-structured interview with two expert users, representing the relay perspective and the surface team perspective.

4.8.2 Findings

In this section, we interpret expert users' approaches to explaining scheduling decisions using the MarsIPAN visualization. We adopt the *ladder of abstraction* from the creativity support literature as a lens of power, allowing us to appreciate how much influence experts exert upon the schedule by adopting four different vantage points, in order of decreasing abstraction (Fig. 4.8): the [Tradespace] , the [Barcode] , the [Predicate] , and the [Ground] .

These four vantage points cut across the tasks of each user role, as they are linked instead to the MarsIPAN data abstraction. [Ground] truth consists of passes as defined in Table 4.1. [Predicate] structure is visible from the schedule as brackets around the execution window, representing decisional cycles. [Barcode] structure is the same decisional cycles represented in the navigation track. [Tradespace] structure describes the space of all possible scenarios (i.e. inputs to the scheduling algorithm), which .

We have divided each expert interview into separate tasks, and grouped excerpts which are thematically related. Each utterance is tagged with a level of abstraction, demonstrating how each task circulates between the four levels of abstraction. We observe that experts control the *vertical movement* of their own reasoning in order to control the schedule: they drill down into pass details, and then zoom out to consider broad questions of scenario design.

Experts readily explain scheduling decisions in terms of an appropriate level of abstraction

The core use case of MarsIPAN is for experts to identify problem spots in (a set of) allocations generated by MARMLADE, and to intervene according to their own responsibilities. Although each MARMLADE allocation itself is optimal according to a fixed set of constraints, real-world scenarios involve a great amount of situational variability in most constraints. Whereas domain experts are free to ignore variability that doesn't impinge on a given decision, implementing open-ended variability into MARMLADE algorithm would involve (impractically) writing all of it down in full technical detail.

A sketch of variability in assumptions follows. For example, surface teams may respond to a problem by reducing their data volume (DV) needs on a given planning cycle. The relay team may respond to the same problem by changing the operating parameters of a given orbiter.

Locating Passes The relay expert used the navigation track to identify the 40-sol period over which the Earth Return Orbiter (ERO), which passes over the surface assets every 24 Earth hours, cycles fully across the Mars day. During the span of time in which ERO overflights completely overlap or precede the operations window (i.e. Mars daylight), the expert diagnosed an alternating pattern of restricted sols between M20 and SRL (due to the scarcity of return passes).

> [Ground] Trying to see where I am in the pass pattern. So there's my 6th [ERO overflight pass]. I think it's hard to transition between the passes that are allocated and the ERO passes that are hidden down here [in the

> unallocated track]. I have to look up here for the first two, down here for three, and up here for the 6th. It's not super intuitive.
>
> I have to look up here for the first two, down here for three, and up here for the 6th. It's not super intuitive. Yeah, I only see one MRO so where are the others? There should be at least two opportunities, if not three or four.

The relay expert told us that, since unallocated passes occupy their own track in the allocation schedule, it was hard to find the overflight pattern they expected. Although we chose to facet the schedule by surface asset in order to make it more legible to surface teams, our relay expert did not benefit from this additional layout step.

Diagnosing a missed uplink One task that the relay expert performs is to preserve passes in order to give the mission more buffer against anomalies, by preserving valuable passes. The relay expert used MarsIPAN to investigate a pattern of missed FWD commands in a pass allocation.

> [Predicate] For whatever reason they can't command until after their execution window [on the 5th sol]. I think this issue is more on sol 6 than sol 7.
>
> [Ground] And the ERO pass that would be available is right in the middle of the execution window [on Sol 9]. Understanding MRO's split pass capability. But it would have a limited - I see. [sol 9 19:15 lmst]

They used their knowledge of the specific characteristics of MRO, which provided the evening pass in this situation, but could not be split between both surface missions (so that one became restricted). They observed this as a recurring pattern through Sol 99, and pointed out that the allocation should alternate which surface mission was left off of the evening pass.

> [Ground] So I wouldn't allocate this [MRO pass] actually with only 71 Mb. I'd just use this ERO up here.

> [Barcode] M20 and SRL should probably be alternating back and forth because M20 is [restricted on Sol] 99 and SRL is at 94. [Scrolling through 40, 81, 120.]

Getting oriented in a tradespace The surface expert characterized MarsIPAN as a tool which increases the legibility of their existing tradespace visualizations. They used our allocation browser to select an appropriate worst-case scenario to investigate, and were able to rapidly find the sols of greatest concern, incorporating their fine-grained knowledge of the mission timeline.

> [Tradespace] I'd use MarsIPAN to understand which allocation fits the scenario we're testing [and building our schedule around].
>
> Let's do worst case. Min staffing, earliest overflight, latest landing. I also like these thresholds, which can give me any of the runs with an SRL ops eff greater than 92 regardless of the orbiters.
>
> [Barcode] It looks like I can see where the clustering [of restricted sols] is that I should be concerned. Like this chunk is in the prime mission.

They described the process of drilling down into scenarios as a mixed-initiative intervention, which cannot be captured by fixed policies, but need to be solved by experts within the context of the scenario.

> It's not easy to make policy-based decisions [for specific scenarios], a user needs to come [into the mission planning environment] and adjust it.

Our interviews solicited expert knowledge about which operational assumptions are hard (i.e. difficult to change during operations, such as orbital geometry), and which are soft (e.g. day-to-day data volume needs). In practice, we observed that a hard constraint from the perspective of surface planning may be a soft constraint from the perspective of a relay operator, so that ultimately, most constraints are soft from someone's perspective.

At the same time, we saw that individual domain leads were able to explore larger tradespaces than the mission lead chose to pursue. This organizational strategy can be characterized as 'branch-and-bound exploration', with bottlenecks present at each stage of analysis (i.e. of the tradespace of *a subset of* possible constraints) that is shared out to other stakeholders. As a visualization tool, MarsIPAN intervenes at the boundary between domains (i.e. during communication between stakeholders), by enabling visual analysis of the results from 'branching' analyses across many MARMLADE runs.

Experts use the schedule to trade slack between teams, making each others' jobs easier at critical times

MarsIPAN enables experts with a variety of domain backgrounds to use a common visualization of the schedule of operations they all share.

Comparing allocations In order to address an expert's question, we prompted them to load an allocation assuming 24-hour staffing rather than 18-hour staffing. However, this excursion disrupted their context, because they had to navigate away from the allocation and into the alternate one.

> [Predicate] Don't recall the day [on the other allocation] - here it's almost like you want a head-to-head.

Currently, the most practical solution is to open both allocations in side-by-side browser windows, each containing a MarsIPAN instance. However, we plan to implement an operation to change the allocation assumptions during certain sols, as bounded by the end of the execution window, by importing sections of alternate versions of the allocation from within MarsIPAN.

Using visualization to communicate The surface lead noticed that the DFE FWD on Sol 7 had been allocated during the execution window, meaning that the planning window would not have been long enough otherwise ('we don't have margin'). This prompted them to switch from Marstime to Earthtime. They identified the decisional RET arriving at 5:30am PST, meaning the nominal staffing start at 6am had led to the planning overrun. They were satisfied that actual staffing would not miss this sol, because during the first two weeks, their team intended to staff on Marstime already.

The surface lead identified that the second MRO pass on this sol was missing, and wanted to ask the MARMLADE algorithm designer why it had been omitted (as a low-quality pass). They flipped back to Marstime, observing that the restricted sol mark was no longer grayed out - as the 'restricted day' had contained no passes allocated to SRL, while the restricted sol did.

> [Tradespace] Great for asking the MARMLADE lead questions about their algorithm.
>
> [Predicate] I'd take a screenshot and export the passes [to use my filters on them]. I'd use the information that's here and call it out. That orients it toward the problem or concern we're talking through.

Bringing schedule data into team-specific tools The surface expert observed that the use of sol/day abstractions in our tool was unusual, as their tools simply had the two parallel time axes with no additional faceting. One of the intrinsic complexities of Marstime planning is that decisional cycles can span one or two days. They suggested augmenting the secondary x-axis in MarsIPAN with hour marks (in the parallel time axis), because the existing marks at Earth (or Mars) midnight are not fine-grained enough.

> [Ground] The planning tool consumes [temporal event] data all the time. We have the ability to add state timelines underneath, and write constraints to check for conflicts.
>
> [Tradespace] MarsIPAN is the tool we [would] use to sanity check that what we're saying is correct. More of a strategic planning tool and a development planning tool.

Strategic planning is when teams find days they won't be able to operate, in order to determine their baseline operational efficiency in the reference scenario (where nothing has gone wrong). Development planning lets teams understand which capabilities in the available orbital assets and surface assets are most desirable, and should be prioritized for development by the time the mission launches.

Experts want to start with plausible scheduling decisions, but to control critical decisions themselves

MarsIPAN enables experts with a variety of domain backgrounds to interpret and control the results of the MARMLADE algorithm. We refer to this relationship as algorithmic trust, following a growing literature that includes visualization as a specific site of intervention [8]. Often, experts reason simultaneously about the structure of a single allocation, and what consequences a change to upstream assumptions would make in that structure.

Evaluating algorithm assumptions and decisions The relay expert characterized some of the ground truth entering into the MarsIPAN schedule as in fact flawed, due to coarse-grained upstream assumptions.

> [Ground] [on Sol 193] So here [the return pass is at] 14:44. Seems like [MARMLADE] allows some overlap [with execution] but not all the time.

> [Tradespace] [on Sol 192] In reality this is conservatism upstream from MARMLADE. This pass is likely to be 55 Mb [rather than 33Mb], typical for these low elevation passes.
>
> [Ground] I wanted to tell it this DV is too conservative, and we actually get 55Mb from MRO. That way we can split the pass and operate ...

They wanted an option to recompute the allocation from within MarsIPAN with updated assumptions, or at least to manually override the allocation of a single pass.

> [Ground] I want to see if I could split this [MRO] pass so that I can get some amount for SRL [on this sol with bad ERO passes].
>
> [Tradespace] What would I need to change in the system to get this DFE pass before the execution? Because I can't actually force it to recompute, or force planning.
>
> [Ground] I almost need to override the DV for MRO [sol 191] and force allocate this one.

However, even the read-only interaction we have provided enabled the relay expert to rapidly assess the quality of the algorithm's decisions, and focus their effort on sections of the allocation where it experienced difficulty. MarsIPAN was able to bridge the gulf of evaluation for MARMLADE assumptions, which makes us optimistic that we can bridge the gulf of execution next.

> [Barcode] I don't fully trust the algorithm to do a better job than me, and I can find opportunities. [...] For these sections I would let it do its thing and then come [scrolls] down here and fix it.
>
> [Tradespace] It makes [MARMLADE inputs] much easier to work with and digest, rather than something that's hidden away.

Scope of software engineering and integration MarsIPAN is designed to participate in workflows with MarOS upstreams (incorporating both geometry and relay) and various downstream tools. On the surface planning side, these include timeline scheduling tools and pass selection tools, which vary between the surface mission teams. On the relay side, expert analysis and discussion will typically result in changes to the upstream orbiter configuration assumptions.

According to our relay expert, MarOS is similar in scope to MarsIPAN and MARMLADE together. As MarOS is maintained by a team of 6–10 developers at a given time, we anticipate that maintenance of a MarsIPAN-like tool in production will require a comparable amount of developer effort.

4.9 Discussion

In this section, we discuss how we pre-processed MARMLADE outputs in order to unpack the key metric of operational efficiency.

4.9.1 Design practice reflection

Design studies identify sites of visualization intervention into user workflows. The visualization expert's job in facilitating the design study is to understand the experts' reading methods in a way that can be implemented in a visual analytics tool. By operating this tool, the experts will be able to rapidly produce figures that are easier to read than anything produced from a domain-agnostic tool.

Writing in science and technology studies, David Ribes [101] observes that experts working across domains do not create shared ontologies, but instead create automated translations across them. The work of data visualization, is to create such a translation between the 'paper tool' (a conceptual domain-specific representation) used by the experts, and the dataset as it exists (whose schema reflects upstream data sources and their limitations).

Building the visualization on top of an existing system gave us both ontological traction and diverse stakeholders. Developing MarsIPAN enabled us to build trust between the MARMLADE developers and their stakeholders. We needed, for example, to establish a shared vocabulary - as when one expert explained that the ‘operations window’ was an ambiguous thing to call the execution window on Mars, because it could also refer to the planning window that the operations team on Earth would take advantage of.

Starting with early MarsIPAN prototypes, we used actual MARMLADE allocation proposals, which are characterized by a single table of 3000 passes, and some metadata about operational assumptions. This was advantageous, as we identified temporal scale issues in the first week of working with the data. After consulting with the MARMLADE team, we decided to prioritize the 3-sol schedule view throughout development (Fig 4.3), which is supported by 300-sol and 30-sol navigation views.

Yet the allocation proposals themselves are the product of significant systems design research by the MARMLADE team. For example, the notion that orbiter overflights yield a certain number of relay passes of fixed duration is a simplifying abstraction [112] compared to the geometric data that the relay team actually reasons about. Based on this collaboration, we argue that visualization researchers and systems researchers are natural allies.

We identified operational cycles as the unit of reasoning by understanding how the key metric is calculated. MarsIPAN uses decisional cycles as the unit of abstraction for the pass allocation schedule; in other words, it partitions the individual relay passes. We were able to make this decision during early prototyping (as Conlen et al. suggest [29]) because the operational efficiency metric measures which sols had closed operational cycles. MARMLADE sifts over sequences of passes to reason about how it can close as many operational cycles as possible, and thus incorporates information about the arrangement of passes (i.e. the gaps between them), even though its output is an annotation of passes.

Building the visualization in a reactive framework gave us flexibility to respond to changing user requirements. MarsIPAN was developed using the Observable runtime[2], which is a light-weight reactive prototyping framework for client-side Javascript applications. It is designed to integrate both HTML-based and SVG-based visualization widgets, which are typically bound to datasets using the API of D3.js [15], into larger applications with persistent selections.

Using SVG widgets allowed us to rapidly iterate on interaction design within the navigation and schedule views. Using Observable allowed us to incorporate Arquero query selection using the pre-existing Data Wrangler as its frontend. We also used chart grammars in early prototypes, although we would decide that the relationship between Mars and Earth time should visually be represented using its own chart type. Implementing the intuitive expectation of 'jumping to' a bookmarked pass, for example, required the bookmarked passes widget to programmatically change the scroll state of both the navigation track and the schedule.

[2]The `@observablehq/runtime` library is open-source on GitHub.

4.9.2 Explainability in scheduling

Explainability has been widely circulated as a desirable characteristic of AI systems including schedulers. Prior work on formal definitions of explainability has been focused on classifiers and single-sorted data tables [106], in order to define further desiderata in explainable AI systems. We aim to strengthen this work by grounding it in Phil Agre's critical technical practice, following the provocation of the first two workshops on human-centered explainable AI (HCXAI) [39].

HCXAI is especially interested in identifying in particular issues related to governance and AI-supported decision-making. These calls echo Susan Leigh Star's writing about *due process* in the structure of scientific communities, addressed to the AI community in 1989 as an analogue of the frame problem [123], i.e. whether any given choice of abstraction is relevant to the situation that an AI system is used to reason about.

The HCXAI workshop organizers have also asked [38], is explainability a property of the expert, the model, or both together? We can ask this question post-hoc using their five 'components of human rationales' to evaluate explanations of decisions written in natural language.

Returning to the formal definition of explainability by Rosenfeld et al., we propose to extend it with granularity, i.e. mixed levels of abstraction. In moving from classifier to scheduler applications, we reveal that multiple data sorts and their relations participate in the process of explaining a schedule.

We are motivated by recent work on due process in visualization systems (in particular, creativity support tools) by Li et al. [73] which in part argues that experts switch ‘horizontally’ to using another visualization system when their current one has inadequate granularity of abstraction to enable movement ‘up and down’ the ladder of abstraction [129] (as adapted in Fig. 4.8).

4.10 Conclusion

In the broader context of software instruments, MarsIPAN demonstrates that complex sociotechnical ecologies genuinely benefit more from well-delineated boundary object prototypes than from producing arbitrary visualization systems. We also reflect upon the design study process, which took place across several months of work, and supported by different stakeholder groups at the Lab at different phases of prototyping.

MarsIPAN makes it possible to evaluate a scheduled relay pass allocation at a glance. Because many teams will coordinate across the same schedules, our visualization of the schedule turns a complex coordination problem into a simpler one. We avoid treating the schedule dataset as a fixed object, but rather as a canvas ready for annotation. Through schedule analysis systems like MarsIPAN, we foresee that diverse Lab stakeholders will be able to successfully consult and iterate upon algorithmically-assisted pass allocations in a joint Mars-Earth communications schedule.

Chapter 5

DendryScope: Narrative Designer Support via Symbolic Analysis

This chapter is based on the paper "DendryScope: Narrative Designer Support via Symbolic Analysis" coauthored by myself, Autumn Chen, and Adam Smith [95] In this chapter, "we" refers to the team of authors.

Quality-based narratives (QBN) are hypertexts with extensive implicit linked structure. The observation of one passage can have non-obvious long-range implications for the reachability of other passages, which poses an authoring challenge. To help narrative designers address this issue, we produced an interface which visually summarizes all possible playtraces. Our interface leverages answer set programming to produce a query language over possible playtraces, allowing narrative designers to drill down to interesting scenarios. We introduce this interface through the DendryScope tool, which accepts most QBNs written in the Dendry language. We evaluated DendryScope by interviewing four narrative designers as they used the tool to explore *Bee*, a notable

QBN written by Emily Short.

5.1 Introduction

Narrative designers are interested in creating works of interactive digital narrative (IDN) that can be enjoyed across multiple playthroughs by many readers, who are given some agency in shaping the arrangement of scenes they will encounter, and even led to believe they can control the outcome of the story.

Extensive prior work in IDN discusses the authoring challenges inherent to this medium [84]. Calls for tool assistance in IDEs [10] have thus far been taken up within the IDN community, especially by narrative designers with strongly computer science backgrounds. At times, the specific needs of playtesting been discussed in the AI literature, which we discuss further in Background. This chapter argues that *quality-based narrative* (QBN) is a particularly robust subset of IDN, and demonstrates the power of symbolic AI tools for addressing practicing narrative designers' actual needs.

In this work, we define *skeins of playtraces* in terms of answer set programming queries (ASP queries) that describe properties of certain playtraces. We have implemented a novel *creative interface* ([33]) for a subset of ASP queries, designed to empower narrative designers to reason about complex works of IDN. Building on the design space approach advocated by [119], DendryScope enables narrative designers to manipulate the design space of skeins in a given work of IDN. Our research implementation deals specifically with *quality-based narratives* written in the Dendry language [85].

This chapter introduces the four following research contributions, which each build on each other, and apply symbolic artificial intelligence to key challenges in the production of interactive digital narratives (IDN).

- A transpiler from Dendry to the ASP domain. Each space of solutions in the domain of a given IDN, given a certain query, is a *skein of playtraces*.

- The DendryScope query language, where each query is a set of ASP constraints authored by the narrative designer. We foresee query sharing between designers as enabling continuous integration for IDN, and other novel forms of testing and collaboration.

- An inventory of seven tasks performed by narrative designers who are playtesting an IDN, especially in the context of sculptural hypertext (with discrete passages and implicit links).

- The heatmap interface to a skein, which is a direct manipulation interface for novel DendryScope queries. This interface allows narrative designers without prior ASP knowledge to 'drill down' through the families of possible playtraces in a Dendry game.

Figure 5.1 summarizes these contributions in the context of a DendryScope-driven playtesting workflow, using the acclaimed IDN *Bee* [117] as the object of study. We conducted four expert interviews to investigate this proposed workflow. The designer tasks we gathered explicate how DendryScope queries address the challenge of reasoning about hundreds of thousands of possible playtraces.

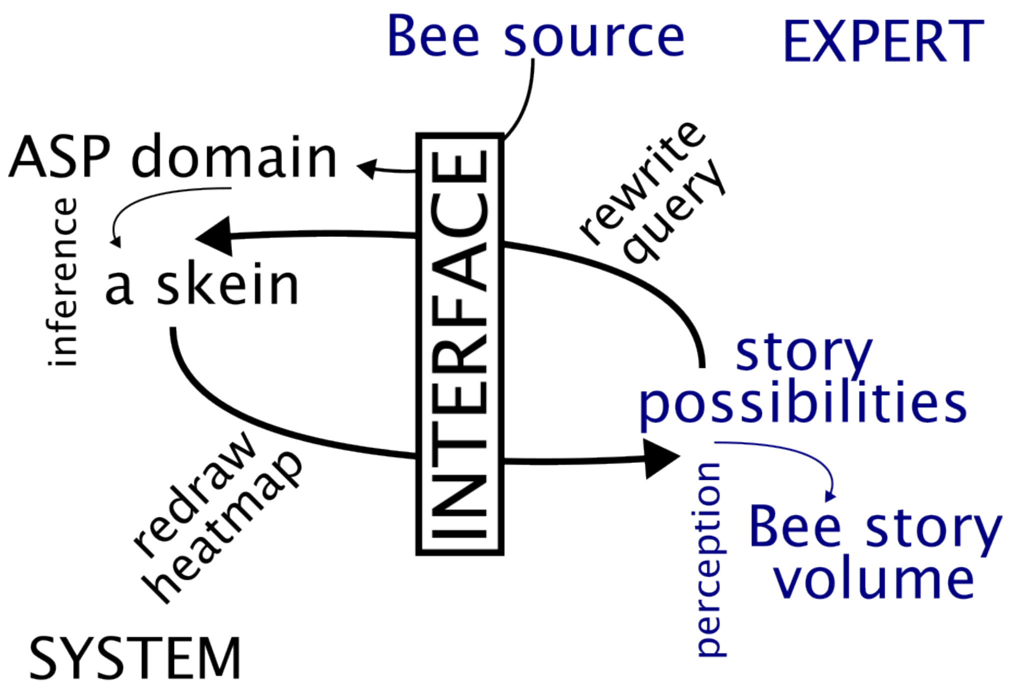


Figure 5.1: Diagram of a typical DendryScope workflow. Experts may use DendryScope to ingest the source of their QBN (in this case, *Bee*) and help them to envision the story volume accessible to players. Components shown in black are contributions of this chapter, including: a Dendry to ASP compiler, an isomorphism between ASP queries and 'skeins' of playtraces, and the heatmap representation of a skein.

5.2 Background

20 years ago, the authors of *Card Shark* [10] called for tighter loops between authoring and playtesting in sculptural hypertext. Our playtesting tool supports narrative designers' existing authoring strategies for IDN, which allow them to skillfully reason about player retellings, replay stories ([86]), and other forms of play in and around the *story volume* ([62]).

Grinblat describes the story volume of a given IDN as constructed in the player's mind through 'reparative play' in the course of one or more playthroughs [50]. The narrative designer, in the course of playtesting their IDN, is not only looking for bugs (i.e. obvious narrative failures), but also to emulate this process of story volume construction (Fig. 1). In this section, we will discuss the authoring formalism of quality-based narrative (QBN), contexts in which QBN is used, and how it enables automated symbolic playtesting that can assist with story volume construction.

5.2.1 Situating Quality-Based Narrative

Sculptural hypertexts are IDNs constructed from discrete passages, which are implicitly linked by preconditions and postconditions (Figure 5.2). A QBN is a sculptural hypertext whose world model is a string-int dictionary; this helps limit spurious complexity in the preconditions of each *storylet* (as discussed by [67]). Practitioners of QBN call these dictionary keys *qualities*; each has an integer *value* that changes over time, as the player traverses passages with relevant postconditions. The value of a quality defaults to 0, and is often glossed by a string.

Quality-based narratives form a rich narrative substrate from simple ludological underpinnings: every goal can be expressed in terms of some qualities that have attained certain values. This design constraint greatly limits ‘programming scope’, and can prevent it from taking over an IDN project [59].

Unlike traditional hypertexts with explicit link structures, connectivity between passages in a sculptural hypertext is unknown until playtime, because it is dynamically produced by the combination of game state (in terms of qualities) and the content selection procedure - i.e. which available passages from the current ‘deck’ (as a function of world state) are presented to the player in any given ‘hand’ of choices.

Many full-scale commercial games, such as *Fallen London*, *Hades* (per [97]), and *I Was A Teenage Exocolonist*, have been written primarily in QBN. Yet dedicated languages of QBN like Dendry are unusual; both QBN and sculptural hypertext are often embedded inside of other scripting languages, as Jones describes. For example, *Hades* is scripted in Lua, and *Facade* is scripted in ABL [77].

5.2.2 IDN Authoring Tools

A recent survey of IDN authoring tools [48] has found that narrative designers prefer tools that visualize the structure and connectivity of a story volume. They are also interested in tools that enable focused playtesting sessions, and in tools that encourage novice designers to experiment with branching.

In a broader sense, IDN authoring tools are a kind of IDE used by artists and teachers to explore an interactive medium. [80] have tested IDE features ranging from simple code linting, to inline code evaluation, to bidirectional manipulation of a generated artifact. DendryScope uses an intermediate heatmap visualization as the

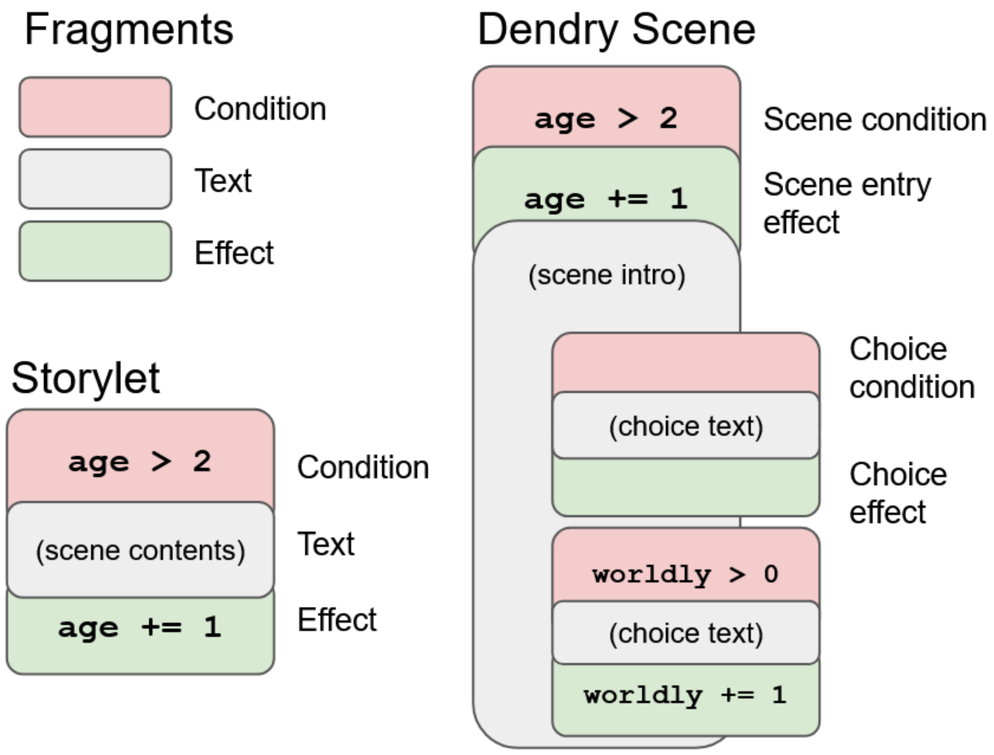


Figure 5.2: In sculptural hypertext, a storylet consists of three authored fragments: preconditions to check against state, postconditions to apply over state, and the passage of text that the player sees. In Dendry, a scene is typically a directed acyclic graph of linked storylets.

interactive surface for structured user input, such as placing 'pins' to refine a query.

Narrative designers must reason both as authors and as game designers. [21] describe narrative games in terms of two parallel goal hierarchies: narrative goals forming a descending chain of reasoning, and ludological goals forming an ascending chain of causality. For example, I might (narratively) want to open the door, and therefore need the key; at the same time, I would (ludologically) enable the 'open door' verb by getting my 'key' quality to 1 instead of 0. The narrative designer can cleverly align their player's narrative goals with the logic of which qualities they intend to track, as we will see in *Bee*.

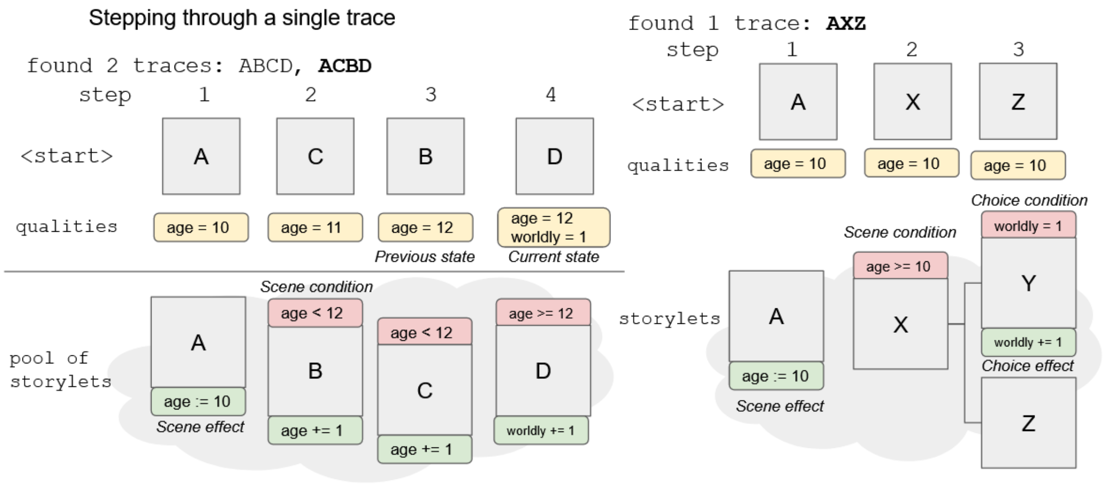


Figure 5.3: Each playtrace in the skein of a sculptural hypertext (in this example, a QBN) is a possible sequence of storylets. **Left:** A skein with two playtraces: passages B and C can be seen in either order, but both must be seen before D. **Right:** A skein with one playtrace: Either Y or Z must follow X. The precondition of Y is unsatisfiable, so AXZ is the only possible playtrace.

5.2.3 Automated Playtesting in Hypertext and IF

The goal of playtesting is often to characterize the possible playtraces of a given sculptural hypertext. Each such playtrace is a sequence of passages (Figure 5.3) that a player could visit, determined by the preconditions and postconditions of each passage, and assuming the content selection procedure presents each passage at the relevant step.

We do not claim that automated playtesting strategies incorporate any knowledge of narrative goals at all. Rather, they are designed to help narrative designers diagnose issues with a set of ludological goals intended to parallel those narrative goals. To this end, we review existing playtesting strategies used by narrative designers.

Traversal Statistics

An automated playtesting strategy can produce thousands of playtraces in a short time. As mass-generated playtraces tend to repeat large sections of play, to the point that reviewing them one-by-one is tedious, they should be aggregated or visualized. Indeed, statistical summaries of thousands of playtraces are commonly used by authors using ChoiceScript, Dendry, and custom narrative systems; an example of the latter is documented by the authors of *Ice-Bound* [44].

A conventional traversal summary strategy is to count the percentage of sampled playtraces that saw each passage. In extending this idea to their own system of sculptural hypertext, the *Ice-Bound* authors understand their playtesting strategy as a 'level profiler' which locates combinations of 'symbols' (i.e. those qualities a player could have or acquire in a given level) that are insufficiently 'fleshed out' by passages corresponding to that combination. In other words, their histogram visualization aggregates many playtraces over time and instance according to state - in this case, the availability of various combinations of symbols.

Traversal Diagrams

Recent work by [128] produced the *Story Validator* IDE for Twine games, which produces a skein-like view of playtraces by automatically generating the graph of possible links between passages. Their enumeration strategy relies on a greedy graph search heuristic, based on weights the designer assigns to each quality. They visualize the resulting playtraces in a flow chart, which aggregates all the possible orderings into a sequence of passages over time. Prior work in IDN authoring tools for Twine have explored this concept using heuristic traversal strategies [115, 91]. Similarly, the Inform 7 skein [90] is a flow chart produced from world state in an IF game, i.e. an IDN where passages are dynamically composed together according to complex world state. Inform's skein is constructed manually, as the author supplies each possible (partial) playtrace by playing through their own game. This makes it suitable for asking questions about choice points that ensue after the first few beats of the game, but not about skeins in the mid-game or late-game.

A symbolic approach, quite similar to the one we use in DendryScope, to enumerating Twine playtraces was demonstrated by [91], who summarized these in terms of values that variables could attain, and `end`-tagged passages that could be reached.

Symbolic Playtesting

Narrative design tasks involving the story volume can be facilitated by using modern answer set programming (ASP) systems to implement playtesting strategies. ASP is applicable when the possible values of game state variables can be modeled as a finite set, the length of interesting playtraces can be bounded, and the rules for how the state should be updated in response to each available player choice can be expressed with rules in symbolic logic. For example, there are solver-based authoring tools that support level designers creating difficulty progression in educational puzzle games [19]. In the IDN context, every solution of the answer set program is a distinct playtrace of the IDN.

Player behavior can be modeled as certain constraints, such as 'I will reach this particular ending.' This playtesting strategy has been demonstrated in *PlotEx* [98], a tool created by the author of *Hadean Lands* to guarantee that all four endings in his Inform 7 code were reachable. Each door in the game requires specific resources to open, and certain resources are consumed thereby, so it is not apparent which resources should be located behind which doors. If an ending did prove to be unreachable, then Plotkin would need to make specific resources available earlier or in greater quantity at certain points in the game; or otherwise, make certain doors barring the endgame less demanding to open.

In PlotEx, sets of constraints are understood as a kind of unit test, which can be run against the latest development build of an IDN in order to spot regressions. In Ludocore [120], these are called speculative assumptions, because they represent constraints on play styles that apply to some scenarios and not others. In the context of DendryScope, we will call these *queries*. All three of the above systems may be construed as story planners in the domain of a particular IDN or gameplay model, which produce (all possible) examples from that story volume.

5.3 Methods

As a software instrument, DendryScope transforms Dendry games from an abstract symbolic form to a concrete set of possible playtraces, which we define as the *skein representation* of the story volume. In this section, we describe the technical effort which allows us to compute said boundary object.

We now identify the elementary passages in Dendry's authoring language, and how they relate to the changing value of qualities over time in a given playtrace. Our symbolic analysis strategy is to enumerate all playtraces which satisfy a given query, which we will call a *skein*. The empty query corresponds to the universal skein, containing all possible playthroughs of the given QBN.

5.3.1 The *Dendry* Authoring Language

We chose to build the DendryScope prototype on top of the Dendry language (Fig. 5.2): it is narrowly scoped around QBN, its compiler is open-source, and the complete source of *Bee* is available. However, any language of IDN that can be expressed in terms of ASP rules is amenable to our overall strategy, and any QBN can be represented in the DendryScope interface.

The Dendry language includes three kinds of storylet:

- **Top-level scenes**, which are accessible from the player's hand of choices.
- **Linked choices**, which either link to further choices or exit to the player's hand.
- **The player's hand**, which contains a small number of top-level scenes whose preconditions are met. The player decides which to pursue next.

The postconditions of every Dendry storylet describe how to transform the state of qualities from one timestep to the next, influencing which storylets' preconditions can be satisfied on the next timestep. In *Bee*, none of the qualities attain values outside of a small integer range (from -1 to +24); in general, knowing the range of each quality is useful to speed up ASP grounding.

Every Dendry scene file consists of scene and choice storylets, and has a

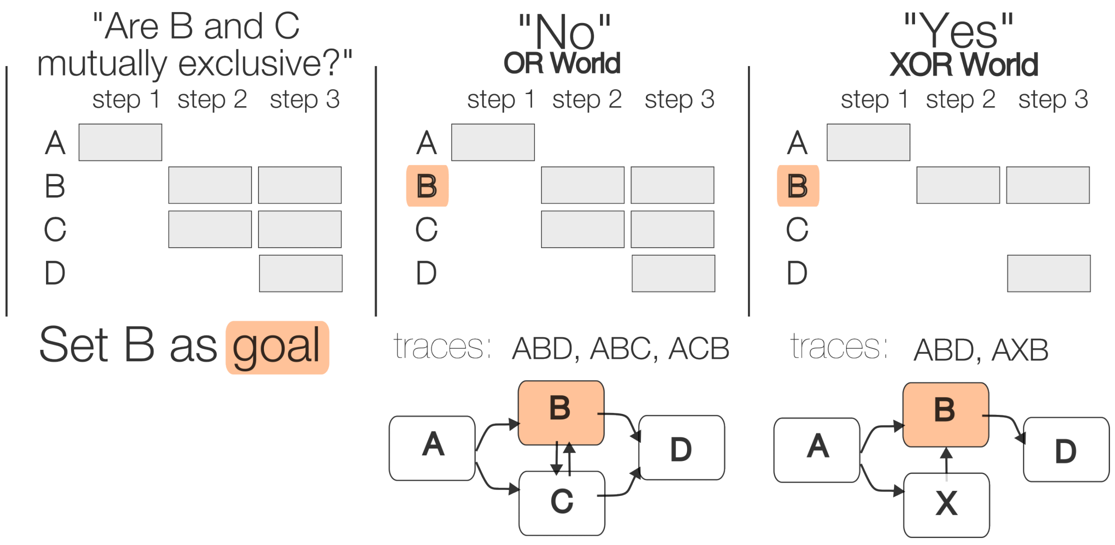


Figure 5.4: Worked example of reading a skein, and using a goal to discover implicit structure from subskeins. Four rows correspond to four Dendry passages. Each grey cell shows that some traces hit that passage at that timestep. We set 'B' as a query goal, and recalculate the skein (which now contains only playtraces visiting 'B'). Now if the playtraces 'ABC' and 'ACB' exist, then B and C are not mutually exclusive ('OR World'). Conversely, if C disappears from the skein, then B and C must exclude each other ('XOR World'), and (in this case) some offscreen passage X explains the trace that sees B at step 3.

Markdown-like representation[1]. These correspond to scenes in the sculptural hypertext system Lume [76], whose authors likewise chose to embed branch-and-bottleneck choice structures (i.e. DAGs) inside of content selection, as a sensible unit of authoring.

5.3.2 Reducing *Dendry* Traversals to ASP

In the literature of answer set programming (ASP), it is common to sketch an ASP formulation in terms of which logical facts the solver is allowed to nondeterminstically *guess*, which facts the solver must *deduce* from others, and which potential facts the solver must *forbid* from appearing in solutions to be enumerated.

[1]For example, scenes in *Bee* are represented by `.scene.dry` files found in its repository: `https://github.com/aucchen/bee/blob/master/source/scenes/church.scene.dry`

In our formalization of Dendry, we say the solver must **guess** exactly one passage to select for display among those passages that have their preconditions satisfied by the current quality values. From the sequence of selected passages, we **deduce** the evolution of quality values over time, influencing which passages are available.

Finally, we **forbid** solutions that do not respect the user's query conditions: we reject solutions that do not satisfy all *goals* (scenes that need to play at least once), *poisons* (scenes that must never play), and *pins* (scenes that must play at least once on the specified timestep).

DendryScope is implemented using the `clingo` ASP solver [46] through its WebAssembly port [87]. Rather than asking the solver for just one satisfying solution, we systematically enumerate a few hundred solutions. These concrete examples allow the designer to observe differences in state between playtraces.

Applying `clingo`'s brave enumeration (a form of projected enumeration that efficiently computes the union of all of a program's answer sets), we also obtain the complete set of which passages could be seen at which timesteps while abiding by the query conditions, without directly constructing every playtrace.

Full implementation details, including our exact traversal strategy and our method of tracking integer-valued quality variables over time, can be found in our system's source code at: `github.com/JazzTap/DendryScope`.

5.4 Visualization in DendryScope

Now that we have produced a skein from a DendryScope query, we must complete the software instrument by displaying these playtraces on the screen. Because story volumes are expressed in terms of state over time, we produced a heatmap of scene/timestep pairs observed across all playtraces, so that scenes visited at certain times (e.g. during the introduction) appear in certain parts of the visualization (i.e. towards the upper-left of the skein). In addition, we equip the rows and cells of this heatmap with query-authoring operations, yielding a set of 'brushes' with which to manipulate the query artifact.

In addition to formalizing the QBN domain, we created a frontend for query design via direct annotation of playtesting information in the form of a skein, i.e. the set of playtraces satisfying a given query. The visual skein interface makes it easier to read and write DendryScope queries than working from scratch. We foresee the narrative designer loading their own Dendry story, but for evaluation purposes, we pre-compiled *Bee* from Dendry to ASP.

5.4.1 Visualizing ASP Queries as a Skein

Upon loading their story in DendryScope, the designer will see a *skein* corresponding to their QBN, similar to Fig. 5.4. In our demo, this skein contains the first 20 steps of playtesting results for the empty query (i.e. 'What passages can I see?'). From here, they may relax the time horizon (as we chose an arbitrary value, to avoid timing out on long QBNs), or refine the query by directly manipulating storylets and steps in the heatmap.

The skein is represented by a heatmap whose cells correspond to the timestep vs. current storylet. Each cell counts the number of concrete playtraces that have seen it, and more playtraces corresponds to a darker blue (Fig. 5.5). This number can be zero, because cells may be observed by brave enumeration without corresponding to a concrete playtrace.

Because scenes (and all storylets therein) are automatically sorted by the first time they are observed, the skein appears linear where the player is certainly in a given scene at a given time (i.e. where the story volume is constricted), and appears diffuse where there are many possible passages the player could be in.

The initial skein corresponds to the empty query, returning all possible playtraces. Adding terms to the query produces subskeins (Fig. 5.6), corresponding to a more specific scenario. Each DendryScope query produces a fixed skein, up to four tuning parameters: a timestep horizon, a maximum number of constructed playtraces, a maximum iteration count on bravely-enumerated playtraces, and a timeout.

DendryScope includes graphical tools for browsing scenes by their tags, and for viewing complete scene trees. In the scene tree widget, storylets whose title is highlighted in blue were visited by a concrete playtrace in the current query; storylets highlighted in grey are possible and have been enumerated, but not constructed. Storylets that are not highlighted cannot be reached given the current query.

5.4.2 Visualizations of State across Related Playtraces

Each cell in the skein corresponds to a certain passage at a certain timestep in a set of visiting playtraces, but each of these playtraces has its own world state. In the skein, we broke out the 'current passage' as the most salient element of world state, but many more visualizations of this dataset are possible. For example, the narrative designers we interviewed wanted to know what values could be attained by specific

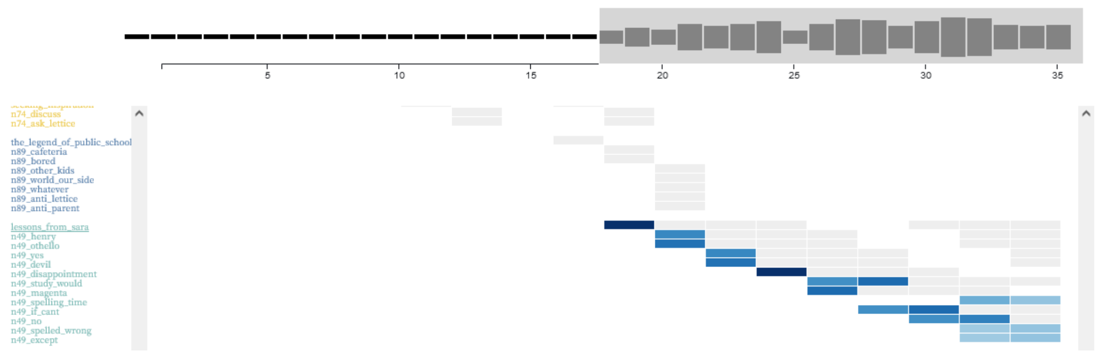


Figure 5.5: Screenshot of the DendryScope visualization of a skein in Emily Short's *Bee*. The user has clicked on the passage `lessons_with_sara`, setting it as the *goal*: it must be seen in each playtrace matching this query. This skein has been zoomed into steps 18 – 30 using the histogram (counting distinct passages at each timestep) on top. Gray cells show all possible passages at each step across all playtraces. Cells highlighted in blue have been observed in a concrete playtrace, with a deeper blue for more playtraces hitting that cell. The expressive range [121] of a family of playtraces hitting a certain cell can be inspected by hovering over it.

qualities at various cells in the skein.

The preliminary state visualization widget in DendryScope is a heatmap of quality versus value, describing the final observed state of each playtrace that touches the given cell. Although it is clearly more intuitive to produce the state of each playtrace as observed at the given timestep, the time cost of writing all state at all timesteps of all playtraces was excessive in our naive implementation. Besides, the AST of the IDN provides enough information to reconstruct (up to an arbitrary timestep) the state of every playtrace from its ordering of passages, although we did not implement this function.

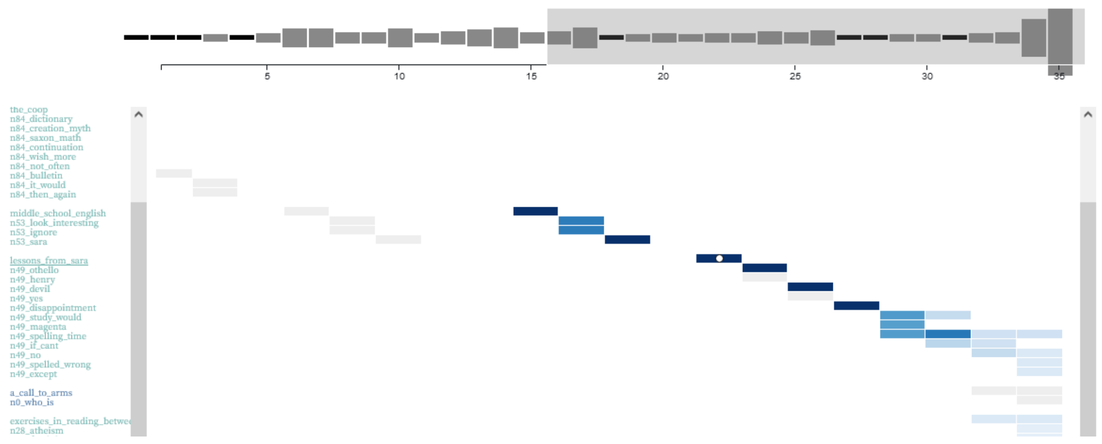


Figure 5.6: The result of refining the previous query (Fig. 5.5) by *pinning* the `lessons_with_sara` visit to step 28 (rather than step 27, the earliest possible moment). The other two scenes visible in this part of the skein have changed. Moreover, the 'cloud' of every other step at which we could enter the scene has now vanished. We confirm that `lessons_with_sara` begins linearly, and branches near its end. The skein fans in toward `lessons`, and fans out from it.

5.5 Designer Tasks

We conducted a data-first design study [93] to identify narrative designers' existing playtesting strategies, taking *Bee* itself as our dataset (shown in Figure 5.5). *Bee* is a QBN written by Emily Short in 2012, which follows the life of a home-schooled girl who is training for a spelling bee. Designers used the prototype DendryScope interface (Fig. 5.1) to read skeins and write queries according to their interest.

We recruited five narrative designers for these interviews, using a snowball sampling technique. All five narrative designers (including co-authors) were familiar with reading sculptural hypertext in videogames such as Fallen London and Hades. Each designer has written games in languages that implement sculptural hypertext, including ChoiceScript, Inform 7, and languages of their own design. Five of the six designers were familiar with *Fallen London* [42], a long-running work of QBN which

Emily Short has worked on. Two of the designers were familiar with *Bee* itself prior to the study. A mixture of three academic programs and three industry backgrounds are represented across the five designers.

One designer became a co-author on the paper from which this chapter is derived; their interview was omitted from coding as such. By coding the remaining four interviews, we identified seven tasks that narrative designers perform while writing or revising IDN.

Narrative Designer Tasks:

T1: Interpret the game's source code as a *story volume*.

T2: Answer story progression questions about the story volume. Is this passage reachable?

T3: Model player behavior as it influences the story volume. Are there rare or difficult-to-reach passages that might become player goals?

T4: Identify subsets of playtraces that correspond to certain player knowledge, or lack thereof.

T5: Understand how subsets of playtraces correspond to the query, and rewrite the query if it does not match the intended scenario.

T6: Discover how long it can or must take the player to reach a certain passage, including passages near the end of long stories.

T7: Develop flexible and powerful authoring patterns, such as *scenes* and other branching structures; or *menaces* and other kinds of qualities.

5.6 Evaluation

Our semi-structured interview process was designed to gather information about the four narrative designers' experience working with DendryScope for the first time. We wanted to understand whether the underlying ASP solver could be controlled by designers without prior solver-specific experience, through their own domain knowledge and our data-driven interface.

The first 10 minutes of each interview introduced *Bee*. The interviewer then asked the designer to describe structural features of *Bee* in Dendryscope's skein view (Figure 5.5), up to and including endings of the game. The interview takeaways below are organized according to the seven tasks we identified from the interviews.

5.6.1 T1 - Reading the Story Volume

The DendryScope skein visualization is "like an extra sense," said one designer. Another designer reported, "The tool might be able to surface combinations of things much more easily than a human could, because of the sheer number of combinations possible. The exhaustive approach that a machine can take allows you to see the possibility space way easier than just playtesting it."

In this chapter, we have treated traditional playtesting as a form of search over skeins. In practice, experienced designers described playtesting using language about the player experience. "If I get here, what are the qualities I need to have had?" We found that, regardless of what combinatorial framework or experiential goal is being interrogated by a particular narrative designer, the skein allowed us to discuss both specific passages in the context of the game, and possible ways in which players could have reached those moments.

5.6.2 T2 - Answering Story Progression Questions

We found that the designers were not only anticipating specific routes through the game, but also that they were interested in supporting variation along any given route. “I almost want to be playing [...] whatever the normal way to do it is, and then having this as some kind of ‘what could have happened’ off to the side?”

One of the designers recounted their process in establishing the consequences of early player choices. They understood one such mechanic as a quality that can build past a certain threshold, and were interested in discovering what values that quality can possibly attain at certain points in the story.

> “Let’s imagine you were writing something where the player was able to take on a role, but there were a couple of different roles possible. If you had challenges which you could solve by punching your way through or talking way through. Then, seeing it’s not actually possible for my punching score to be very high in this area. I didn’t know that [either] I shouldn’t have had this choice here, or I should have given more opportunities to raise [punching] earlier.”

This designer is interested in not wasting authoring effort: if a scene in the story responds to a quality past a certain threshold value, then they want to make sure that threshold is actually attainable, so that the branch is reachable. They said later, “It’s kind of a mystery at any given point in the game, what [the range of values for this quality] might be. Being able to actually visualize that looks wonderful.”

5.6.3 T3 - Modelling Player Behavior

The designers understood that DendryScope queries are not themselves knowledgeable about player intent, unlike statistical models of players. One designer asked us, "There's going to be a disconnect between what the tool is modeling, and how players actually would play through, right?"

The purpose of the DendryScope interface, therefore, is to re-incorporate the narrative designer's awareness of what goals players might seek or be guided toward in the story volume into their DendryScope query. We do not claim that visual interfaces must be used in every context that involves a query and its skein; but in the context of an IDE, they are a useful way to orient the designer in the story volume.

5.6.4 T4 - Discovering Structure in Skeins

DendryScope supports the designer by visualizing the skein of a query. Sculptural hypertexts can contain both branching structure and complex multicursal structures, similarly to IF games like *Hadean Lands* - and scenes can vary over time, due to the complex world model. One designer explained, "You can imagine a game where there was more of a puzzle element to it and you could loop around a bunch of times before figure out the option to move forward."

Another designer described their reasoning while creating a query: "And there are two opportunities, three, actually. [...] So if I pin something extremely late - it's going to give me just a little bit of variation."

"I think it really works like a cloud of possibilities. I said that that was quite explicit here, where what we saw before was quite a lot of different possible nodes. But now once we've reduced the [scope to] what we actually wanted to see, it linearized a lot."

5.6.5 T5 - Discovering Missing Playtraces

Designers were not only able to reason about the presence of playtraces which they did not anticipate, but also about the absence of playtraces which they did expect to find.

Describing the `doll` quality in *Bee*, one designer talked through writing a query for a certain absent playtrace, which the DendryScope skein is designed to prevent (because proving nonexistence is expensive, and yields little information):

> "I was thinking about when we did the doll thing earlier if I could pin, or like customize a pin, say I want [the doll] to happen at the 12th time step. And then try to run that. Right now I can't click it because [it] didn't do it. But okay, we knew that doll could happen earlier. So, if it just isn't possible, it would just give me, that's not possible. This is a deliberate interface constraint, right?"

Another designer described a hypothetical scenario in which an abundance of opportunities to increase some quality caused some content to be seen too often. They would then want to write queries that show the passage is rare:

> "Let's say you have a choice which is supposed to be possible, but only really possible if you've pushed your choices in a certain way beforehand. But let's say you actually just balanced it wrong, and everyone can reach that, or 75% of the time you can reach that."

5.6.6 T6 - Dealing with Long Stories

Although DendryScope solutions are complete, in practice, large story volumes with 30+ steps of lookahead slow down the solver. As queries reach 10+ seconds of runtime, the designer's process of exploration is clearly impeded. We tried to mitigate this issue with timeouts, but these produce truncated sets of solutions, which can be misleading.

We found out that some queries take too long to run, and interrupting runaway ASP is an important step in debugging. While this is a natural action in command-line IDEs, we did not have adequate support in the DendryScope interface during the pilot. When a query contains a contradiction, the interface should eventually time out; but the ideal threshold might be a minute, or it might be ten seconds, depending on the author's debugging strategy. For longer response times, we expect that both a fast approximate result and a time estimate for the full result would be useful.

One designer asked, "Is there any way to get real-time feedback from the ASP, so that you could be populating the view [as the solver runs]?" Because combinatorial explosion is intrinsic to sculptural hypertext, as a medium of orderings, we do not anticipate that all performance issues with large DendryScope queries can be optimized away. Therefore, it is important to learn from designers how they reason about and use 'poison' in queries to block off irrelevant kinds of playtrace.

5.6.7 T7 - Developing Authoring Patterns

In the course of understanding what qualities are used by Short in *Bee*, our designers identified certain qualities with mechanics from tabletop storygames that gate certain forms of progress. "It's Blades in the Dark that has [Countdowns], it's like the concept of [the spelling quality]. [...] A number of different storylets could progress that clock [...] when that clock hits 12 it unlocks new possibilities."

Whilst `spelling` is a progress quality in *Bee*, Short has also included menace qualities like `parents` (which increases when the protagonist defies her parents) and `lettice` (which increases when she clashes with her sister), that can lead to alternate endings of the game. Like in tabletop games, it may be more narratively satisfying to end *Bee* in a 'loss' of this kind; not only is the protagonist not going to win the spelling bee, but she is also going to grow apart from her family.

Short also includes thematic qualities such as `worldliness`, which increases as the protagonist gains experience with places outside of homeschooling (like the hair salon); and `poverty`, which increases as the protagonist discovers it is unusual to, say, barter homemade jam in exchange for pantry items. These qualities do not generally impact the reachability of any scenes (only certain choices), but instead tend to reward and motivate reflective choices which are thematically linked.

5.7 Findings

We divide the research contributions of this chapter thus: with implications for close readings of IDN on the one hand, and implications for authors of IDN on the other. Either way, the contribution of symbolic analysis is to support the narrative designer by visualizing the skein. Once visualized, this structure becomes more tangible to novices, and easier for experts to compare with their own intuitions. Ultimately, we view queries on QBN (and other forms of IDN) as a promising domain-specific language for future development.

The medium of ordering Ultimately, the passages in a sculptural hypertext are fixed; what varies is which ones the player encounters, and the order in which they appear. Ordering can constitute either diegetic or extra-diegetic choice, in the terms of Peter Mawhorter's choice poetics [78], sometimes ambiguously. For example, Aaron Reed recounts [100] his playthrough of Yoon Ha Lee's sculptural hypertext *Winterstrike* [70] in terms of progressing the `Ice` quality to reach a certain tragic ending:

> "I will need to build my own story of how I became the person who had enough Ice to reach this particular goal, out of the various opportunities in the story world [...] Perhaps if I keep investigating, I'll find better ways to increase this stat — better, for my own personal sense of my character and their morality."

Junius et al. call for research into the ordering of scenes as a mechanism of dramatic agency [60], describing many techniques used by theater practitioners to modify a fixed script into a dramatic experience for the performer - one that is rich with awareness of the moment, and all the meanings it could have. Like theater performers, IDN players do not have the power of an author over the story, but rather the power of interpretation.

We propose that an IDN is 'reactive' [76] when the player takes part in the juxtaposition of dramatic elements. Narrative designers do not only reason about IDN in terms of state and reachability, although these are powerful tools afforded by sculptural hypertext formalisms, but simultaneously about narrative causality and player agency.

For example, in *Bee* the player character is introduced to an English tutor named `sara`, after much debate by their `parents`. One of the first things they can ask Sara is whether she is really a feminist, which Sara takes care to explain is not actually a bad thing. Later, if the protagonist should run away from home (after tensions with their `parents` reach a breaking point), one of the characters to whom they can go is `sara`. This bespoke scene can occur only if the player has met her (i.e. the value of the `sara` quality is nonzero), and if they think this is the right choice for the playthrough.

In-situ playtesting DendryScope is designed to assist authors in maintaining their mental map of a story volume, given the source code of their game. A future, dedicated Dendry IDE should include features like syntax highlighting, scene and choice stub generation, and line highlighting upon compiler error, as well as access to the the queryable DendryScope skein.

Storing queries

We were asked to implement query saving features by one of the narrative designers. Without saved queries, there is no way to revert from a failed query (i.e. which produces an unexpected contradiction) to a good one. Moreover, query saving would double up as query sharing, as DendryScope queries (like raw ASP predicates) can be encoded as URL argument strings.

Enhancing queries

We envision pinning (Figure 5.6) as a strategy for guiding sampling within the story volume, producing colorful 'stains' of the skein corresponding to storylines within the IDN. Multiple colors of pin could represent divergence within a single skein. Also, pinning could be implemented at the level of scenes (rather than passages), allowing the relative ordering of scenes to vary without excluding alternative scenes of different length.

5.8 Conclusion

In order to demonstrate the value of software instruments to narrative designers, as a class of domain experts who work independently or in small teams, we designed and implemented the DendryScope playtesting tool. We chose to base our tool on the story volume concept of Grinblat et al. [50], which makes highly-generalizable claims that happen to be realized in the highly-tractable narrative structure of QBN [67], as discussed in this section. Finally, DendryScope would not exist without the public *Bee* source code, a high-quality dataset representing a QBN work of note.

Our work in this chapter demonstrates how symbolic artificial intelligence can be applied to the story volume of works of IDN, and help narrative designers address their authoring problems[2]. Conversely, we see this work as connecting narrative design expertise to the AI research tradition of plot generation, through the medium of sculptural hypertext.

DendryScope translates the designer's intent to examine a portion of the story volume, through direct annotation of the skein representation, into a suitable ASP query over playtraces. We have described 'goals', 'poison', and 'pin' constraints in this chapter. In the future, other types of queries can be developed in terms of ASP to extract more nuanced information from a given QBN.

From a practitioner's perspective, we aim to raise awareness among narrative designers of symbolic analysis as a debugging technique. It is easy to start writing queries in DendryScope: all five narrative designers we interviewed could develop simple

[2]Acknowledgements: We thank Bjarke A. Larsen, Leonardo Abate, Joey D. Jones, and Maxwell Joslyn for contributing their time and expertise as narrative designers.

queries after exploring the interface for about 30 minutes.

We hope to develop DendryScope further into a practical tool with integrated script authoring, built-in detection of unreachable scenes and critical-path scenes, and useful debugging feedback on complex queries. We believe that designers would readily adopt an IDE for sculptural hypertext with a playtesting capability that is as accessible as Twine and as powerful as PlotEx.

Part III

Synthesis

Chapter 6

Conclusion

I have sought to convince you that the study of expert practices is inseparable from the study of intelligent user interfaces. Recall that I proposed to answer three research questions.

1. **HCI**: What are the properties of sociotechnical systems, paper tools, and creativity support tools (CST) that make them useful to domain experts who need to share out data?
2. **AI**: What are the properties of programming systems, domain-specific languages, and knowledge representations that make them useful to domain experts who need to reason over data?
3. **CSCW**: How are experts' tools and systems shaped by power, resources, and agendas within an organization? Where do language-boundaries become reified into tool-boundaries?

Alas, the properties of software instruments themselves cannot be readily formalized and decomposed into simpler parts. After all, any practice involving a software instrument is an enactive loop: to cut the loop is to destroy its unique properties. Considering that all of the really interesting problems with data are also wicked problems, I must admit that software instruments are not one-shot interventions. Rather, the process of creating a software instrument is the means by which wicked problems are identified, stakeholders are brought into conversations with each other, and the possibility of solving non-wicked sub-problems is realized.

Thus, I must modify the question in order to answer it. Which processes of producing and evaluating software instruments are themselves supporting communication and reasoning? In the following sections, we will assess the impacts of visualization prototyping on communication needs, and the impacts of reasoning needs on interface design. Ultimately, we will construct a history of critical technical practice as a counter-movement to systems thinking, deeply sensitive to the challenges of communication across collaborating domains in both professional organizations and personal artistic practices.

6.1 Communication: From Systems to Theory

We saw in Ch.3 that boundary objects are ubiquitous in visualization design studies, both as inputs to an iterative prototyping process, and as the intermediate results of wireframing and implementation. Threats at all four levels of Munzner's nested model (i.e. distinct levels of abstraction) can be addressed by different boundary objects within this process. Through creating MarsIPAN, we observed that experts themselves reason over multiple levels of abstraction simultaneously, and use this fluency to fully exploit software instruments in communicating with many possible audiences.

Ch.3 is itself an example of new theory emerging from practice, specifically from conversations about MarsIPAN and its role as a boundary object; but also, the role of intermediate prototypes created during the design study process as boundary objects. We draw on design theory to re-establish that a piece of visualization software is not a one-off intervention that solves a communication problem. Rather, the process of working with a team for many months to produce the visualization software is itself the intervention.

Each of my systems is an example of a software instrument. The notion of artifact and brushes I used was broad enough to encompass these widely separated applications. We can now describe each system in terms of paper tools and boundary objects.

6.2 Reasoning: From Theory to Systems

We saw in the case studies (Ch.4 and Ch.5) that domain knowledge is central to the implementation of intelligent user interfaces. To solicit domain knowledge is a critical step in visualization design studies, either in the form of datasets, interviews, or ideally both. From this perspective, intelligent user interfaces and programming systems exist on a spectrum. In particular, we demonstrated with DendryScope that narrative designers can readily adopt static analysis of their own codebases by exploiting our direct manipulation interface.

With the aim of developing software instruments as a theory, I pursued these projects in order to work closely with domain experts from different areas who would use these particular instruments. The axes of scientist–artist and of institution–individual form the opposite poles on which MarsIPAN and DendryScope, respectively, sit: MarsIPAN serves the institutional scientist, and DendryScope serves the individual artist.

Software instruments are tools for collaborating with systems; they are not tools for letting systems make decisions instead of experts. After all, experts wouldn't need a visual analytics tool if they weren't going to verify the decisions. In practice, we see that every domain has contextual information that leaks out of the systemic representation, and it is vital to take up a theory of software instruments in order to mediate between sophisticated technical reasoning and the actual needs of stakeholders.

6.2.1 Case Studies

DendryScope is a static analysis tool in the domain of narrative design, with a direct manipulation interface inspired by the *story volume* as a 'paper tool'. Story volumes are the paper tool of DendryScope, and we claim that narrative designers intuitively reason about story volumes as a range of narrative possibilities. However, digital authoring tools often fail to represent the story volume, because it must be constructed using automated playtesting techniques, which are quite complex. For example, DendryScope does not perform psychological modeling of players, but only describes sequences of events – mere playtraces, – which a skilled narrative designer can proceed to interpret as different player experiences.

MarsIPAN is a schedule for building algorithmic trust, adapting a prior paper tool in operations planning. Its operational logics center upon the *decisional cycles* produced by an optimal pass-allocation algorithm, and the sensitivity analysis of that allocation algorithm. MarsIPAN will ideally save experts from doing 300 days of work out of a 320 day schedule, but it is not and cannot be a faithful representation of the entire design space with its myriad contingencies.

Both MarsIPAN and DendryScope represent the *design space* of possible **artifacts**, and equip that space with *brushes* associated with particular design moves through suitably spatialized operational logics. In these examples, the design space is either the set of (tentatively) optimal pass allocations associated with each operational scenario considered by Mars Sample Return (MSR), or a set of possible playtraces of a given quality-based narrative (QBN). The software instrument always acts as a map of the design space, and points in the design space – artifacts, – are model representations of the world. An instrument serves to inform the expert of the necessary limitations of both maps: the design space is bounded, after all, just as the artifact is like a box drawn around part of the world, an *atomized representation.*

6.2.2 Future Work

Here are a few concrete next steps that we were not able to address within the scope of either the MarsIPAN or DendryScope design studies. These projects represent foreseeable research trajectories minus external inputs. Even so, they take on a variety of different forms.

1. Developing bespoke exploratory programming support tools for languages of behavior. These arise in domains such as aerospace operations with semi-autonomous craft, and in narrative design for dramatic social simulations. This work would characterize the tradespace of IDEs for domain-specific languages, contributing to

human-centered programming languages research.

2. Surveying existing widgets for media creation in in IDEs. These instruments appear in creativity support tools tailored to domains such as videogame development, music production, and even science mission operations. This work would serve to clarify the goals of generative methods, and characterize practical domain-specific pipelines currently overlooked in the broader literature, contributing to games AI.

3. Teaching tools for interactive digital narrative, including QBN as a writing strategy. This might entail developing a full Dendry IDE based on the existing Borogove web editor, and exposing students to the DendryScope skein as a debugging aid. A round of design prototyping should first be conducted with students on automated playtesting for IDN, to understand their needs as non-experts.

6.3 A history of software instruments

This section returns to the question of why AI, HCI, and CSCW are different fields at all. If intelligent user interfaces are an object of study common to all three, then how have their values diverged so wildly, to the point that two research articles on the same software system can turn out mutually unintelligible?

The answer, as ever, lies in historical contingency. The rise of AI as a distinct field came in the context of Cold War-era research programs dedicated to computing technologies, which went on to disrupt practically every process in information work.

6.3.1 The Triumph of Systems Thinking

I embarked on the work of my dissertation with the conviction that *systems thinking* would be of use to all of my collaborators — surely they just lacked access to tangible models! I am not the only person who has thought along these lines, as the tech-industry blend of interface designers and software developers has given rise to many makers of **explorable explanations**, including Bret Victor and Nicky Case. Members of this informal community have all encountered the challenges of engaging in *critical technical practice*, as identified by Phillip Agre [2], which lie just beyond the implementation work.

I fortunately found opportunities to work with teams of aerospace professionals and with independent narrative designers. I sat down to write those tangible models in the exploratory notebook, and I learned to prototype rapidly in order to ask questions often. I learned that understanding the nuances of the model as the expert understood it – regardless of whether they complimented the software, – both required and enabled me to perform interviews with real depth, as demonstrated in Ch.4 and Ch.5.

In other words, the problems these experts had were not reasoning problems. They were perfectly capable of making their models tangible to themselves, after all. And I was eager to acquire their modes of systems thinking, which were not the same as mine, and indeed not the same as their colleagues'. So there was the real problem: not every collaborator came to the problem well-equipped to reason about the expert's model, through no fault of their own. If only the model were communicated better to *them*! If only they had gotten to consume the interviews compiled by my team.

6.3.2 The Tragedy of Systems Thinking

In the aerospace domain, my team pulled large amounts of domain knowledge from teams with different responsibilities, and mediated between the efforts of these teams to push the visualization scope in different directions. In the narrative domain, my team employed the *story volume* as a lens to unify the interviews with independent practitioners, who each used their own terms. In other words, to code all of these interviews, I employed both sides of *constructive grounded theory*: both gathering data from our experts as it was presented to us, and also imposing structure on that data through reflection and the literature. Neither the experts' chosen ontologies nor their practical needs can be fully understood without the other.

According to Phoebe Sengers' thesis *Anti-Boxology* [114], based on her own work designing sets of social behaviors for intelligent agents, *atomization* simultaneously allows intelligent behaviors to be defined, and yet cannot by any known algorithm be integrated into intentional behavior. In the workplace, any team of experts must always negotiate which 'boxes' really suit their (visualization) problem – that is, which model, which fact, or which practice is called upon; and therefore, which context and assumptions must be negotiated with other teams.

Star characterizes failures of such negotiations as 'anomalies' in the workplace [72]: for example, she once observed a software engineering team and their scientist collaborators caught in a cycle of failed communication. Curiously, neither party knew that their meaning was not understood, which I must imagine came about because the engineers spoke about data (as in, bytes of memory) and the scientists spoke about data (as in, worms on a slide).

Star's theory of boundary objects is precisely tailored to understanding, for example, paper tools in the workplace; and beyond these, it implies a gap filled in, I propose, by software instruments. Yet while Star's work has been taken up by managers and by designers alike, it is not widely known inside of AI. Why is that?

6.3.3 What Comes Next

My *critical technical practice* of data visualization produced software instruments; just as Phoebe Sengers' critical technical practice of cultural studies produced *Anti-Boxology*. According to Philip Agre's definition of critical technical practice [2], these methods challenge the standard methods of AI, which we may now observe as *systems thinking* in the world at large. Agre identifies four obstacles in the AI literature to the establishment of these alternative languages; historically, they have split off into formative pieces of CSCW and design theory, without seeing re-incorporation into AI.

- Given an extensive set of assumptions underlying existing practices, involving genre conventions and semi-technical metaphors, it is difficult to become aware of these premises.
- Given an alternative premises, it is difficult to produce a novel technical system or method.
- Given a novel alternative method, it is difficult to prevent the boundaries of the existing technical system from elastically growing to encompass it.
- Given a novel alternative language, it is difficult to apply this language to real problems without incorporating several interlocking methods, which may distort

the novel method back toward the traditional language.

In this work, I have identified expert interviews, workshops, and reading groups as approaches to noticing and establishing alternative premises. I have employed exploratory visualization notebooks (in particular) and intelligent user interface prototyping (in general) as novel alternative methods, falling within the toolbox of visualization design studies. And from design studies come the HCI methods of evaluation based on users' different roles and expectations, in direct contrast to endogenous strategies of evaluation like benchmarking and expressive range analysis.

I am not claiming that it is better to think about roles and communication than to think about systems and reasoning. Actually, it is impossible to walk around the *enactive loop* to view it from the other side without employing both. And without both perspectives, it is impossible to identify systems where atomization has failed, or a wicked problem has gone unaddressed — in other words, where the *boundary object* has gone missing.

Software instruments are a step toward a terminology of critical technical practices and computational media in general. We need such a terminology in order to advocate for synthesis between AI, HCI, and CSCW ways of seeing the world and addressing the needs of our expert collaborators. Let me define successful advocacy as access to the necessary institutional resources. The designer who secures experts for interviews does not have to be the software developer who creates the prototype instrument. But what if the developer needs IRB approval for a hyper-local study? What if the designer needs to deploy their lightweight prototype on a server? If these domains have collapsed into each other, then let them be known as computational media, as human-data interaction, and as visualization. These are the living traditions that enable us to bring folks into the work.

Bibliography

[1] Eytan Adar and Elsie Lee. Communicative Visualizations as a Learning Problem. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):946–956, February 2021. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[2] Philip E. Agre. Toward a Critical Technical Practice: Lessons Learned in Trying to Reform AI. In *Social Science, Technical Systems, and Cooperative Work*. Psychology Press, 1998. Num Pages: 27.

[3] Derya Akbaba, Devin Lange, Michael Correll, Alexander Lex, and Miriah Meyer. Troubling Collaboration: Matters of Care for Visualization Design Study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, pages 1–15, New York, NY, USA, April 2023. Association for Computing Machinery.

[4] Bill Albert and Tom Tullis. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Newnes, May 2013. Google-Books-ID: bPhLeMBLEkAC.

[5] Obead Alhadreti and Pam Mayhew. Rethinking Thinking Aloud: A Comparison of Three Think-Aloud Protocols. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, Montreal QC Canada, April 2018. ACM.

[6] Tim Althoff, Xin Luna Dong, Kevin Murphy, Safa Alai, Van Dang, and Wei Zhang. TimeMachine: Timeline Generation for Knowledge-Base Entities. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 19–28, Sydney NSW Australia, August 2015. ACM.

[7] Sandra Bae, Federico Rossi, Joshua Vander Hook, Scott Davidoff, and Kwan-Liu Ma. A Visual Analytics Approach to Debugging Cooperative, Autonomous Multi-Robot Systems' Worldviews. In *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 24–35, October 2020.

[8] Emma Beauxis-Aussalet, Michael Behrisch, Rita Borgo, Duen Horng Chau, Christopher Collins, David Ebert, Mennatallah El-Assady, Alex Endert, Daniel A. Keim, Jörn Kohlhammer, Daniela Oelke, Jaakko Peltonen, Maria Riveiro, Tobias Schreck, Hendrik Strobelt, and Jarke J. van Wijk. The Role of Interactive Visualization in Fostering Trust in AI. *IEEE Computer Graphics and Applications*, 41(6):7–12, November 2021. Conference Name: IEEE Computer Graphics and Applications.

[9] Jeroen Beliën, Brecht Cardoen, and Erik Demeulemeester. Improving Workforce Scheduling of Aircraft Line Maintenance at Sabena Technics. *Interfaces*, 42(4):352–364, August 2012.

[10] Mark Bernstein, David E. Millard, and Mark J. Weal. On writing sculptural Hypertext. In *Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*, HYPERTEXT '02, pages 65–66, New York, NY, USA, June 2002. Association for Computing Machinery.

[11] Jacques Bertin. *Semiology of graphics.* University of Wisconsin press, 1983.

[12] Ann Blandford. Semi-structured qualitative studies. In *The Encyclopedia of Human-Computer Interaction.* The Interaction Design Foundation, Aarhus, Denmark, 2nd edition, 2013.

[13] Margaret A. Boden. Creativity in a nutshell. *Think*, 5(15):83–96, 2007.

[14] Richard J. Boland and Ramkrishnan V. Tenkasi. Perspective Making and Perspective Taking in Communities of Knowing. *Organization Science*, 6(4):350–372, 1995. Publisher: INFORMS.

[15] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, December 2011. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[16] Adam James Bradley, Mennatallah El-Assady, Katharine Coles, Eric Alexander, Min Chen, Christopher Collins, Stefan Jänicke, and David Joseph Wrisley. Visualization and the Digital Humanities:. *IEEE Computer Graphics and Applications*, 38(6):26–38, November 2018. Conference Name: IEEE Computer Graphics and Applications.

[17] Cynthia A Brewer. Color use guidelines for mapping and visualization. *Visualization in modern cartography*, 1994(123-148):7, 1994. Publisher: Pergamon, Oxford.

[18] Alex A. T. Bui, Denise R. Aberle, and Hooshang Kangarloo. TimeLine: Visualizing Integrated Patient Records. *IEEE Transactions on Information Technology in Biomedicine*, 11(4):462–473, July 2007.

[19] Eric Butler, Adam M. Smith, Yun-En Liu, and Zoran Popovic. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 377–386, St. Andrews Scotland, United Kingdom, October 2013. ACM.

[20] Marta Caccamo, Daniel Pittino, and Fredrik Tell. Boundary objects, knowledge integration, and innovation management: A systematic review of the literature. *Technovation*, page 102645, October 2022.

[21] Rogelio E. Cardona-Rivera, José P. Zagal, and Michael S. Debus. Narrative Goals in Games: A Novel Nexus of Story and Gameplay. In *International Conference on the Foundations of Digital Games*, pages 1–4, Bugibba Malta, September 2020. ACM.

[22] Andrea Casalino, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco. Optimal Scheduling of Human–Robot Collaborative Assembly Operations With Time Petri Nets. *IEEE Transactions on Automation Science and Engineering*, 18(1):70–84, January 2021.

[23] Kathy Charmaz. Grounded theory as an emergent method. In *Handbook of emergent methods*, pages 155–170. The Guilford Press, New York, NY, US, 2008.

[24] Min Chen and Amos Golan. What May Visualization Processes Optimize? *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2619–2632, December 2016. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[25] Min Chen, Georges Grinstein, Chris R. Johnson, Jessie Kennedy, and Melanie Tory. Pathways for Theoretical Advances in Visualization. *IEEE Computer Graphics and Applications*, 37(4):103–112, 2017. Conference Name: IEEE Computer Graphics and Applications.

[26] Marta Cialdea Mayer, Andrea Orlandini, and Alessandro Umbrico. Planning and execution with flexible timelines: a formal account. *Acta Informatica*, 53(6):649–680, October 2016.

[27] William S Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984. Publisher: Taylor & Francis.

[28] Katherine E. Compton. *Casual Creators: Defining a Genre of Autotelic Creativity Support Systems*. PhD thesis, UC Santa Cruz, 2019.

[29] Matthew Conlen, Sara Stalla, Chelly Jin, Maggie Hendrie, Hillary Mushkin, Santiago Lombeyda, and Scott Davidoff. Towards Design Principles for Visual An-

alytics in Operations Contexts. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–7, Montreal QC Canada, April 2018. ACM.

[30] Michael Correll. Are We Making Progress In Visualization Research? In *2022 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV)*, pages 1–10, October 2022. arXiv:2208.11810 [cs].

[31] Robert P Crease. Paper Tools. *Physics World*, 32(12):20, December 2019.

[32] Dennis Martensson. How I Procedurally Generate Music (EXPLAINED), October 2021.

[33] Sebastian Deterding, Jonathan Hook, Rebecca Fiebrink, Marco Gillies, Jeremy Gow, Memo Akten, Gillian Smith, Antonios Liapis, and Kate Compton. Mixed-Initiative Creative Interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 628–635, Denver Colorado USA, May 2017. ACM.

[34] Alexandra Diehl, Alfie Abdul-Rahman, Benjamin Bach, Mennatallah El-Assady, Matthias Kraus, Robert S. Laramee, Daniel A. Keim, and Min Chen. Characterizing Grounded Theory Approaches in Visualization, April 2022. arXiv:2203.01777 [cs].

[35] Evanthia Dimara, Harry Zhang, Melanie Tory, and Steven Franconeri. The Unmet Data Visualization Needs of Decision Makers Within Organizations. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4101–4112, Decem-

ber 2022. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[36] Benjamin P. S. Donitz, Alan M. Didion, Austin K. Nicholas, Thaddaeus J. Voss, and Charles H. Lee. Assessing Relay Communications for Mars Sample Return Surface Mission Concepts. In *2021 IEEE Aerospace Conference (50100)*, pages 1–10, March 2021. ISSN: 1095-323X.

[37] J. Durkin. Expert systems: a view of the field. *IEEE Expert*, 11(2):56–63, April 1996. Conference Name: IEEE Expert.

[38] Upol Ehsan and Mark O. Riedl. Human-centered Explainable AI: Towards a Reflective Sociotechnical Approach, February 2020. arXiv:2002.01092 [cs].

[39] Upol Ehsan, Philipp Wintersberger, Q. Vera Liao, Elizabeth Anne Watkins, Carina Manger, Hal Daumé Iii, Andreas Riener, and Mark O Riedl. Human-Centered Explainable AI (HCXAI): Beyond Opening the Black-Box of AI. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–7, New Orleans LA USA, April 2022. ACM.

[40] Heinz Erzberger. Design principles and algorithms for automated air traffic management. AGARD, November 1995. NTRS Author Affiliations: NASA Ames Research Center NTRS Document ID: 19960012295 NTRS Research Center: Legacy CDMS (CDMS).

[41] Boris Ewenstein and Jennifer Whyte. Knowledge Practices in Design: The Role of

Visual Representations as ‘Epistemic Objects’. *Organization Studies*, 30(1):07–30, January 2009. Publisher: SAGE Publications Ltd.

[42] Failbetter Games. Fallen London, 2009. Digital game accessed 2023-09-01. Available online at: fallenlondon.com.

[43] Kenneth A. Farley, Kenneth H. Williford, Kathryn M. Stack, Rohit Bhartia, Al Chen, Manuel de la Torre, Kevin Hand, Yulia Goreva, Christopher D. K. Herd, Ricardo Hueso, Yang Liu, Justin N. Maki, German Martinez, Robert C. Moeller, Adam Nelessen, Claire E. Newman, Daniel Nunes, Adrian Ponce, Nicole Spanovich, Peter A. Willis, Luther W. Beegle, James F. Bell, Adrian J. Brown, Svein-Erik Hamran, Joel A. Hurowitz, Sylvestre Maurice, David A. Paige, Jose A. Rodriguez-Manfredi, Mitch Schulte, and Roger C. Wiens. Mars 2020 Mission Overview. *Space Science Reviews*, 216(8):142, December 2020.

[44] Jacob Garbe, Aaron A Reed, Melanie Dickinson, Noah Wardrip-Fruin, and Michael Mateas. Author Assistance Visualizations for Ice-Bound, A Combinatorial Narrative. *Foundations of Digital Games*, 2014.

[45] Alessandro G. Gardi, Kathryn de Ridder, Roberto Sabatini, and Subramanian Ramasamy. 4-Dimensional Trajectory Negotiation and Validation System for the Next Generation Air Traffic Management. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, Boston, MA, August 2013. American Institute of Aeronautics and Astronautics.

[46] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub.

Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82, January 2019. Publisher: Cambridge University Press.

[47] Carolina Gill, Scott Shim, and Liz Sanders. Prototypes as Inquiry, Visualization, and Communication. September 2011.

[48] Daniel Green, Charlie Hargood, and Fred Charles. Use of Tools: UX Principles for Interactive Narrative Authoring Tools. *Journal on Computing and Cultural Heritage*, 14(3):1–25, July 2021.

[49] Jason Grinblat. Emergent Narratives and Story Volumes. In *Procedural Generation in Game Design*. A K Peters/CRC Press, 2017. Num Pages: 9.

[50] Jason Grinblat, Cat Manning, and Max Kreminski. Emergent Narrative and Reparative Play. In Alex Mitchell and Mirjam Vosmeer, editors, *Interactive Storytelling*, volume 13138, pages 208–216. Springer International Publishing, Cham, 2021. Series Title: Lecture Notes in Computer Science.

[51] Uta Hinrichs, Stefania Forlini, and Bridget Moynihan. In defense of sandcastles: Research thinking through visualization in digital humanities. *Digital Scholarship in the Humanities*, 34(Supplement_1):i80–i99, December 2019.

[52] Judith Holton. The coding process and its challenges. *The SAGE Handbook of Grounded Theory*, pages 265–290, January 2007.

[53] Stephanie Houde and Charles Hill. What do Prototypes Prototype? In Marting G. Helander, Thomas K. Landauer, and Prasad V. Prabhu, editors, *Handbook of*

Human-Computer Interaction (Second Edition), pages 367–381. North-Holland, Amsterdam, January 1997.

[54] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct Manipulation Interfaces. *Human–Computer Interaction*, 1(4):311–338, December 1985. Publisher: Taylor & Francis _eprint: https://doi.org/10.1207/s15327051hci0104_2.

[55] Joel Jakubovic, Jonathan Edwards, and Tomas Petricek. Technical Dimensions of Programming Systems. *The Art, Science, and Engineering of Programming*, 7(3):13:1–13:59, February 2023.

[56] Yvonne Jansen, Pierre Dragicevic, Petra Isenberg, Jason Alexander, Abhijit Karnik, Johan Kildal, Sriram Subramanian, and Kasper Hornbæk. Opportunities and Challenges for Data Physicalization. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3227–3236, Seoul Republic of Korea, April 2015. ACM.

[57] H. Z. Jia, J. Y. H. Fuh, A. Y. C. Nee, and Y. F. Zhang. Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems. *Computers & Industrial Engineering*, 53(2):313–320, September 2007.

[58] Jaemin Jo, Jaeseok Huh, Jonghun Park, Bohyoung Kim, and Jinwook Seo. Live-Gantt: Interactively Visualizing a Large Manufacturing Schedule. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2329–2338, December 2014.

[59] Joey Donald Jones. Authorial Burden. In Charlie Hargood, David E. Millard, Alex Mitchell, and Ulrike Spierling, editors, *The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives*, Human–Computer Interaction Series, pages 47–63. Springer International Publishing, Cham, 2022.

[60] Nick Junius, Michael Mateas, and Noah Wardrip-Fruin. Towards expressive input for character dialogue in digital games. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, pages 1–11, San Luis Obispo California USA, August 2019. ACM.

[61] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, October 2011. Publisher: SAGE Publications.

[62] Isaac Karth, Nic Junius, and Max Kreminski. Constructing a Catbox: Story Volume Poetics in Umineko no Naku Koro ni. In Mirjam Vosmeer and Lissa Holloway-Attaway, editors, *Interactive Storytelling*, volume 13762, pages 455–470. Springer International Publishing, Cham, 2022. Series Title: Lecture Notes in Computer Science.

[63] D.A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, January 2002. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[64] Juho Kim, Haoqi Zhang, Paul André, Lydia B. Chilton, Wendy Mackay, Michel

Beaudouin-Lafon, Robert C. Miller, and Steven P. Dow. Cobi: a community-informed conference scheduling tool. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, UIST '13, pages 173–182, New York, NY, USA, October 2013. Association for Computing Machinery.

[65] Ursula Klein. Paper tools in experimental cultures. *Studies in History and Philosophy of Science Part A*, 32(2):265–302, June 2001.

[66] Robert Kosara. Notebooks for Data Analysis and Visualization: Moving Beyond the Data. *IEEE Computer Graphics and Applications*, 43(1):91–96, January 2023.

[67] Max Kreminski and Noah Wardrip-Fruin. Sketching a Map of the Storylets Design Space. In Rebecca Rouse, Hartmut Koenitz, and Mads Haahr, editors, *Interactive Storytelling*, Lecture Notes in Computer Science, pages 160–164, Cham, 2018. Springer International Publishing.

[68] Sudhir Kumar, Glen Stecher, Michael Suleski, and S. Blair Hedges. TimeTree: A Resource for Timelines, Timetrees, and Divergence Times. *Molecular Biology and Evolution*, 34(7):1812–1819, July 2017.

[69] J. Layland and L. L. Rauch. The Evolution of Technology in the Deep Space Network: A History of the Advanced Systems Program. November 1994.

[70] Yoon Ha Lee. Winterstrike, 2012. Digital game accessed 2023-09-01. Available online at: winterstrike.storynexus.com.

[71] T. L. Lei and R. L. Church. Mapping transit-based access: integrating GIS,

routes and schedules. *International Journal of Geographical Information Science*, 24(2):283–304, February 2010.

[72] Susan Leigh Star. This is Not a Boundary Object: Reflections on the Origin of a Concept. *Science, Technology, & Human Values*, 35(5):601–617, September 2010. Publisher: SAGE Publications Inc.

[73] Jingyi Li, Eric Rawn, Jacob Ritchie, Jasper Tran O'Leary, and Sean Follmer. Beyond the Artifact: Power as a Lens for Creativity Support Tools. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–15, San Francisco CA USA, October 2023. ACM.

[74] Jock Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141, 1986. Publisher: Acm New York, NY, USA.

[75] Gloria Mark, Kalle Lyytinen, and Mark Bergman. Boundary Objects in Design: An Ecological View of Design Artifacts. *Journal of the Association for Information Systems*, 8(11):546–568, November 2007.

[76] Stacey Mason, Ceri Stagg, and Noah Wardrip-Fruin. Lume: a system for procedural story generation. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, pages 1–9, San Luis Obispo California USA, August 2019. ACM.

[77] Michael Mateas and Andrew Stern. Structuring Content in the Façade Inter-

active Drama Architecture. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 1(1):93–98, 2005. Number: 1.

[78] Peter Mawhorter, Michael Mateas, Noah Wardrip-Fruin, and Arnav Jhala. Towards a Theory of Choice Poetics. *Foundations of Digital Games*, 2014.

[79] Alex McLean. Making programming languages to dance to: live coding with tidal. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, pages 63–70, Gothenburg Sweden, September 2014. ACM.

[80] Andrew McNutt, Anton Outkine, and Ravi Chugh. A Study of Editor Features in a Creative Coding Classroom, January 2023. arXiv:2301.13302 [cs].

[81] Miriah Meyer and Jason Dykes. Reflection on Reflection in Applied Visualization Research. *IEEE Computer Graphics and Applications*, 38(6):9–16, November 2018. Conference Name: IEEE Computer Graphics and Applications.

[82] Miriah Meyer and Jason Dykes. Criteria for Rigor in Visualization Design Study. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2019. arXiv: 1907.08495.

[83] Miriah Meyer, Michael Sedlmair, and Tamara Munzner. The four-level nested model revisited: blocks and guidelines. In *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors-Novel Evaluation Methods for Visualization*, pages 1–6, 2012.

[84] David E. Millard and Charlie Hargood. Hypertext as a Lens into Interactive

Digital Narrative. In Alex Mitchell and Mirjam Vosmeer, editors, *Interactive Storytelling*, volume 13138, pages 509–524. Springer International Publishing, Cham, 2021. Series Title: Lecture Notes in Computer Science.

[85] Ian Millington and Autumn Chen. Dendry, 2015. Software accessed 2023-05-25. Available online at: github.com/aucchen/dendry.

[86] Alex Mitchell. Writing for Replay: Supporting the Authoring of Kaleidoscopic Interactive Narratives. In Charlie Hargood, David E. Millard, Alex Mitchell, and Ulrike Spierling, editors, *The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives*, Human–Computer Interaction Series, pages 131–145. Springer International Publishing, Cham, 2022.

[87] Dominik Moritz. Hello Clingo, November 2022. Software accessed 2023-05-26. Available online at: observablehq.com/@cmudig/clingo.

[88] Brian K. Muirhead, Austin Nicholas, and Jeff Umland. Mars Sample Return Mission Concept Status. In *2020 IEEE Aerospace Conference*, pages 1–8, March 2020. ISSN: 1095-323X.

[89] Tamara Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, November 2009.

[90] Graham Nelson. Natural language, semantic analysis, and interactive fiction. *IF Theory Reader*, 141(99):104, 2006.

[91] Elisabeth Oliver and Adam M. Smith. Twinealyzer: Static Analysis for Twine Games, 2018. Software accessed 2023-05-16. Available online at: twinealyzer.org.

[92] Erica L. Olmsted-Hawala, Elizabeth D. Murphy, Sam Hawala, and Kathleen T. Ashenfelter. Think-aloud protocols: a comparison of three think-aloud protocols for use in testing data-dissemination web sites for usability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2381–2390, Atlanta Georgia USA, April 2010. ACM.

[93] Michael Oppermann and Tamara Munzner. Data-First Visualization Design Studies. *arXiv:2009.01785 [cs]*, September 2020. arXiv: 2009.01785.

[94] Catherine M. Otto. *Textbook of Clinical Echocardiography.* Elsevier/Saunders, 2018. Google-Books-ID: C3UgtAEACAAJ.

[95] Jasmine Otto, Autumn Chen, and Adam M. Smith. DendryScope: Narrative Designer Support via Symbolic Analysis. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 19(1):315–325, October 2023. Number: 1.

[96] Lace M. Padilla, Sarah H. Creem-Regehr, Mary Hegarty, and Jeanine K. Stefanucci. Decision making with visualizations: a cognitive framework across disciplines. *Cognitive Research: Principles and Implications*, 3(1):29, July 2018.

[97] People Make Games. The System Behind Hades' Astounding Dialogue, November 2020. Video essay accessed 2023-05-26. Available online at: youtube.com/watch?v=bwdYL0KFA_U.

[98] Andrew Plotkin. PlotEx: a tool for exploring puzzle plot constraints, 2012. Software accessed 2023-05-26. Available online at: eblong.com/zarf/plotex/.

[99] Helen C Purchase, Natalia Andrienko, Thomas J Jankun-Kelly, and Matthew Ward. Theoretical foundations of information visualization. In *Information visualization: Human-centered issues and perspectives*, pages 46–64. Springer, 2008.

[100] Aaron Reed. *Changeful Tales: Design-Driven Approaches Toward More Expressive Storygames*. PhD thesis, UC Santa Cruz, 2017.

[101] David Ribes. STS, Meet Data Science, Once Again. *Science, Technology, & Human Values*, 44(3):514–539, May 2019.

[102] David Ribes, Andrew S Hoffman, Steven C Slota, and Geoffrey C Bowker. The logic of domains. *Social Studies of Science*, 49(3):281–309, June 2019.

[103] Alexander Rind, Tim Lammarsch, Wolfgang Aigner, Bilal Alsallakh, and Silvia Miksch. TimeBench: A Data Model and Software Library for Visual Analytics of Time-Oriented Data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2247–2256, December 2013. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[104] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, June 1973.

[105] Jen Rogers, Austin H. Patton, Luke Harmon, Alexander Lex, and Miriah Meyer. Insights From Experiments With Rigor in an EvoBio Design Study. *IEEE*

Transactions on Visualization and Computer Graphics, 27(2):1106–1116, February 2021. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[106] Avi Rosenfeld and Ariella Richardson. Explainability in human–agent systems. *Autonomous Agents and Multi-Agent Systems*, 33(6):673–705, November 2019.

[107] James Ryan. *Curating Simulated Storyworlds.* PhD thesis, December 2018.

[108] P. C. Sabeti, S. F. Schaffner, B. Fry, J. Lohmueller, P. Varilly, O. Shamovsky, A. Palma, T. S. Mikkelsen, D. Altshuler, and E. S. Lander. Positive Natural Selection in the Human Lineage. *Science*, 312(5780):1614–1620, June 2006.

[109] Michael Saint-Guillain, Jean Vanderdonckt, Nicolas Burny, Vladimir Pletser, Tiago Vaquero, Steve Chien, Alexander Karl, Audrey Comein, Cheyenne Chamart, Cyril Wain, Ignacio S Casla, Jean Jacobs, Julie Manon, Julien Meert, and Sirga Drouet. ENABLING ASTRONAUT SELF-SCHEDULING USING A ROBUST MODELLING AND SCHEDULING SYSTEM (RAMS): A MARS ANALOG USE CASE. *16th Symposium on Advanced Space Technologies in Robotics and Automation*, page 8, June 2022.

[110] James Schier, John Rush, Pete Vrotsos, and Wallace Williams. Space Communication Architecture Supporting Exploration and Science: Plans & Studies for 2010-2030. In *1st Space Exploration Conference: Continuing the Voyage of Discovery.* American Institute of Aeronautics and Astronautics, 2005. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2005-2517.

[111] Donald A. Schon and Vincent DeSanctis. Design as a Reflective Conversation with the Situation. In *The reflective practitioner : how professionals think in action*, volume 34, pages 76–104. July 1986.

[112] Michael Sedlmair. Design Study Contributions Come in Different Guises: Seven Guiding Scenarios. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, BELIV '16, pages 152–161, New York, NY, USA, October 2016. Association for Computing Machinery.

[113] Michael Sedlmair, Miriah Meyer, and Tamara Munzner. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, December 2012.

[114] Phoebe Sengers. *Anti-Boxology: Agent Design in Cultural Context.* PhD thesis, Carnegie Mellon University, August 1998.

[115] Yotam Shibolet, Noam Knoller, and Hartmut Koenitz. A Framework for Classifying and Describing Authoring Tools for Interactive Digital Narrative. In Rebecca Rouse, Hartmut Koenitz, and Mads Haahr, editors, *Interactive Storytelling*, Lecture Notes in Computer Science, pages 523–533, Cham, 2018. Springer International Publishing.

[116] Ben Shneiderman, Gerhard Fischer, Mary Czerwinski, Mitch Resnick, Brad Myers, Linda Candy, Ernest Edmonds, Mike Eisenberg, Elisa Giaccardi, Tom Hewett, Pamela Jennings, Bill Kules, Kumiyo Nakakoji, Jay Nunamaker, Randy Pausch, Ted Selker, Elisabeth Sylvan, and Michael Terry. Creativity Support Tools: Re-

port From a U.S. National Science Foundation Sponsored Workshop. *International Journal of Human-Computer Interaction*, 20(2):61–77, May 2006.

[117] Emily Short. Bee, 2012. Digital game accessed 2023-05-25. Available online at: ifdb.org/viewgame?id=8pe83e92v4nvabic.

[118] Andrejs Skaburskis. The Origin of “Wicked Problems”. *Planning Theory & Practice*, 9(2):277–280, June 2008. Publisher: Routledge _eprint: https://doi.org/10.1080/14649350802041654.

[119] Adam M. Smith and Michael Mateas. Answer Set Programming for Procedural Content Generation: A Design Space Approach. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):187–200, September 2011. Conference Name: IEEE Transactions on Computational Intelligence and AI in Games.

[120] Adam M. Smith, Mark J. Nelson, and Michael Mateas. LUDOCORE: A logical game engine for modeling videogames. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 91–98, Copenhagen, Denmark, August 2010. IEEE.

[121] Gillian Smith and Jim Whitehead. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–7, Monterey California, June 2010. ACM.

[122] Robert F. Sproull. Raster graphics for interactive programming environments. In *Proceedings of the 6th annual conference on Computer graphics and interactive*

techniques, SIGGRAPH '79, pages 83–93, New York, NY, USA, August 1979. Association for Computing Machinery.

[123] Susan Leigh Star. The structure of ill-structured solutions: boundary objects and heterogeneous distributed problem solving. In *Distributed Artificial Intelligence (Vol. 2)*, pages 37–54. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, December 1989.

[124] Susan Leigh Star and James R Griesemer. Institutional ecology,translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social studies of science*, 19(3):387–420, 1989. Publisher: Sage Publications London.

[125] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32–47, January 1987.

[126] Thinley Tharchen, Raghu Garud, and Rebecca L. Henn. Design as an interactive boundary object. *Journal of Organization Design*, 9(1):21, October 2020.

[127] Melanie Tory, Sheryl Staub-French, Dandan Huang, Yu-Ling Chang, Colin Swindells, and Rachel Pottinger. Comparative visualization of construction schedules. *Automation in Construction*, 29:68–82, January 2013.

[128] Carolina Veloso and Rui Prada. Validating the plot of Interactive Narrative games.

In *2021 IEEE Conference on Games (CoG)*, pages 01–08, August 2021. ISSN: 2325-4289.

[129] Bret Victor. Up and Down the Ladder of Abstraction: A Systematic Approach to Interactive Visualization, 2011.

[130] R. Vuillemot, P. Rivière, A. Beignon, and A. Tabard. Boundary Objects in Design Studies: Reflections on the Collaborative Creation of Isochrone Maps. *Computer Graphics Forum*, 40(3):349–360, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14312.

[131] James M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430–437, September 2003.

[132] Jessica Wynne. *Do Not Erase: Mathematicians and Their Chalkboards*. Princeton University Press, June 2021. Google-Books-ID: NR4OEAAAQBAJ.

[133] Caroline Ziemkiewicz, Peter Kinnaird, Robert Kosara, Jock Mackinlay, Bernice Rogowitz, and Ji Soo Yi. Visualization theory: Putting the pieces together. *IEEE VisWeek Panel*, 2010.

[134] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 493–502, San Jose California USA, April 2007. ACM.