2014

Defining a software genre: timeline navigators

Eduardo Rubio
Iowa State University

Defining a software genre: Timeline navigators

by

Eduardo Rubio

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Human Computer Interaction

Program of Study Committee:

Stephen Gilbert, Major Professor
Paul Bruski
Debra Satterfield

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

ABSTRACT

This research defines a widely used, but largely unrecognized genre of software. From a User Experience Design standpoint, this genre definition considers any application that helps users capture, understand, and manipulate time-series data.

Computer user interfaces investigated within this genre help users manage time-series data. Digital interactions with time-series data range from simple presses of the play/pause button, loops to find the right information, to complex analyses necessary for greater understanding. A child may watch a movie played on a media player like Apple's iTunes. A student may compose a musical presentation with Steinberg's Cubase. Trying to determine market trends, investors may conduct a stock analysis on Charles Schwab's OptionsXpress.

This thesis investigates applications and discovers properties shared by each application. By looking at how the interfaces in this genre are composed, this thesis establishes common practices and conventions enabling other designers to create successful user experiences.

CHAPTER I

INTRODUCTION: THE TIMELINE NAVIGATOR

Time-Series Data

Wide audiences consume data indexed on time. Digital interactions with time-series data range from simple presses of the play/pause button, loops to find the right information, and then finally very complex analyses necessary for greater understanding. A child may watch a movie played on a media player like Apple's iTunes. A student may compose a musical presentation with Steinberg's Cubase. Trying to determine market trends, statisticians may conduct a time-series data analysis on Charles Schwab's OptionsXpress. These applications provide a similar service to different domains because they share certain attributes. This thesis observes and defines that which holds these applications in common: interaction with time-series data. This research investigates how interfaces help users capture, manipulate, and understand time-series data. With an established user experience design framework, this research then prescribes interaction design patterns and practices intent on providing these users with an optimal user experience.

Because the research presented in this thesis focuses not just on the consumption of time-series data, but also the *interaction* with time-series data, this thesis focuses on a significantly more dynamic user experience. Helping users reach an understanding of their time-series data by managing it typically demands complex user

interactions. Complex user interactions mean they often come with high learning curves. Rarely are these applied user experience designs simple.

To fully comprehend how this specific genre of software meets user needs, this research takes a closer look at what may be dubbed *timeline navigators*, which are defined here as those user interface (UI) designs which enable users to interact with time-series data. Applications from audio-video processing, signal processing, medical physiology, kinesiology and sports, history, and knowledge management guide the work presented here. Users working in each of these domains may be far in terms of subject matter, but their common interaction with data indexed on time pays testament to this software genre's prevalence and therefore its prime candidacy for investigation. This research focuses entirely on the problem space shared between these domains.

One might frame this research by considering what Richard Saul Wurman (1989) states in his pivotal book, *Information Anxiety.* He coined the term *information anxiety* as "the black hole between data and knowledge." In the book he outlines the importance of organizing information by arguing how most information really does not *inform*. Humans, when presented with disorganized content, may not effectively process it past a state of confusion. Because the "information" cannot help the human consumer reach understanding, these disorganized cases are better served by a term like "data" which reaches the senses and no more. Since users working with time-series data often make *understanding* a top priority, this research considers Wurman's guidelines fundamental.

Wurman stresses the importance of organizing information because anxiety introduced by "the black hole" may leave users with the following conditions:

1. Not understanding the information
2. Feeling overwhelmed by the amount of information to be understood
3. Not knowing if certain information exists
4. Not knowing where to find information
5. Knowing where to find the information, but not having the key to access it

Fortunately, Wurman forms a solution fit for the *Information Architect's (IA)* toolbox with a convenient acronym. LATCH stands for the five ways data can be organized for efficient mental processing: L) location, A) alphabetical, T) time, C) categorical, and H) hierarchy. Unfortunately, users are not always equipped with the tools needed to manage what Wurman would naturally put under the letter "T". That deficit motivates this research on *timeline navigators*. Take the following researchers and trainers in the U.S. Army as a primary example.

The After Action Review

In military combat, soldier responses often mean the difference between life and death. Untimely decisions may increase battlefield complications. Mismanaged firefights may increase combat exposure. Inappropriate commands may endanger human lives. With these stakes applied, soldiers often operate in what experts refer to as the "the fog of war," a complex working environment no soldier can navigate without specialized

training. Given these circumstances, ineffective soldier training risks valuable resources. The U.S. Army, specifically, suffers time, money, and at worst, lives. Significantly interested in all three, the Army Research Institute for the Behavioral and Social Sciences (ARI) ensures soldiers perform properly by carefully reviewing time-series data captured in specialized training methods dating back to the 1970s.

U.S. Army trainers and researchers prepare their soldiers with simulated collective training. Trainers assign a soldier unit, or echelon, to specific mission objectives and tasks. Upon receiving the training's briefing, the unit is immersed in carefully designed environments. The session simulates the varying factors trainees may face in real live combat. These training environments may be simulated using only real-word settings (Bosley, Onoszko, & Sevilla, 1979), but with recently increased technological advances, many sessions are now being facilitated using virtual simulations as well (Alluisi, 1991; Peterson et al., 2012; Sikora & Coose, 1995).

Approaching potential methodologies, the U.S. Army considers observational techniques a natural training tool and they *always* capture the training events using some recording instrument (Morrison & Meliza, 1999). By recording a training session's events ethnographically, trainers and researchers mine very rich data ripe with context, a method very suitable for soldier performance improvement.

At the simulation's conclusion, a discussion leader (U.S. Army researcher not participating in the simulation) facilitates an active discussion with the participating unit. This discussion is called an After Action Review (AAR). Morrison & Meliza (1999) state

that in any given AAR, the participating unit, guided by the discussion leader, reviews data gathered from the simulation in hopes of answering three key questions:

1. What happened during the collective training exercise?
2. Why did it happen?
3. How can units improve their performance?

Used to train the U.S. Army's population of soldiers, AARs cover broad training scenarios. Multiple AAR training programs, accompanied by extensive guidebooks, have been adapted to fit unit roles across the military branch (Campbell et al., 1999; Deatz et al., 1998; Koger et al., 1996). An AAR conducted in the western region of the country may help train special operations units while a southern facility may concentrate on training transportation units; each and every AAR returns a different analysis. Group dynamics, the unit's echelon, the assigned mission objectives, executed tasks, and countless other variables affect an AAR's conclusion.

The fundamentally varying nature of these training exercises and diverse unit roles have challenged the ARI to continuously evolve the AAR process with effectively applied methodologies (Bosley, Onoszko, Knerr, & Sulzen, 1979; Shlechter, Bessemer, Nesselroade, & Anthony, 1995; Word, 1987). Throughout its history, AAR methods needed to remain flexible enough to accommodate the fluid circumstances often occurring within the simulation, but rigid enough to avoid the "laissez-faire" model found threatening task improvements in earlier training methods (Gubler, 1997). The ARI needed a structured approach to conducting AARs.

The ARI responded accordingly with the development of specialized AAR methods and standards. Beginning with Squad Combat Operations Exercises (SCOPES) in 1973, the ARI developed more than ten individual Simulation-Based Training Systems (training programs) all designed to work with the AAR (Morrison & Meliza, 1999). These programs addressed various combinations of training environments (live, virtual, constructive), echelon, and structured vs. non-structured reviews (Word, 1987). Campbell, Deter, and Quinkert (1997), solidified the structured AAR by identifying four common features:

1. A focus on the performance of selected critical tasks.
2. Standardized exercise control to ensure practice of the tasks.
3. Standardized feedback to correct and reinforce performance on the selected tasks.
4. Exercise support by means of comprehensive training material.

The method's continuing support from the military organization indicates an achieved effectiveness in delivering desired results. The most recently published research considers AARs the "single most important event in collective training" (U.S. Army Training and Doctrine Command, 1997), and "one of the most important training interventions ever" (Sullivan, 1995). ARI research even encourages commercial organizations to adopt similar methods (Davenport & Prusak, 1997; Sullivan & Harper, 1997).

Much of this strong support comes from the AAR's basic premise: the importance of immediate accurate feedback. ARI experts argue events, which occurred

during the training session, may not be obvious to the participating unit (Morrison & Meliza, 1999). Moving forward with a unit's improving performance is significantly enhanced with immediate, but also *detailed* feedback (Allen & Smith, 1994; Johnson & Gonzalez, 2008). More details establish an understanding between cause and effect thus enhancing learning and performance.

Everything outlined in this training method's background, its history, development, and current implementation practices heavily depend on what ARI researchers call AAR aids (Morrison & Meliza, 1999). These are the various data gathering devices, note-taking instruments, and review/editing hardware/software used in facilitating the entire AAR. Since the range of required AAR aids all depend on which rich and important training program is picked, meeting user demands can be quite a challenge (Meliza & Tan, 1996) . Recording soldier GPS locations, weapon projectile trajectories, displaying video feeds, visualizing data, and listening to voice commands are just some examples of what may be required.

To clearly understand how an AAR's data, naturally indexed on time, can be understood, let the following example highlight some of the challenges and solutions:

> *Prior to beginning a typical simulation, the discussion leader sets up video cameras on soldier helmets and in the simulation environment, equips them individually with microphones, physically attaches biometric monitors to their bodies, tracks their GPS location, and gives them a specially designed plastic rifle capable of firing virtual bullets. All this is done in hopes of capturing the most relevant data possible.*

In this specific training exercise, military researchers wish to review a platoon's collective performance with new recruits in attendance. The unit, led by a recently promoted officer, will carry out the assigned mission objective of rescuing two hostages from a nearby building. The unit will enter the simulated environment in a physical, militarized vehicle and immediately face hostile fire from real live training confederates using similar virtual projectiles. The unit will navigate a Live, Virtual, Constructive (LVC) environment composed of real physical props like barriers, walls, plastic rifles, etc. and virtual simulations displayed on large and small television screens.

At the simulation's end, researchers are tasked with parsing and analyzing, at most, two hours of data (Morrison & Meliza, 1999). Depending on the training session's parameters, AAR participants and leaders could potentially search and discuss dozens of significant events at any given point in the observation's timeline. Based on the study's planned metrics, they may need to count the frequency of shots fired, "injuries" sustained, how orders were articulated, how orders were followed, how objectives were cleared, etc. Given the proposed scenario, one could reasonably expect large, perhaps overwhelming, quantities of data from a day's observations.

By contemplating these researchers and their daunting task, sympathy may grow in considering one crippling caveat: each and every

data source is out of syncopation. The video feeds, audio feeds, biometric data, GPS, electric notes, paper notes, all captured with separate proprietary devices, come with unique timestamps. These researchers, as current practices go, would be working with multiple applications, identifying splintered events that occurred in real-time (Laughlin, Peterson, & Camou, 2011; Laughlin, 2011; Meliza & Brown, 1996). For example, military trainers may consider Virtual Battlespace (VBS2), a leading product designed and developed to meet most AAR needs, as a facilitating AAR aid, but users cannot easily supplement their analysis with data capturing devices not supported by the system like extra cameras, microphones, or biometric data (Bohemia Interactive Australia Pty Ltd, 2012; Peterson et al., 2012). One can imagine the task interruption introduced by this decentralized approach should trainers need to supplement their analysis.

This constraint requires important annotations in the audio feed be manually synced with video feeds. Likewise, researchers reviewing biometric data and GPS data will have the extra task of figuring out which heart rate belongs to which soldier's location at any given time. User interactions, important to reaching understanding, break down due to this constraint. The user experience, by many standards, fails.

OmniScribe

Researchers at Iowa State University's Virtual Reality Applications Center (VRAC) recognized the potential benefit in providing a system capable of helping users avoid the described constraint. They designed and developed a program called OmniScribe, an open source software application focused on capturing, managing, and manipulating data gathered in an AAR (Peterson et al., 2012). In support of the application's user-centered design, the VRAC researchers surveyed individuals working within military, and other domains like interpersonal communication (Raij & Lok, 2008), behavioral research, and medical system training (Quarles, Lampotang, Fischer, Fishwick, & Lok, 2008).

Feedback gained in the research helped develop two main reports. One summarizes the survey's user responses (Laughlin, 2011). The second outlines the application's functional requirements (Laughlin et al., 2011). The summarization supports OmniScribe's primary intentions in respect to what users want. This report demonstrates how a specific user segment demands the application's design and development. It establishes *user need.* The extensive functional requirements document, as a working document, serves two main purposes: 1) the document specifies the application's software architecture and 2) the document outlines potential use case scenarios (Laughlin et al., 2011). In its first purpose, the document's information helps engineers communicate how the application performs its various functions. In its second purpose, the document specifies use cases, scenarios in which the application may facilitate user needs.

These documents are important to this research because they provide invaluable user data capable of informing this research's most basic purpose: helping users interact with time-series data. Based on information present in the first report, VRAC researchers establish the application's five functional requirements:

1. Record and play-back common data sources
2. Time-synchronize all data sources
3. Annotation of data
4. Add unique and new data sources
5. Offer an API for real-time data-mining and analysis

Among other things, OmniScribe, by centralizing disparate data sources, concentrates the user's main actions to one software application (Peterson et al., 2012). Applications like OmniScribe take center stage in this research. It provides users with an effective way of managing and analyzing time-series data so that they may achieve whatever goals fit their contexts.

Thesis Organization

In the sections to follow, this work will include *timeline navigators* used outside the military context. Chapter 2 outlines the philosophy and thought process behind the software genre's definition. Chapter 3 outlines the methodology used to select and analyze interfaces within the *timeline navigator* genre. Chapter 4 provides the findings of that analysis with the help of screenshots and a pattern language. Chapter 5 discusses the importance, limitations, and future of software genre definitions.

CHAPTER II

LITERATURE REVIEW: USER EXPERIENCE DESIGN

Users like those interacting with time-series data need high quality software, but a number of obstacles lie between present demands and a solution's implementation. A product's role in serving user contexts cannot be accurately defined without first learning their mental models (A. Cooper, Reimann, & Cronin, 2007). Questions about the users remain open:

- How do they accomplish current goals?
- Which are the most significant pain points with their current methods?

Notice how these questions target data fundamentally possessed by the user. These data cannot be gathered elsewhere. Gathering these data helps product teams know their user (Portigal, 2013). By gaining an empathetic perspective, product teams can take further steps to provide a better solution (Patnaik, 2009). Once user data brings new insight, consider the following:

- What *could* serve the needs found?
- What *will* serve the needs found?

The first question explores a range of possibilities and constraints. The second question demands a precise solution. Defining these two answers in combination warrant appropriate methodologies centered on the user. Any number of concepts might satisfy certain needs, but few will fit into one coherent and feasible solution.

By understanding what users need, a robust and tangible solution can form so long as one's process champions the user. Specific methods are selected as projects unfold, but the process strives for one principle: help users achieve their goals. This process is known today as *User-Centered Design.* Before describing how this process and its long list of potential methods will provide a basis for the work in this thesis, a brief background on the process will spotlight its importance.

User Centered Design

Characteristically negative consequences occur when product teams forget their users. A user worked into steps he does not perceive, know, expect, or like, does not perform well. Fortunately, professions currently dedicate their talent to finding and/or preventing these missteps (Crosby, 2000).

Before these professions existed, experiences with software products were not always user-friendly. Seasoned software professionals talk of how a product's design happened by accident, "if at all." (*Keynote Alan Cooper at TNW2012*, 2012). When contextualized interactions between users and their product received little to no recognition, product teams implemented on their own accord. These self-referential design methods lead to failed interactions between man and machine.

In recognizing the same failures, Donald Norman coined the term *User-Centered Design* (UCD) to describe a design process centered on the user (Norman & Draper, 1986). Due to its powerful impact in serving user needs, many industrial entities adopted the process to guide their product teams. Today, product teams recognize why

designs centered on the user succeed. Success stories, built with the UCD process, opened a new professional field.

Often credited as the father of *User Experience* (UX), Donald Norman also paired the pioneering UCD term with an accompanying philosophy. In adopting a UX philosophy, this thesis applies the philosophy to define the *timeline navigator* genre. As philosophies go, their elaboration may require an epic effort, but this thesis will assume one constant: all UX philosophies differ. Herein lies a significant challenge accepted by this thesis: adopting the appropriate UX philosophy.

User Experience Philosophies

The majority of UX practitioners come from different backgrounds. As champions of the user, each practicing discipline finds ways of helping users interact with their products effectively, efficiently, and satisfactorily. Psychologists, engineers, designers, sociologists, statisticians, architects, and more, often contribute their fields' knowledge to good user experiences. However, due to the UX field's interdisciplinary nature, one may find the varying interpretations bewildering. Place the term "user-centered design" or "user experience" in a Google image search and dozens of diagrams depicting different concepts will return. It seems many professionals are willing to offer thoughtful inspirations, but few will serve this thesis well. Understanding and selecting one of these philosophies can be quite a challenge since they all serve different purposes, but certainly not all the concepts depicted appropriately serve a thesis

defining a software genre. One can initiate a selection by looking at the various UCD methods available today.

Some UCD methods used in the UX field establish a clear purpose. For example, *Lean UX* works great with startups because it eliminates many of the time-consuming deliverables (Gothelf & Seiden, 2013). The lack of deliverables, however, may not work for education or this thesis since educators, and the academic setting in general, needs tangible ways of communicating design. A technique like the *online survey* can only gather superficial information, but it can also give a project a quick start. *Contextual inquiries* gather a lot of rich data, but they take skill and practice to conduct effectively. *Heuristic evaluations* may help highlight potential problems, but only a *usability test* can find unpredictable problems with real user data (Mao, Vredenburg, Smith, & Carey, 2005). Businesses proposing concrete solutions may consider techniques like *prototyping* before spending valuable resources. Finding the right UCD method depends on the project's purpose (Albert & Tullis, 2013). The work presented here is no different. Methods compatible to this work's purpose will need consideration and review.

Qualifying the UX Philosophy:

- To identify conventions in using a *timeline navigator*, answering the two very first questions presented above require the evaluation of the experience within the genre. This research's method and its execution are described in Chapter 3.
- To clearly define the software genre, expert knowledge on *design principles* and *patterns* draw similarities between the experiences discovered in the

applications analyzed. An example-based analysis in Chapter 4 forms the basis for a discussion defining the genre in the final Chapter 5.

- To visually communicate the ideas expressed in this thesis, a depiction maps them to a conceptual framework appropriate for this thesis's setting. It is introduced below.

The Elements of User Experience

The Elements of User Experience, visualized in figure 2.01, grounds abstract concepts into conversational pieces made specifically for the user-centered designer (Morrow, 2013). It "pieces apart" the user experience (*UX Week 2009 | Jesse James Garrett | The State Of User Experience*, 2009). It gives professionals from differing idioms a language they can use for a common purpose.

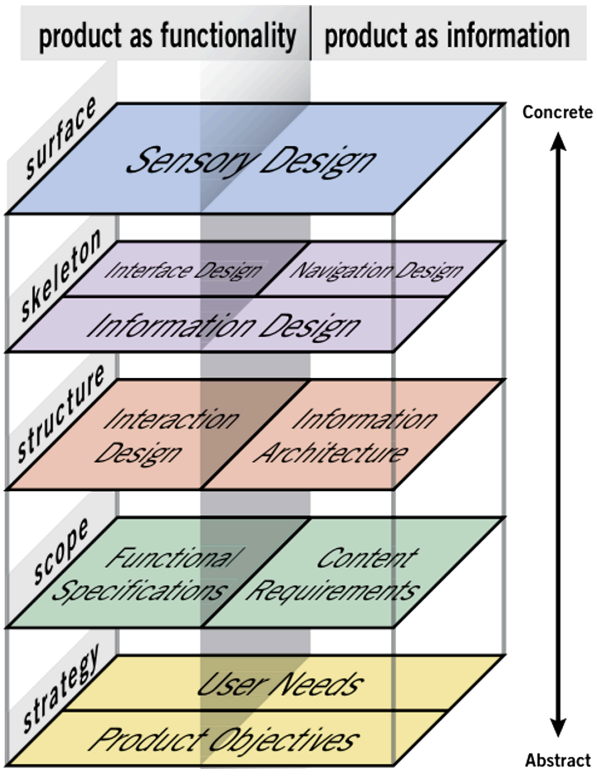

Figure 2.01 *– The Elements of User Experience* (Garrett, 2010)

In 2010, Jessie James Garrett released the 2nd edition of his book, *The Elements of the User Experience* (Garrett, 2010). In the book he provides a detailed description of a

conceptual framework (prominent and successful within the UX community) for UX design. While there are a number of similar frameworks, such as Cooper's *Stage-Gate Model* (2008), Urban, Hauser, and Dholakia's, *Essentials of Product Management* (1987), and Ulrich and Eppinger's *Product Design and Development* (1995), this thesis uses Garrett's because it focuses on user experience.

A hypothetical product team trying to produce a *timeline navigator* for music producers is used to show the elements at work.

The Strategy Plane

Because they are wise and experienced, the product team members have at least one person cover the *Strategy* plane, Mrs. Strategy. She needs to answer two types of questions that inquire about the *business objectives* and *user needs*:

- What will keep the product in business?
- What do users need?

Garrett splits the plane with these two questions. The side covering user needs might also be served by the two questions asked earlier:

- What do users need?
    - How do they accomplish current goals?
    - Which are their most significant pain points?

The strategy plane's wide problem landscape often means Mrs. Strategy is hardily ever just one person. Depending on the project's scale, entire marketing departments may help inform the experience's *Strategy.* User researchers often cover

half of this plane by gathering and modeling user data into different *personas*, archetypes of users defined by detecting behavior patterns and goals (A. Cooper et al., 2007a). Research conducted on this plane often informs decisions made along the four following planes.

If the product team designs a *timeline navigator* because music producers need to track and schedule artists by date, the experience will be failed by the most fundamental and basic element. The music producers may be working with time-series data, but they do not need a *timeline navigator.* They need a calendar.

The Scope Plane

Garrett splits this particular plane for one main reason. Users need content, but they also need to do something with that content. According to Jake Archibald, some digital products help users "get stuff," others help users "do stuff" (Hay, 2013). Many products like *timeline navigators* help with both. Distinguishing the two areas of focus is important because it will help the product team design with two types of very important questions in mind:

- What content do users need access to?
- How do they need to interact with this content?

The *Scope* plane narrows on potential functional and content requirements. Using the research conducted in the previous plane, the product team identifies the most important user needs and ideates the ways of supporting those user needs. This is the

stage of solution generation. All solutions are considered and noted, but many are eventually eliminated to fit a feasible *Scope.*

The product teams' *Content Strategist* covers this plane by supplying the team with a *content audit* (Kissane, 2011). This deliverable maintains a list of prioritized content that must be accessed by the user in the product. The team's *Product Manager* may make an executive call on this plane and demand certain features be implemented over others. *Engineers* may ask the team to reconsider supporting certain functions citing the difficulties of implementing said functions.

If the product team designs a *timeline navigator* with a special feature that lets users share their production directly to social media because one user demanded it, the experience will probably be failed by its second most fundamental element. The music producers' work may eventually end up somewhere on social media, but covering this requirement for one user might stretch the experience's *Scope* a bit too far.

The Structure Plane

Team members on the *Structure* plane plan out a user's journey through the experience. Here, the product team assigns the requirements to articulated user workflows. On this plane, the product team considers ways to support tasks in the context of defined scenarios. Many of the requirements from the *Scope* plane will fall out of favor as the product team concentrates on contextualized interactions. Questions often asked on the plane:

- How can the user best accomplish certain tasks?

- Which information must be accessed first, second, third, etc.?

Again, Garrett splits the plane based on content and function. The *Information Architect* orders the content in an understandable fashion. This team's IA makes sure the music producers do not have to click down through three levels in the menu hierarchy to find the record button. The *Interaction Designer* (IxD), on the other half of the plane, finds ways of supporting the functions required with appropriate workflows. If users need to annotate their time-series data as it is captured, the IxD finds ways of helping users step through intuitive interactions to achieve their goal.

There are many ways a digital product's experience can be failed by the *Structure* element. If the product team implements a design that violates the five principles of IxD (consistency, visibility, learnability, predictability, feedback) users may have trouble stepping through the interactions. If the product team finds that their users are showing the five symptoms of *information anxiety* (listed in Chapter 1), IAs may need to consider other ways of helping users understand the content required.

The Skeleton Plane

Product team members covering the *Skeleton* plane focus on the visual presentations' *rough* form. Wireframes are a common deliverable. Garrett splits the *Skeleton* plane into the most divided element of the five. While on this plane, members on the product team shape the navigational, information, and interface design. Questions that might be asked on this plane for this application include:

- What specific words make the most appropriate label?

- How can the time-series data be most visible?
- Where might the playhead be placed?

Information Designers need to contribute in the visualization of both the product's content and function. They must position content on the screen well enough to help users understand the content's meaning and a function's affordance. Those information designers working on the *timeline navigator* product team need to make sure the data visualized is not an eyesore and does not obscure the data's meaning with unnecessary graphics.

Professionals from the *Structure Plane* may inform on the *Skeleton* plane as well. IxDs may work with a *Graphic Designer* to shape the user interface (UI) and prevent unfamiliar changes to the layout as the user accomplishes a task. *IAs* on the product team may work on this plane to form the menu navigations required to execute the various functions in the *timeline navigator.*

The Surface Plane

The user makes visceral contact with the surface plane. In many cases, but certainly not all, the product team would typically put the experience's finishing touches on this plane. Many graphic design principles and guidelines will be applied in this last stage. Questions asked on the plane for *timeline navigators* might include:

- Which typography to use?
- What color does the audio data need to be?
- What will the click-actions look like?

Because this element is constantly interacting with the user's five senses, any encounter with a usability issue must first be checked here. Before looking to dig deeper to other planes, the product team might want to make sure this element is not failing the user's visceral experience first.

Concrete vs. Abstract Planes

Decisions and their abstraction level become increasingly more concrete as teams move up the planes. The product's process moves upward into the more concrete planes as measured progress is achieved in the preceding plane. This rise in measured progress as teams move up the framework drives the product from concept to launch.

The distinct planes help communicate how to differentiate a product's user experience. However, their distinctions do not mean the decisions made on that plane are isolated from surrounding planes. Professionals like Information Architects and Interaction Designers primarily work on the structure plane, but they may work with a development team to form rough prototypes on the *Skeleton* plane. Entrepreneurs or executives might be the primary team members determining a product's business objectives in the *Strategy* plane, but they may help select various color palettes on the *Surface* plane to match the product's branding.

Each of these five planes entails the various decisions made at each stage in the product's development. The questions asked, deliverables handled, and resulting solutions can all be mapped to Garrett's framework. Herein lies Garrett's critical notion: each user experience has an inherent conceptual model.

Thinking with Models

Helping others reach a conceptual understanding can often begin with pictures (Roam, 2008). Human vision helps the species spot natural hunters. Consequently, human vision keeps the species' verbal competence at a distant second place. A very large portion of the human brain is wired for visual competence (Hoffman, 2000). In cogitating concepts, humans can work with visualizations to supplement verbal understanding.

Pictures are said to be worth 1,000 words, relieving the brain heavily loaded by text. Noting readers' and a language's limitations, this thesis pairs the written word with illustrations. The following picture fits Garrett's framework between two very important parties.

The UX Dialogue

Garrett's elements, layered behind a visual identity, stand between the product team and the users. In figure 2.02, the elements channel a unique dialogue between the parties as the dialogue's medium.

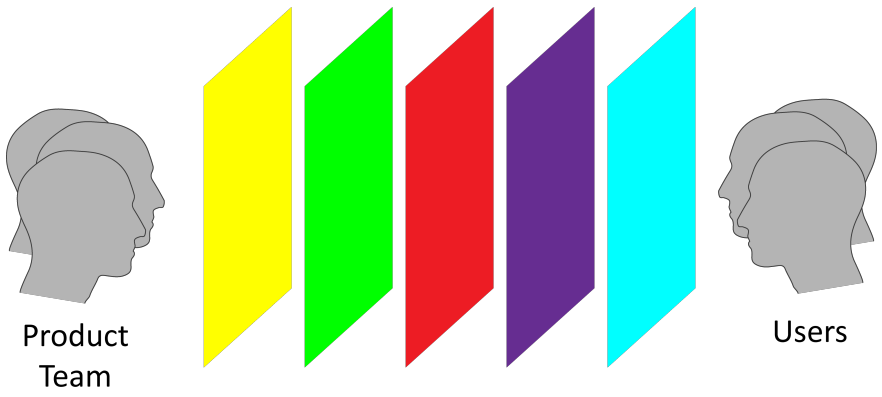

Figure 2.02 *– Think of Garrett's elements as the product team's planes and the dialogue's layers.*

While Garrett's elements work to frame the dialogue within teams, their work is ultimately channeled via sticky notes, sketches, lists, and code. In the case between producer and user, Garrett's elements are the medium. In most cases, the producers can be made up of anyone forming the user experience and the users can obviously be anyone interacting with the experience. In this thesis' case, the figurehead on the right represents *timeline navigator* users and producers trying to piece apart a *timeline navigator's* UX are represented by the figurehead on the left.

Keeping this dialogue in mind is important because UX practitioners argue that one can typically expect errors when a product's conceptual model does not match the user's mental model (Weinschenk, 2011). In others words, errors occur when the product does not function the way the user thinks it functions.

The conceptual models in most *timeline navigators* are very complex. Their elements often host an intricate dialogue with the user. Product teams, new to the genre, can benefit from the defined conventions. In an effort to define the genre, this thesis hopes to clear the uncertainties behind the decisions made along the 5 planes. It hopes to introduce precedence. Product teams, forming *timeline navigator* experiences, can read this thesis and rule out concerns and generate constraints. They can work within reason because they know the conventions are in place.

Chapter 3 sets up the methodology used to conduct the analysis in Chapter 4. An example-based analysis in Chapter 4 points toward other timeline navigator user experiences and lists conventions and highlights common practices. A discussion found in Chapter 5 maps the conventions and practices to Garrett's framework. It covers how

the *timeline navigators* analyzed share UX elements and defines the genre based on the similarities drawn.

CHAPTER III

METHODOLOGY

This thesis proposes a software genre definition other designers can recognize, understand and apply to their designs. Chapter 2 introduced the concept of classifying software in UX terms by incorporating Garrett's framework. The methodology presented here in Chapter 3 aims at three primary objectives:

- Select candidates for classification
- Identify the similarities found in the applications selected
- Define conventions and format them into a pattern language

To define a genre of software in these terms, an analysis must investigate the observable components in each UX element.

Selecting Candidates

In *Designing Interfaces*, Jenifer Tidwell discusses "two big effects on the craft of interface design" (Tidwell, 2011). The first effect describes recognizable *interface idioms*. She describes these idioms as parts of the whole interface. These *idioms* Tidwell refers to are the successes of the interface design discipline. Users, in the aggregate, select an effective pattern and the pattern becomes convention. People get used to the way things work, and the users' mental resources are saved when the interface has the familiar design patterns. This is why a higher value on a temperature dial should always mean hot and a lower value should always mean cold.

Tidwell, while armed with years of experience, must still rely on her five senses to recognize the patterns found in each interface. She makes her assessments beginning with Garrett's most accessible element: the surface layer. Unless one has access to the particular design methods or decisions made in the interface's formation, any analysis must begin with the surface layer from the user's end of the dialogue. In adopting the user's perspective, the method employed in this thesis places initial focus on the interactions afforded by the interface called *affordances*.

Coined by James J. Gibson, an *affordance* refers to the actionable properties of any given object (Gibson, 1977). For example, the following interface affords a number of interactions.

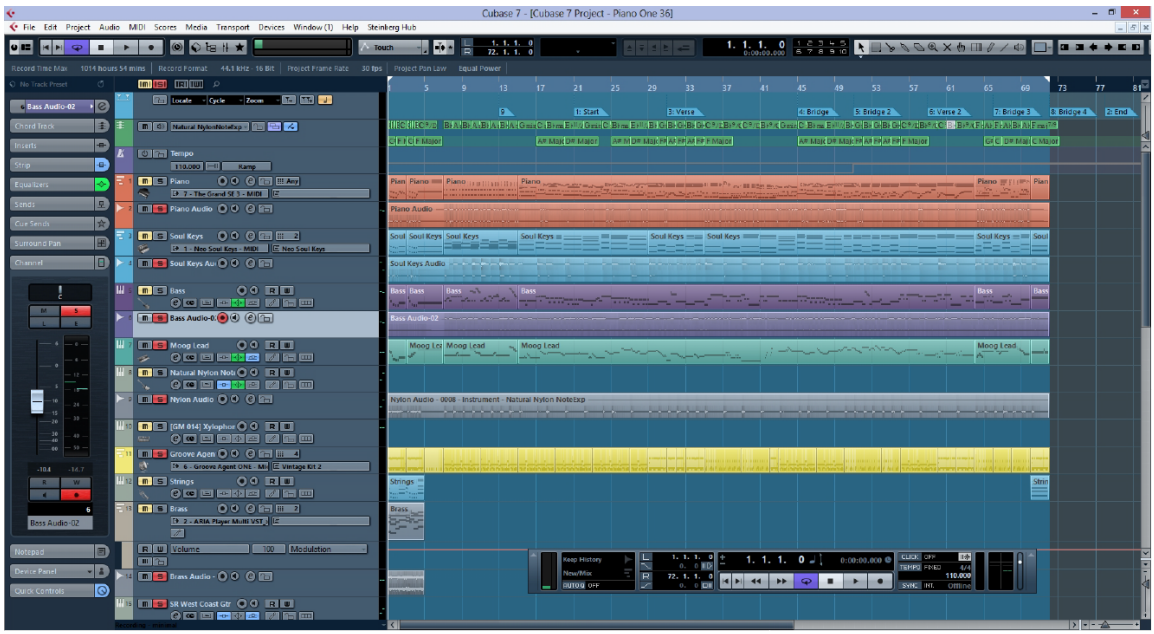

Figure 3.01 *Cubase 4 – Take a look at the grey squares containing shapes near the bottom of the screenshot* (cubase-project001.jpg, 2013).

Figure 3.02 *Identifying Patterns – An example of the Media Player design pattern*

(cubase-project001.jpg, 2013).

Depending on one's familiarity with digital interfaces, the *perceived* affordances found in the highlighted section of the above screenshot should immediately inform one of potential interactions (Norman, 1999). This is a *media player*. This pattern is often used to record, pause, play, and in this particular case, loop time-series data. It is an essential part of any interface affording the interaction with time-series data. This pattern can be found on audio players, video players, simulators, and data analysis tools. Many applications adopt this pattern and adapt it to their application's visual style guide, rendering many *media players* into visually distinct components on the application's surface layer. Below the superficial surface layer, however, this group of objects can be considered one pattern because they usually support the same tasks.

In the context of interface design, the many versions of the *media player* are grouped and considered one design pattern due to the similar affordances found in each version. Beginning with sensory input found on the surface layer, users can reasonably expect a familiar interaction from something like a *media player* because they have seen the pattern before. By observing hard lines and white space differentiating patterns from one another, the analysis identifies the patterns in the similar format displayed in Figure 3.02.

Like the example above, if an application affords the capture, manipulation, *and* visualization of time-series data, with patterns like the *media player*, a pattern observable in the surface layer, it is considered a candidate for the analysis. By these conditions alone, dozens of applications would need to be considered. For example, the options available in Digital Audio Workstations (DAW), a *timeline navigator* subgenre specifically made for audio production, can be "staggering" (Eskow, 2001).

To limit the breadth of the analysis, this thesis targets only one industry leading software application from each professional domain. However, the scope of this thesis prevents it from analyzing the tool used in every professional domain. Again, to limit the analysis, this thesis targets those professional domains that use a *timeline navigator* as one of their primary tools.

The analysis considers four different interfaces to comprehensively define the genre. Each interface, in the form of screenshots, is gathered by searching the Internet using the Google search engine. The application's popularity is determined by credible consumer reports and product reviews. Using industry leading software, the analysis

ensures each application has thoroughly-revised interfaces and user experiences. These designs benefit the analysis by providing matured conventions and practices.

Identifying Similarities and Conventions

The second effect Tidwell recognizes acknowledges the "loose" nature these patterns are applied to different software. Drop-down menus, for example, are almost ubiquitous in the way they show and hide options. Action panels, similarly, can be found in word processors, spreadsheets, and webpage browsers. Because the applications considered in the analysis will in all likelihood serve different user contexts, the patterns observed in one *timeline navigator* may be applied differently in another. To identify similarities, thus narrowing the genre definition, only patterns observed across the interfaces considered in the analysis are presented in the results.

A software genre cannot be defined using the superficial access points in the skeleton and surface layers alone. With more information found deeper in the user experiences of each application considered, this thesis classifies real commercial products definitively and practically. By investigating the user manuals accompanying the interfaces analyzed, this thesis observes the similarities existing in the structure and scope layers of each user experience.

The analysis references information found in available instructional material to determine how each pattern supports certain tasks. Using the interview coding method, the language in the user manual is coded into one phrase describing the task (Hawes, 1972). In the following example, the operation manual for an application called *Cubase*

7 is used to determine how the *inspector* and *mixer* patterns are used to support the task: “Enable Data Source for Recording.”

Record Enable in the Inspector, Track list and mixer.

Record-enabling a track

Cubase can record on a single track or on several tracks (audio and/or MIDI) simultaneously. To make a track ready for recording, click the Record Enable button for the track in the Track list, in the Inspector or in the mixer. When activated, the button(s) turn red, indicating record ready mode.

Figure 3.03 *Instructional Material – These snippets of information help determine how the pattern supports certain tasks (Bachmann et al., 2012)*

Screenshots from the user manuals are not presented in the results section. After the pattern and task data is gathered and analyzed, the results section, like Tidwell, answers the following questions for each pattern:

- What does the pattern do?
- When should the pattern be implemented into a design?
- Why is this particular pattern so important to the user experience?
- How should this pattern be implemented into a design?

Each answer is informed by the information observed from the analysis. The answers provided in the results are intended to help designers become aware of the major practices and conventions found in the *timeline navigator* software genre. This is otherwise known as a *pattern language* (Ishikawa & Silverstein, 1977).

Formatting the Pattern Language

The results section is formatted in the following manner.

DESIGN PATTERN: design pattern 1

Application Name

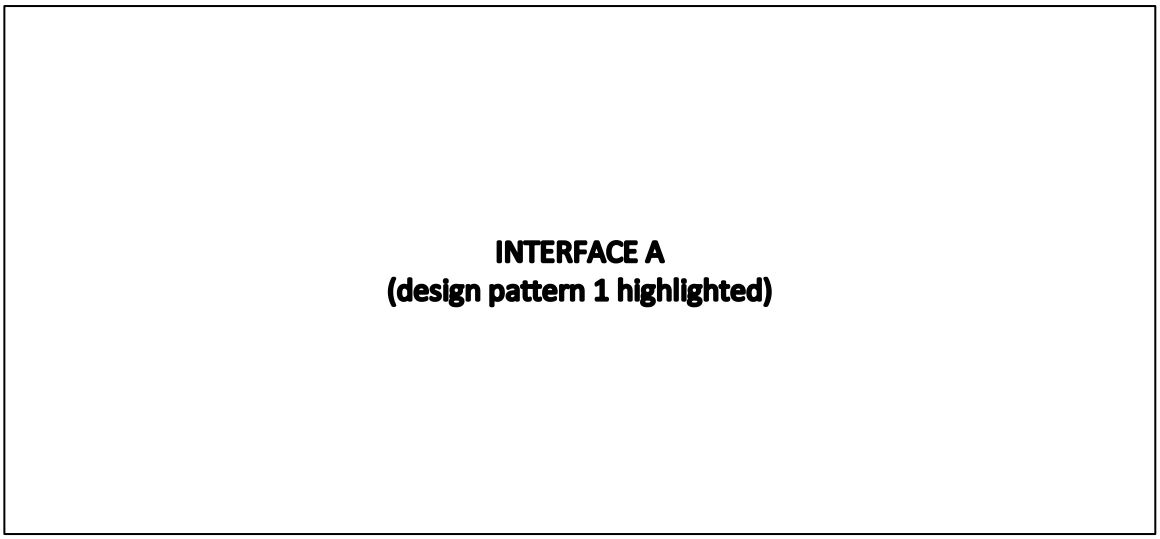

Figure title and description

Application Name

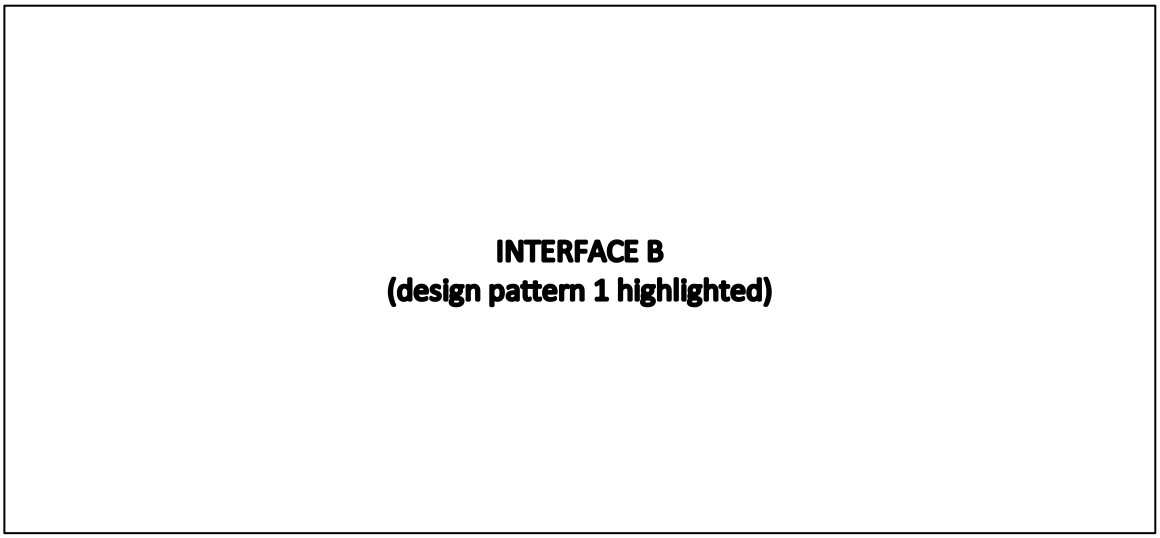

Figure title and description

Application Name

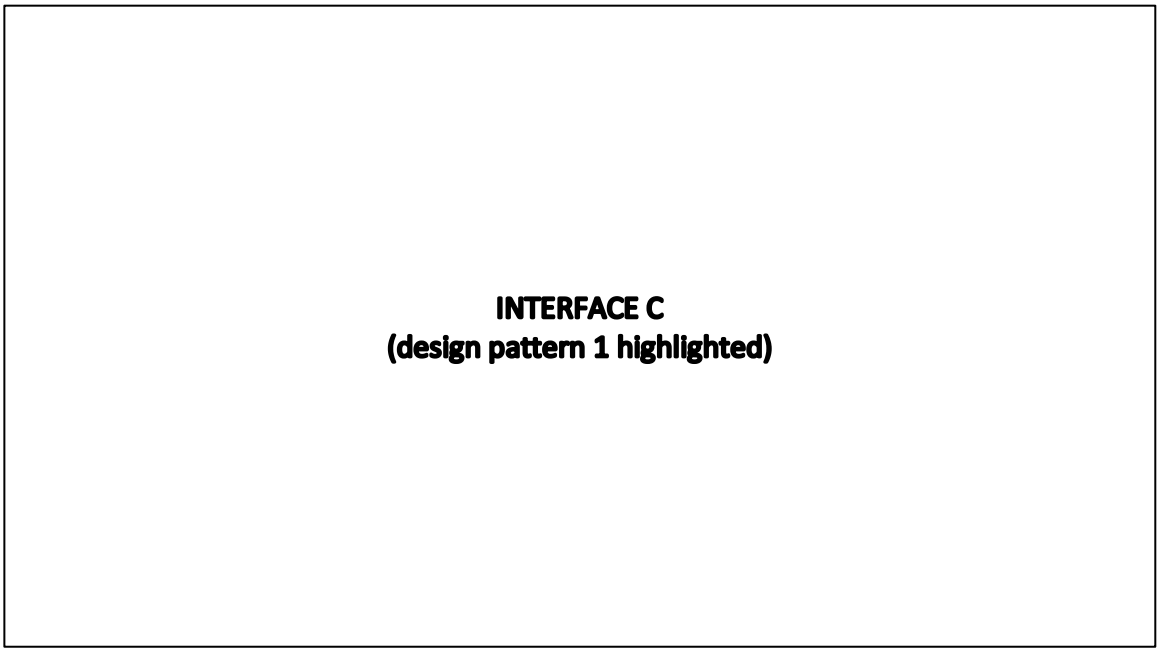

Figure title and description

Application Name

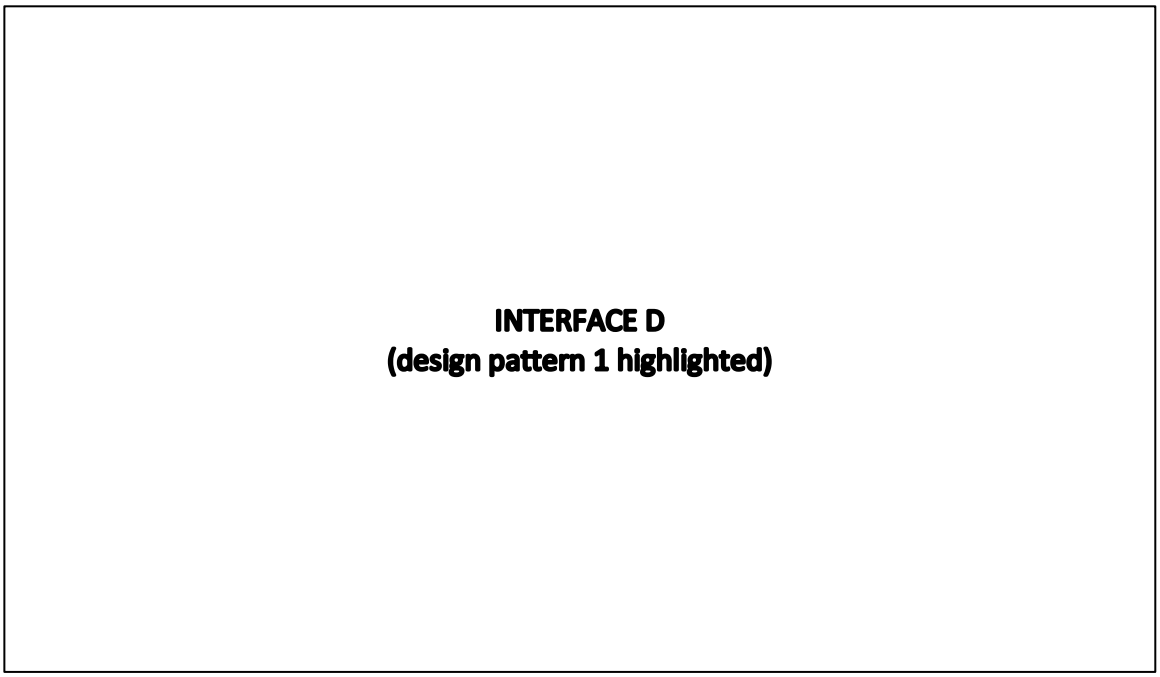

Figure title and description

TASKS SUPPORTED:

- Task supported #1
- Task supported #2
- Task supported #3
- Task supported #4
- Task supported #5
- Task supported #6
- Etc.

What does the pattern do?

[Answer]

When should the pattern be implemented into a design?

[Answer]

Why is this particular pattern so important to the user experience?

[Answer]

How should this pattern be implemented into a design?

[Answer]

DESIGN PATTERN: design pattern 2

Application Name

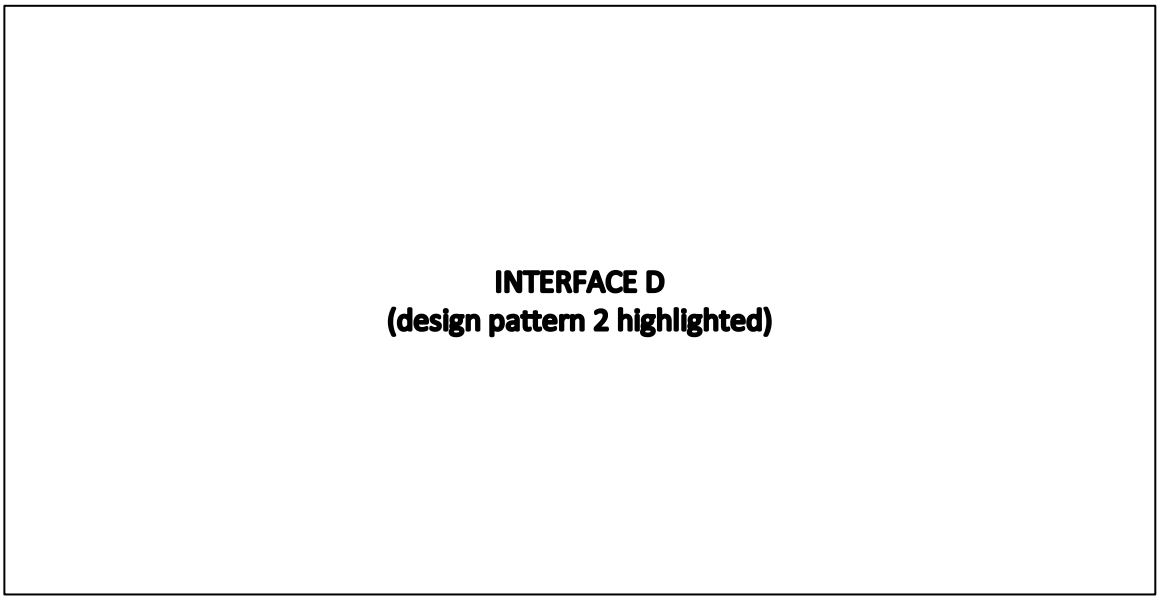

Figure title and description

This format continues as presented for all design patterns.

CHAPTER IV

RESULTS

DESIGN PATTERN: Data Sword

Protools

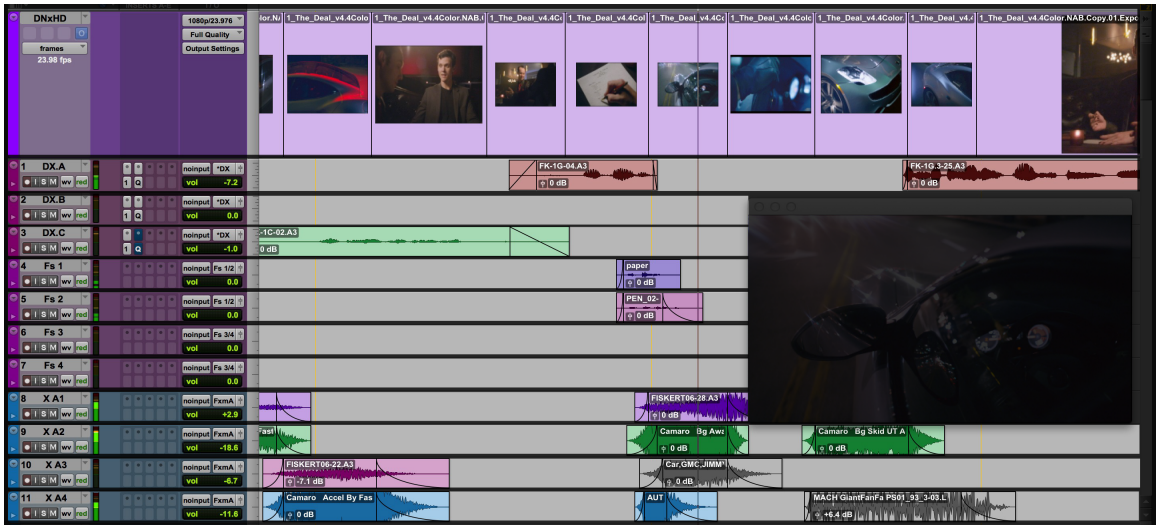

Figure 4.01 *– Data Swords in Protools* (avid_ProTools11_video.jpg, 2013)

SportsCode

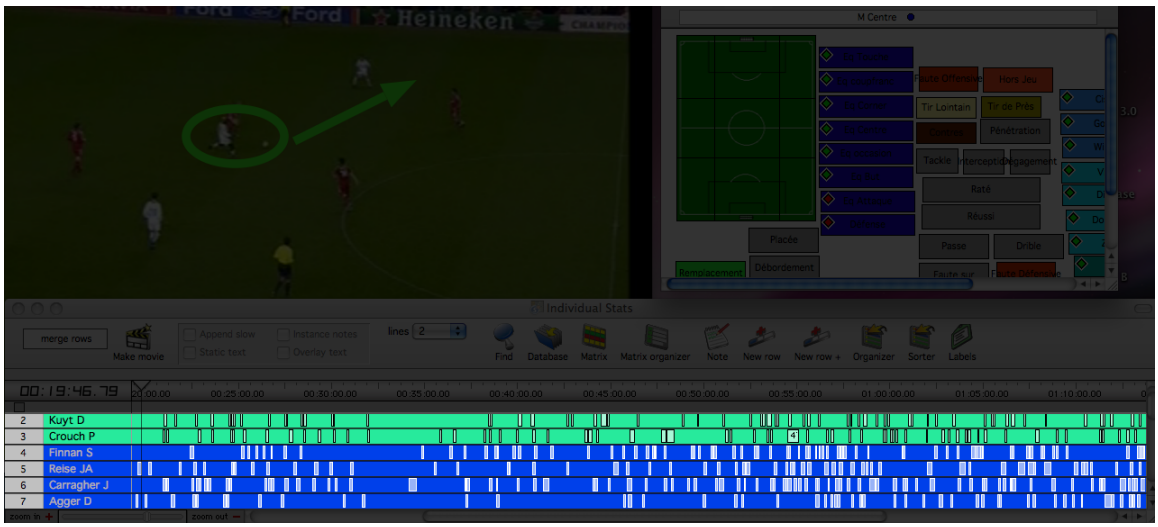

Figure 4.02 *– Data Swords in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

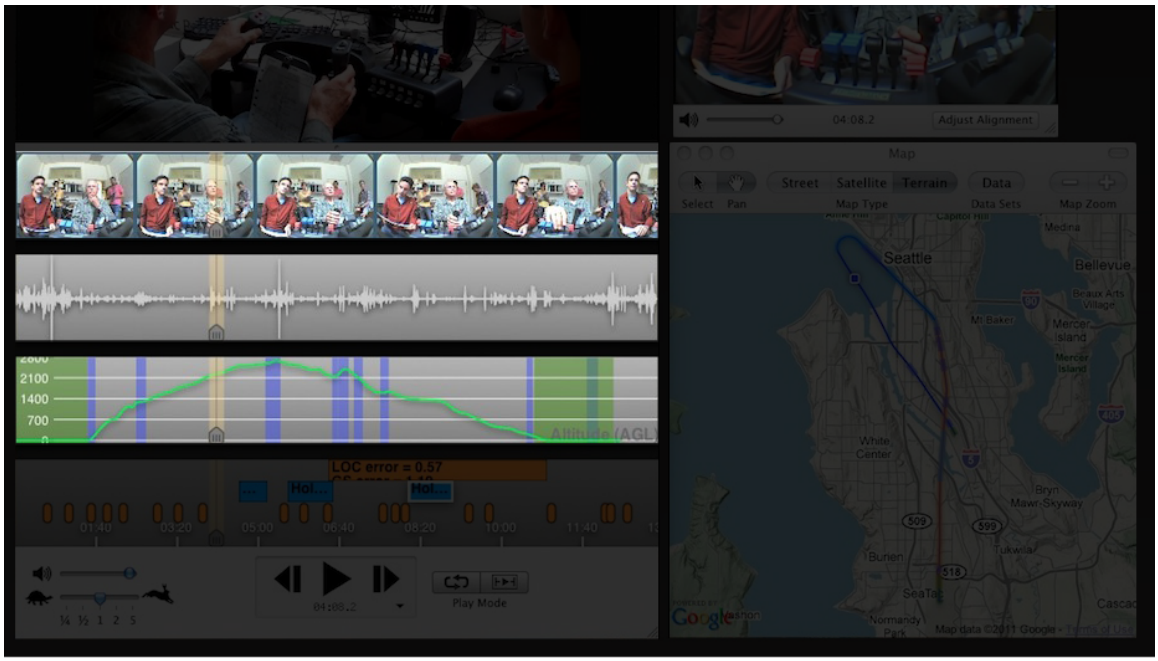

Figure 4.03 *– Data Swords in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

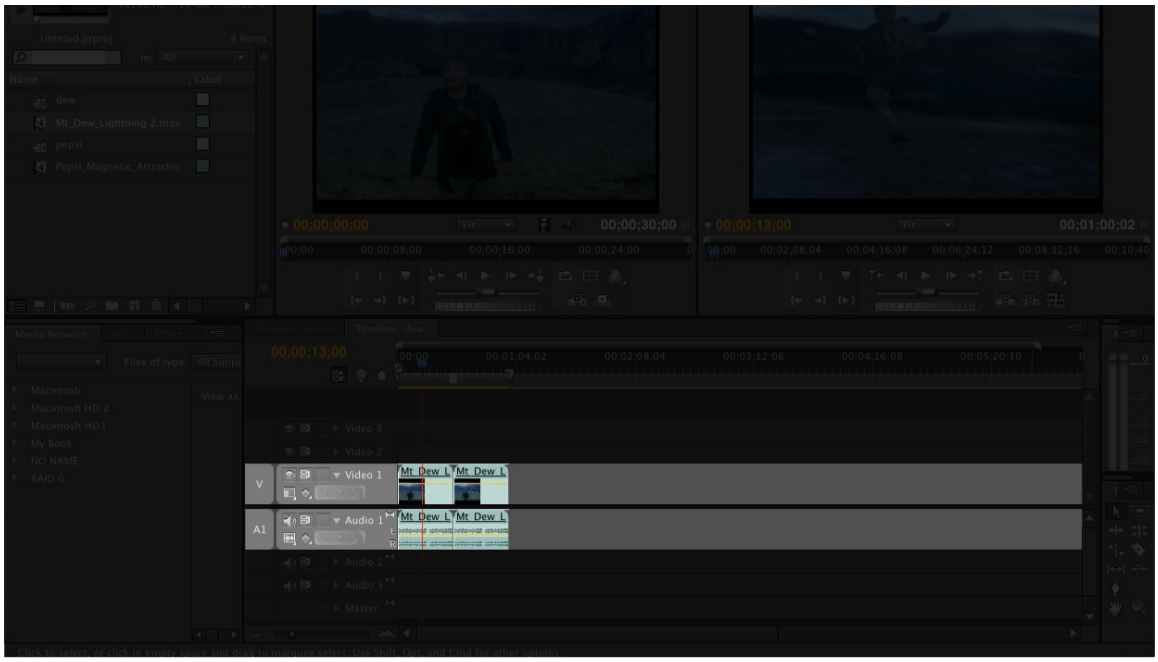

Figure 4.04 *– Data Swords in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- Display time-series data – This pattern is used to visually represent the data as it is indexed on time.
- Import a source of time-series data – Users can use this pattern to impor their data from external applications or sources.
- Adjust the temporal positioning of a segment of time-series data – Users can drag and drop segments of time-series data along the *timeline.*
- Display annotations – Users can use this pattern to view annotations or metadata specific to the data source.
- Sort a list of time-series data sources – Users can order how these patterns are listed. This helps users organize their sources of time-series data.

What does the pattern do?

This pattern is the most essential tool in affording interactions with the time-series data. Up to four design patterns discussed in Tidwell's book can be found in the *data sword*: the *multi-y graph*, the *button group*, and *small multiples* (Tidwell, 2011). A "handle" attached to the time-series data characterizes the *data sword*. The "handle" is often a *button group* which can be used to sort, edit, and capture the individual source of time-series data. Multiple *data swords* are stacked along multiple y-axes, hence the similarity to Tidwell's *multi-y graph*. If the time-series data is a video, small multiples can be used to visualize each frame of the video.

When should the pattern be implemented into a design?

Use this pattern when users need to manipulate time-series data. The presentation of the data attached to the *button group,* or data sword handle, lets users make individual changes or edits to each data source. Use this pattern when users need to visualize time-series data as well. In terms of Tidwell's pattern language, the data sword is a lot like a more interactive *multi-y graph*. The characteristic stacking of the *data swords* helps users compare the data on one baseline. For example, users may need to compare events in the time-series data to understand their sequential order.

Use this pattern when users need to capture individual sources of time-series data. *Protools*, for example, allows users to record individual instruments at a time. They can also record multiple instruments by enabling the record button on each "track." This record function is enabled on the *data sword*'s button group.

Not all the *timeline navigators* investigated supported the capturing of time-series data with the data sword. Applications like *ChronoViz*, import the time-series data using a global import function.

Why is this particular pattern so important to the user experience?

According to Garrett, information can be structured in one of four ways (Garrett, 2010). The *sequential* method can be found in books, audio, and video. The *organic* structuring of information can often be found in the use of metadata like tags or keywords. The *hierarchical* information structure is the common form of information architecture used in websites.

Timeline navigators, with the help of the *data sword* pattern, organize information using the *matrix* structure. This helps users navigate the information in more than two dimensions. In *timeline navigators,* the matrix's dimensions are often time, data source, and data category. For example, users working with *SportsCode* may have access to three dimensions of the time-series data captured at a football game. Using the *data sword* pattern, they can analyze events according to the time each event transpired. They can analyze a team's performance according to the video captured from one camera angle. They can also analyze categories of individuals from the team. They may want to see how all wide receivers ran their routes or how the defensive ends rushed the quarterback. Using the *data sword's* matrix structure, users get to pivot their analysis on one, two, or three dimensions.

How should this pattern be implemented into a design?

Time-series data can be represented in many unique forms (Tufte, 1990). Because humans perceive changes or events in time sequentially, visual representations may benefit by presenting time-series data in a sequence as well (Le Poidevin, 2011). Every application included in the analysis presented the time-series data with the progression of time going from left to right. The data was also presented horizontally.

In visualizing the time-series data, Tufte's principles of information design strongly apply to this design pattern's effective implementation. To maintain graphical integrity, maintain a consistent temporal resolution across each visual representation from the disparate data sources. If users are allowed to view an increased or decreased

temporal resolution of each data source, make sure the time-series data is represented accurately and matches the scaling effect of the other data sources. Do not include unnecessary graphics.

The generation effect can be leveraged in this design pattern to help users recall their categories or sources of time-series data (Slamecka & Graf, 1978). By allowing users to label each data source, users can distinguish categories or sources of time-series data. In the applications analyzed, this was often achieved by letting users assign colors to each *data sword*.

Give users the ability to sort the data sources, either individually or altogether. This can be done with a sorting menu that lets users sort every data source in the project. For example, users may want to sort each data source in alphabetical order, date added, or by category. The organization of these *data swords* may also benefit from giving users the ability to drag and drop data sources to new locations in the list.

To manipulate the time-index of time-series data, users may need to move segments of time-series data along the horizontal time-axis. Some applications afford this by letting users hover over the segment of time-series data and dragging and dropping it to other temporal locations. More precise alterations can be afforded with taps of the keyboard arrows or specific “nudge” buttons on the interface. To prevent the accidental modification of each data source’s time-stamp, the *data sword*’s handle often contains a padlock icon that “locks” the time-series data from any modification.

DESIGN PATTERN: Timeline

Protools

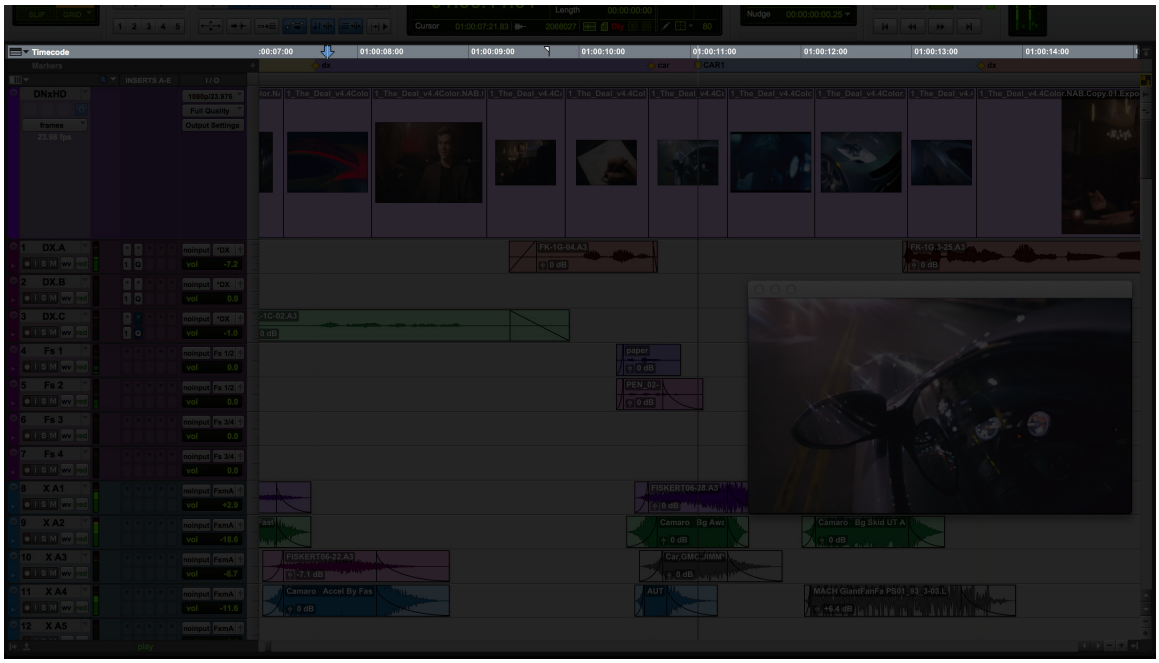

Figure 4.05 – *Timeline in Protools* (avid_ProTools11_video.jpg, 2013)

Sportscode

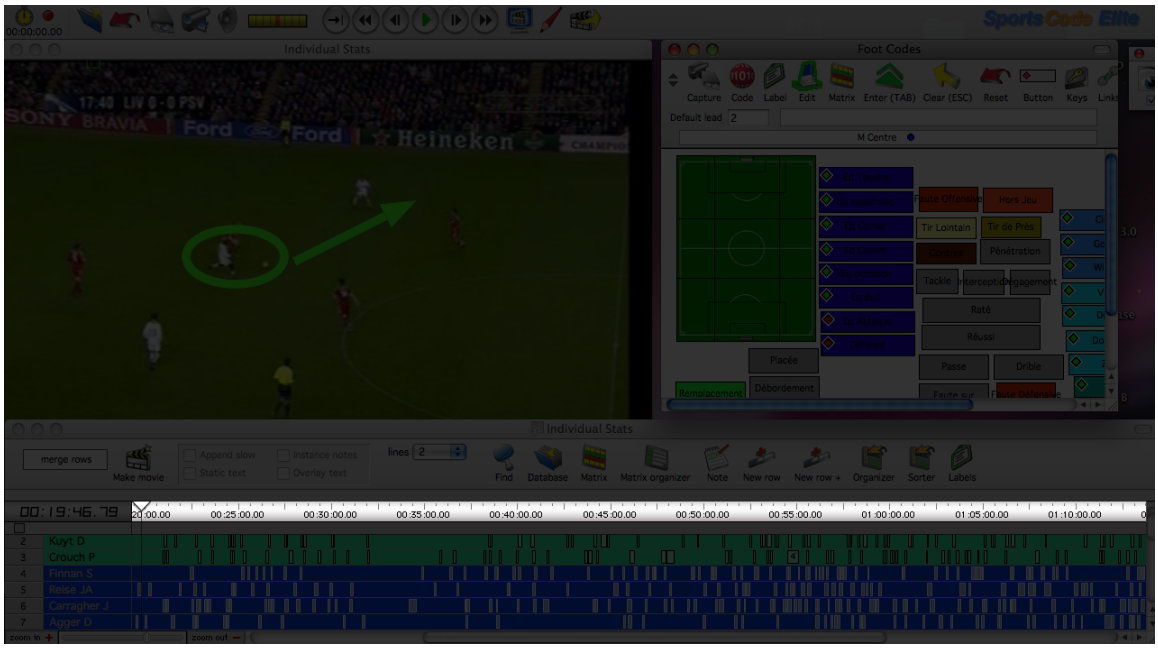

Figure 4.06 – *Timeline in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

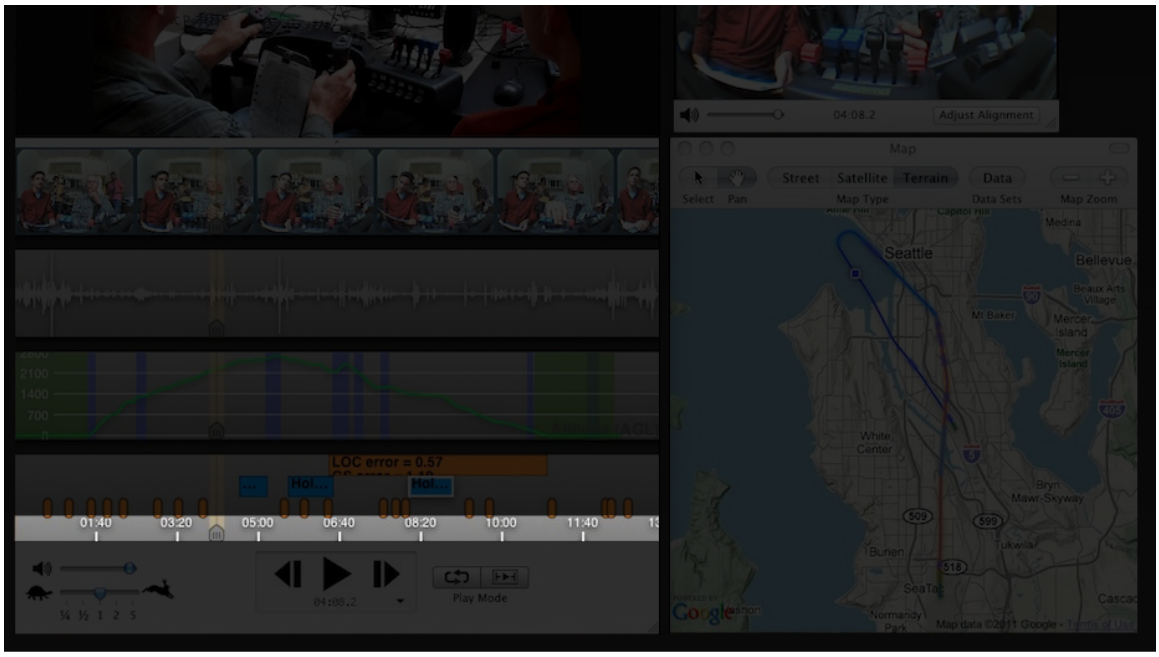

Figure 4.07 *– Timeline in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

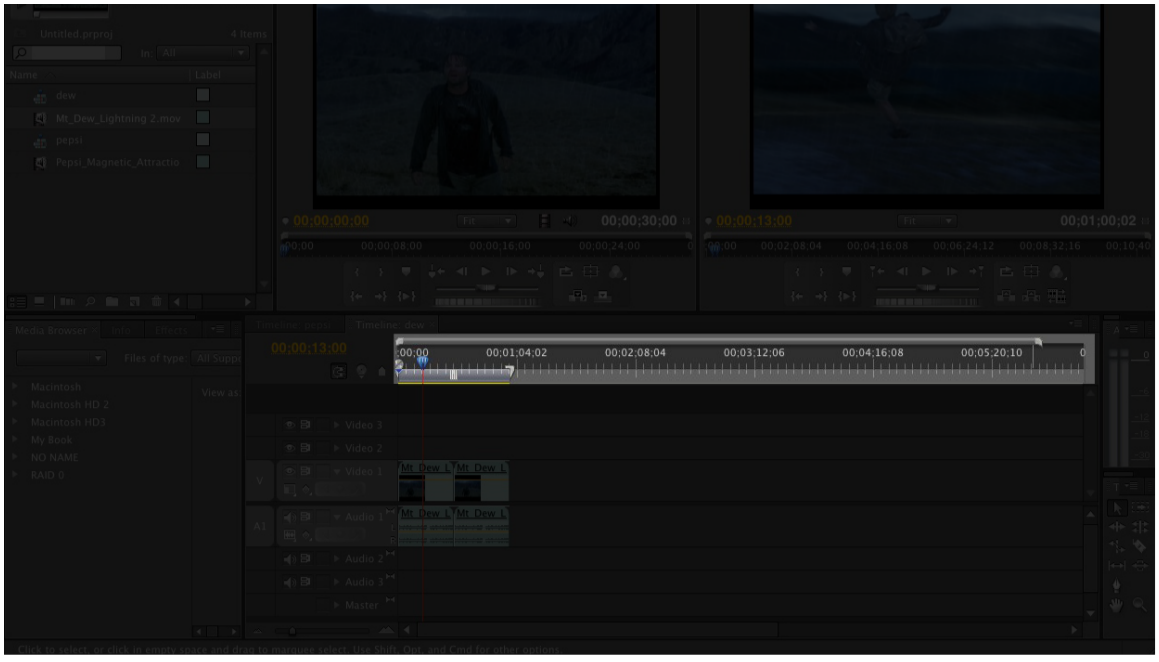

Figure 4.08 *– Timeline in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- Reference the playhead's temporal position – Users can use the *timeline* to understand the playhead's position in time.
- Reference a selected item's temporal position – Users can select items like segments of time-series data and use the *timeline* to understand their position in time.
- Reference an annotation's temporal position – Users can view the annotations based on the *timeline* to determine their position in time.

What does the pattern do?

The timeline acts as a temporal ruler for the project in the *timeline navigator.* It visualizes a consistent range of time users can use to judge the position of the playhead or segments of time-series data.

When should the pattern be implemented into a design?

Use the timeline design pattern when users need a quick and efficient visualization of their data source's time-stamp. This pattern is also effective at helping users visualize the temporal positions of other items on the interface like annotations or the playhead.

Why should this particular pattern be implemented over other alternatives?

By leveraging spatial perception, the horizontal timeline allows users to immediately recognize a data set's position in time. The design pattern takes up very little space on the screen while maintaining the most essential information. Like a ruler, the horizontal timeline provides the user with a constant reference point. In many cases, the timeline is unable to provide a precise measurement of a data set's position. Do not rely on the timeline for the absolute measurement of time. Because the timeline can represent a clear starting point, end point, and range, use the timeline design pattern to help users judge their data set's position relatively. At the macro-level of temporal resolution, users can judge the entire project's relative length. At the micro-level of temporal resolution, users can focus on the smaller units of time.

How should this pattern be implemented into a design?

In the interface's layout, limit the spacing between the time-series data content and the timeline. Users may wish to refer to the timeline to judge their data set's position as well as the position of their cursor in the cases of editing, scrubbing, and play head positioning. By keeping these two pieces of interface together, the design cuts down on eye-gaze travel when they wish to accomplish tasks requiring the whereabouts of their cursor and/or data set in respect to the *timeline*.

Provide a clear distinction in the time values displayed on the timeline and always use a sans-serif typeface to display the time values. The larger time values like hours are typically displayed on the far left of each time display followed by minutes,

seconds, and milliseconds. Different time values can be separated with a colon like the following example: HH:MM:SS:MS.

DESIGN PATTERN: Annotations

Protools

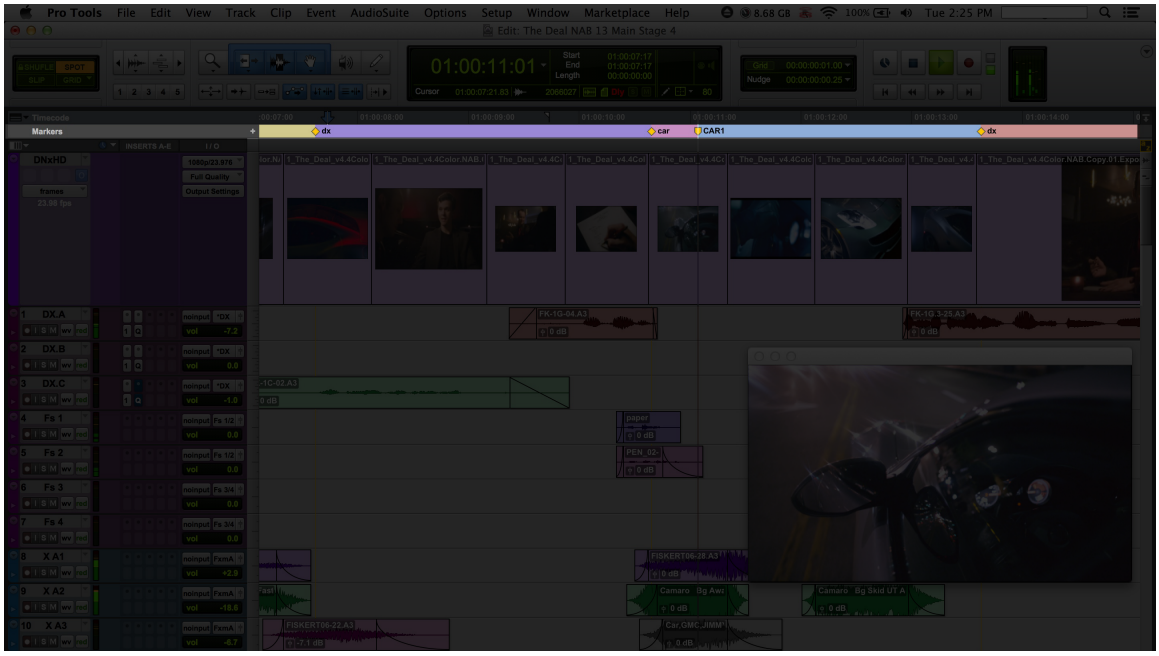

Figure 4.09 *– Annotations in Protools* (avid_ProTools11_video.jpg, 2013)

SportsCode

Figure 4.10 *–Annotations in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

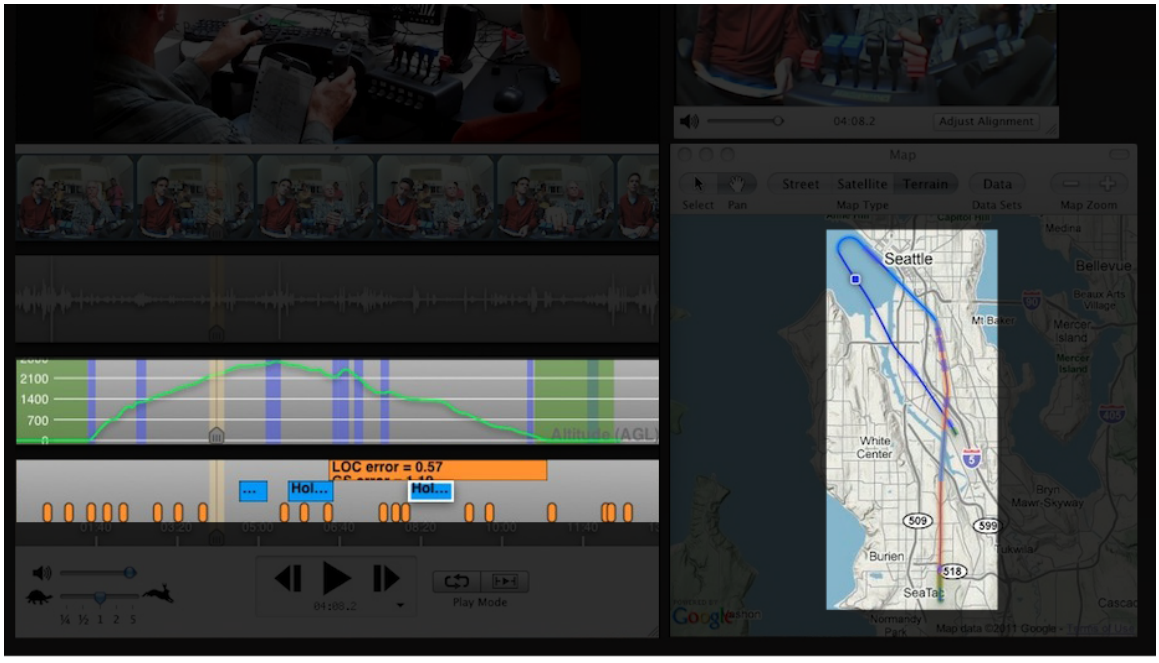

Figure 4.11 *– Annotations in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

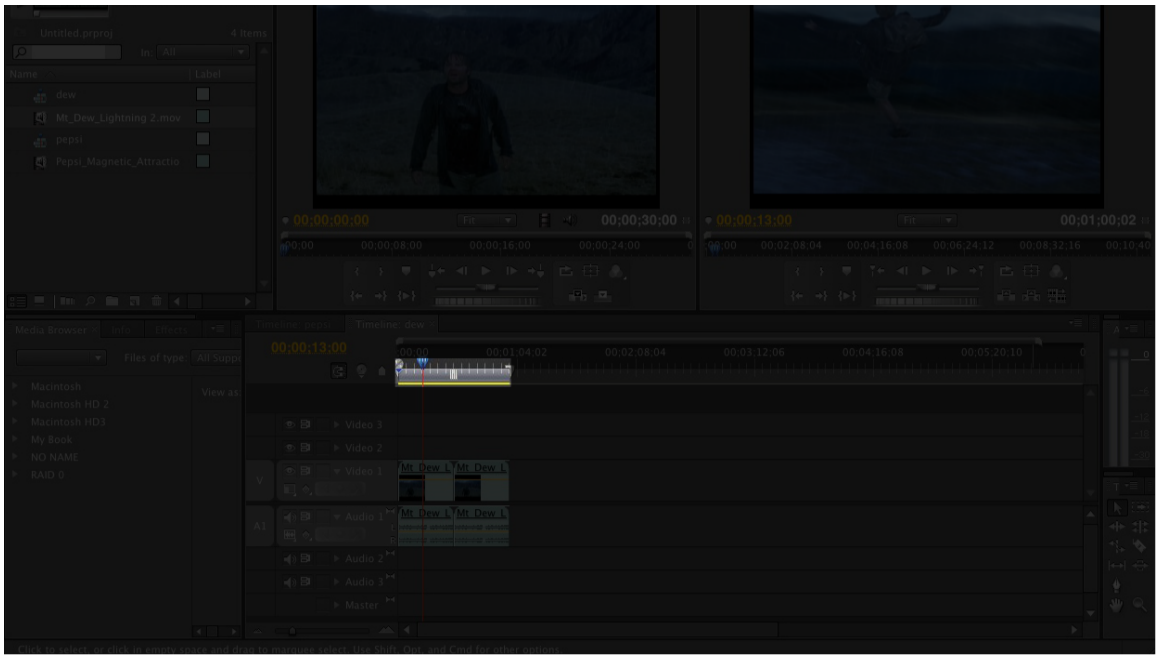

Figure 4.12 *– Annotations in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- Highlight events in the data – Users can mark sections of the *timeline* to indicate special events in their data.
- Add metadata to annotation – Users can view an annotation and add more information regarding the particular annotation.
- Define loop range – Users can mark a section of the *timeline* that will limit the playhead's playback to the defined range.

What does the pattern do?

Annotations come in many shapes and sizes, but they often populate the *timeline* and/or *data swords* as colored markings. These markings often signify

important events within the time-series data. Users typically encode their own annotations based on their project's context. For example, Sportscode users may want to populate one source's *data sword* with annotations signifying each time a team scores. These annotations can be added using a predetermined shortcut key that gets pressed by a reviewer as the event occurs during playback. Users can also drag and drop markers into position as well. A function unique to *ChronoViz* allows users to write down notes during playback using a special pen. The pen records the ballpoint's position and digitally rewrites the user's notes with their own proper time-stamp. These notes can be reviewed as they were written during playback once the pen has recorded them.

Temporary markers placed on the *timeline* can sometimes be used to set up loops for the playhead. For example, Protools users can drag and drop two functional marks across a range of the timeline. During playback, once the playhead reaches the second mark further down the timeline, it restarts at the position of the very first marking.

When should the pattern be implemented into a design?

Annotations provide an extremely important service to users analyzing the time-series data for specific events. Instead of writing down exact time-stamps for each event as they transpire during playback, users can easily click a customizable button on the interface as supported by *SportsCode* or press shortcut key without pausing or stopping the playback. This seamless annotation process allows users to quickly mark the event and continue with the playback process.

Why is this particular pattern so important to the user experience?

Users working with hours of time-series data may not want to constantly pause their playback each time their review process detects a significant event. Providing the function to make quick annotations prevents task interruption. Task interruption, as studies show, can severely deter user performance (Bailey, Konstan, & Carlis, 2001).

Working with hours of time-series data may also place a heavy load on working memory. Once users detect 5-9 events, their working memory is likely to reach full capacity (Miller, 1956). Without markings there to visualize each event's temporal position, the user is not likely to remember it. Visualized annotations also help other reviewers analyze the events without repeating the detection process.

How should this pattern be implemented into a design?

Provide users with a specialized action panel for making annotations. Dragging and dropping may help in cases where users need to casually add notes, but this method will hardily assist those users making heavy annotations across hours of data. In this action panel, users need access to customizable coding for specific events. For example, Army trainers conducting a training scenario may want to mark all recruit injuries, but they may also need to differentiate these annotations from successful projectile hits that also need to be marked. The specialized action panel can provide a way to assign two buttons for each type of event. When clicked, one button could mark the timeline with red triangles and the other with yellow circles. All users have to do is review the footage and click one of two buttons when they detect the event.

If users need to add or review more details from the annotations made, *data tips* can be very helpful at providing this extra information. By clicking on an annotation mark, a small dialogue box can pop up from the annotation mark. Users can enter, edit, or review extra notes attached to the annotation. Metadata can also be presented on the *data tip.*

DESIGN PATTERN: Time Display

Protools

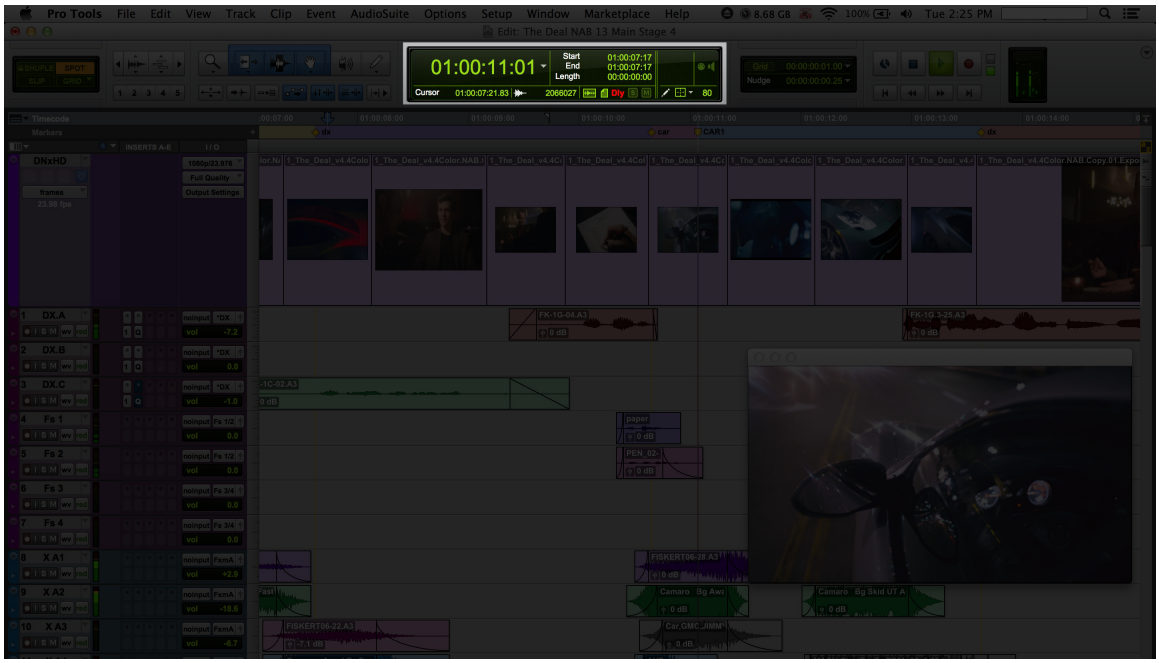

Figure 4.13 *– Time Display in Protools* (avid_ProTools11_video.jpg, 2013)

Sportscode

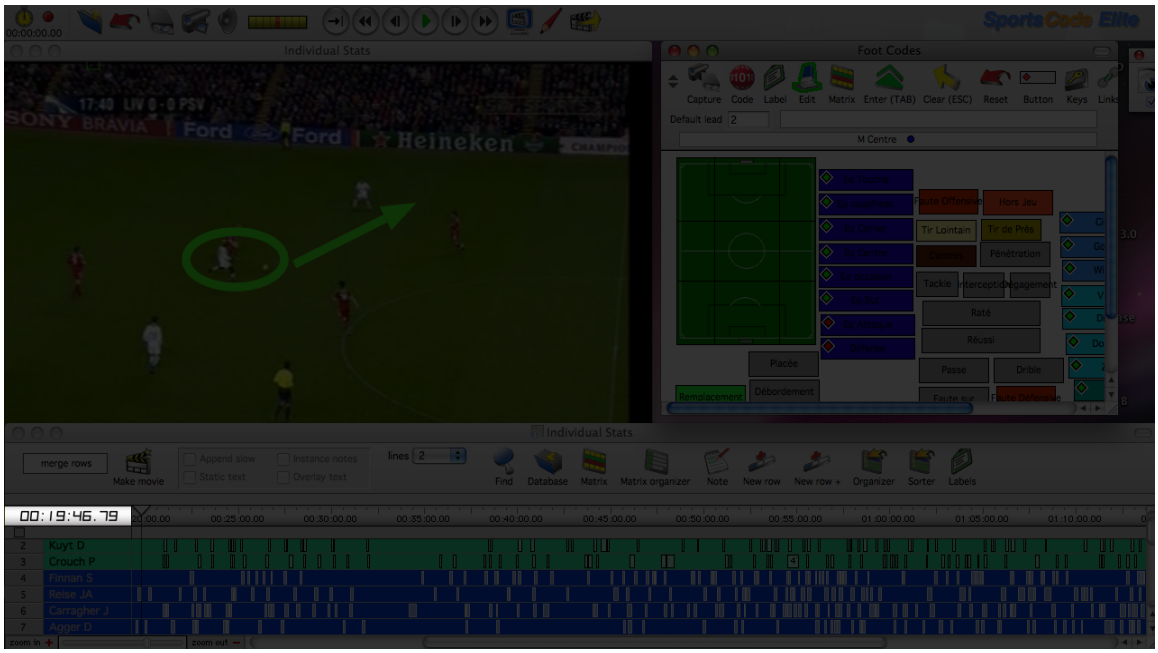

Figure 4.14 *– Time Display in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

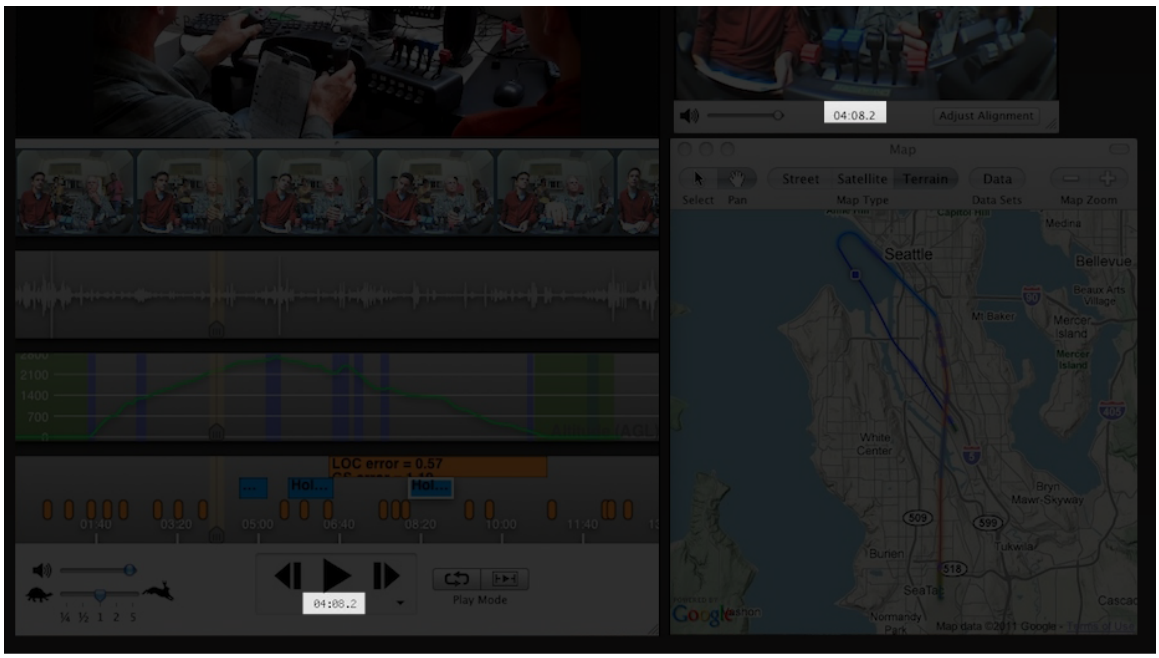

Figure 4.15 *– Time Display in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

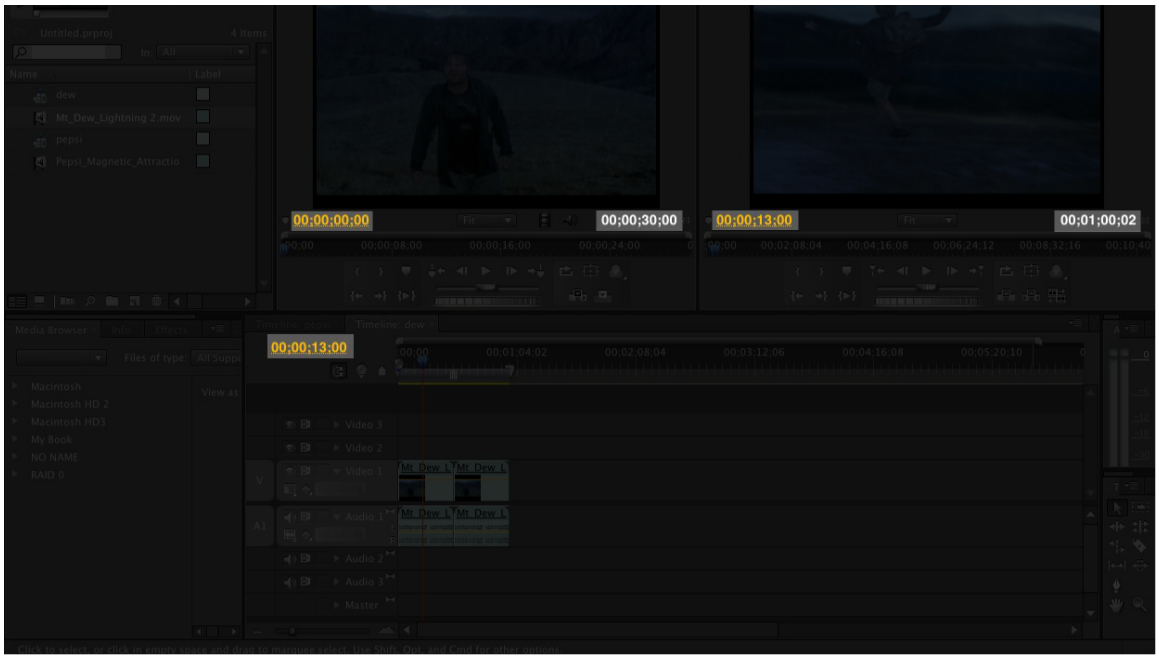

Figure 4.16 *– Time Displays in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- View the playhead's position in time – This pattern helps users know the playhead's position in time represented by a numerical value.
- See the selected item's position in time – This pattern helps users know the selected item's position in time represented by a numerical value. The selected item can be a segment of time-series data or annotation marks.

What does the pattern do?

The *time display* represents the playhead's temporal position commonly as a numerical value. Other time-values can be displayed here as well. If the user selects

other items on the interface like annotations or segments of time-series data, their time-value may also find representation on the *time display* pattern as well.

When should the pattern be implemented into a design?

Use this pattern when users need to understand the exact temporal position of an item in the project. While the *timeline* can be used to visualize an item's relative position in time, the *time display* helps users know the item's absolute position in time.

Why is this particular pattern so important to the user experience?

Users working with time-series data may need to reference an item's temporal position in absolute terms. Trainers, for example, may want to know the exact time an event in the project occurred. These trainers may want to take special note of this metadata for further analysis.

How should this pattern be implemented into a design?

This pattern can be one of the more salient features on the interface. If users are looking to interact with the time-series data more precisely, they may need to reference their data's exact temporal position. A large display may assist users working in this context.

The *time display* typically follows the standard display methods found in conventional digital clocks. Larger time-values precede smaller time-values. A colon may separate hours, minutes, seconds, and so forth. Smaller time-values like milliseconds

may also be displayed if users require a more precise measurement of the item's temporal position.

DESIGN PATTERN: Media Player

Protools

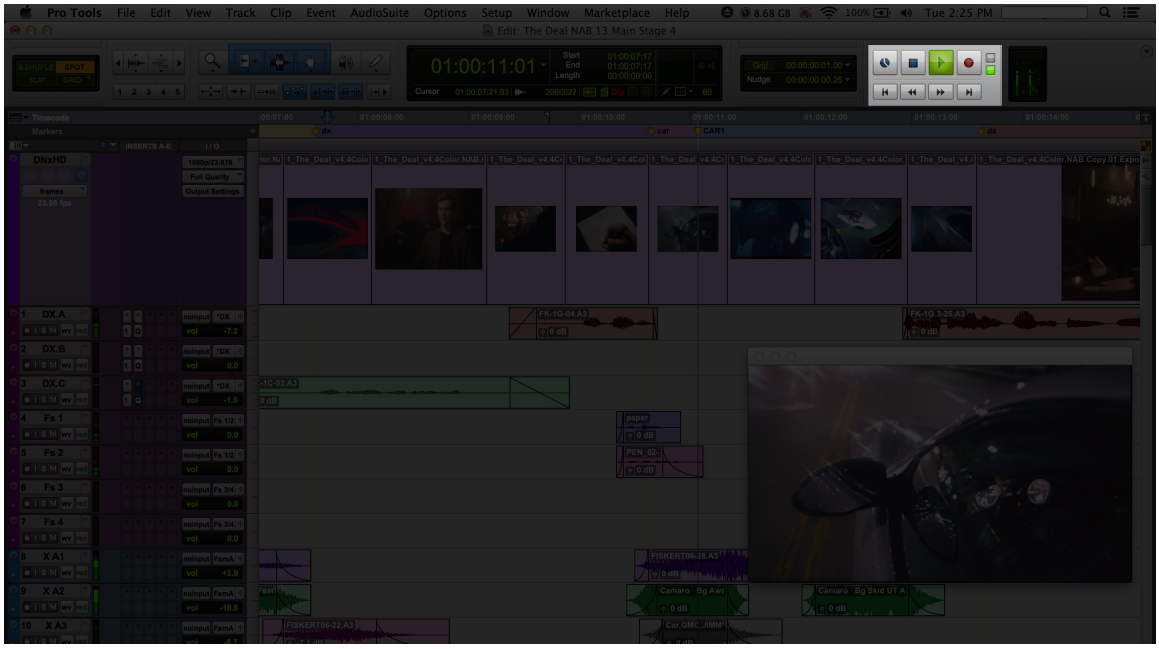

Figure 4.17 *– Media Player in Protools* (avid_ProTools11_video.jpg, 2013)

SportsCode

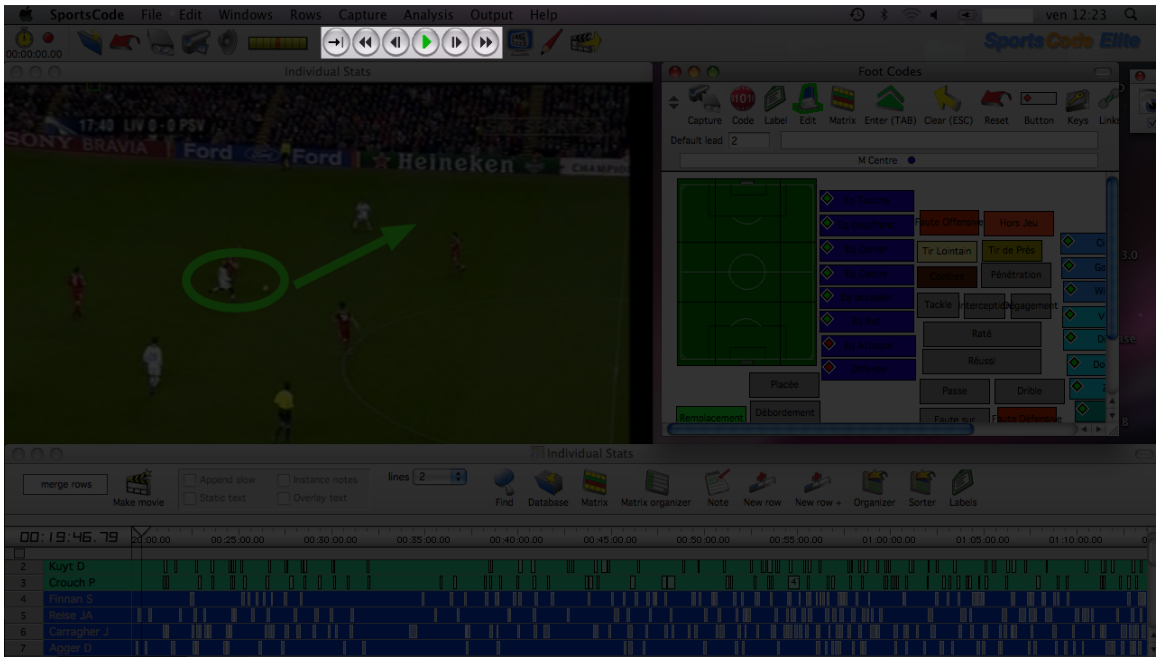

Figure 4.18 – *Media Player in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

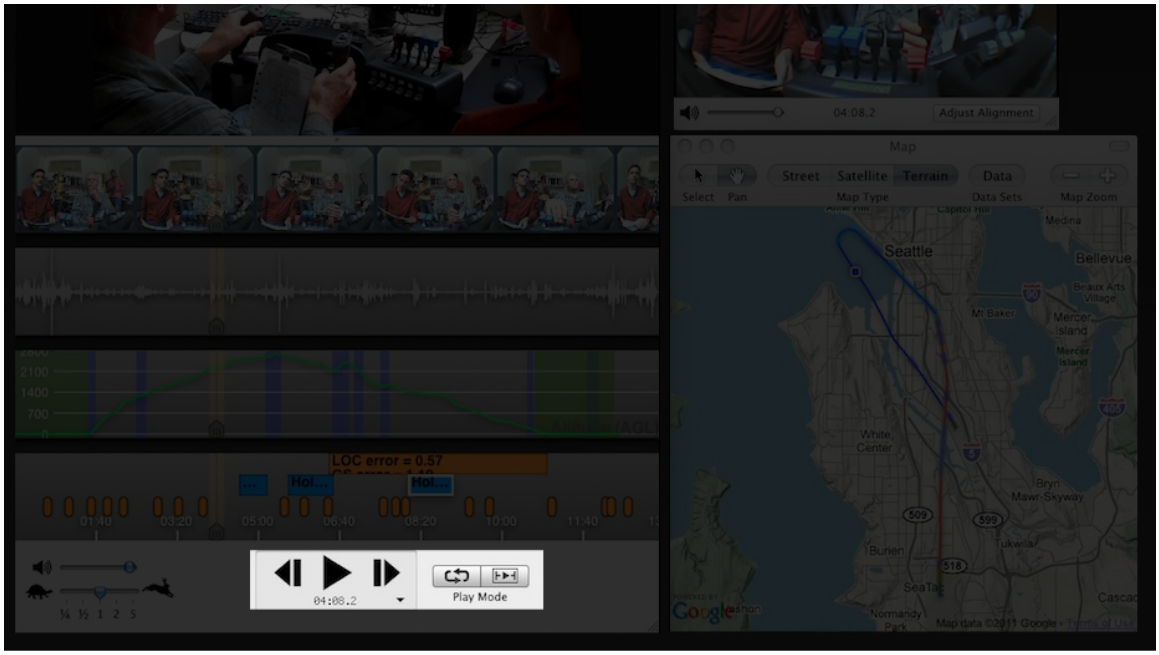

Figure 4.19 – *Media Player in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

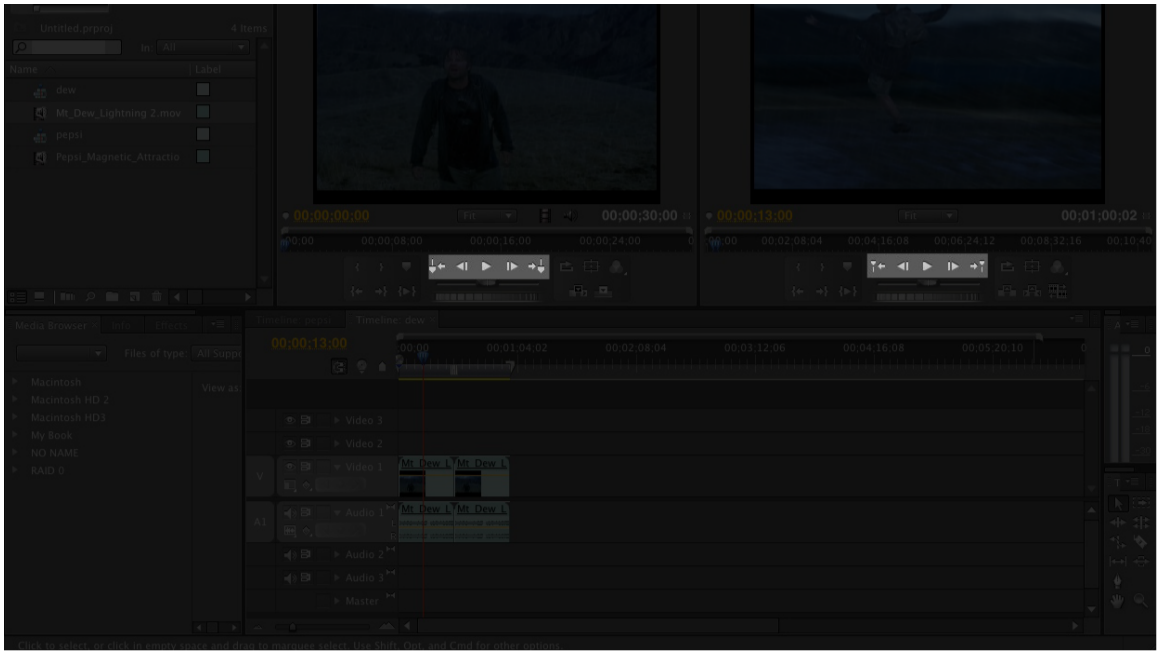

Figure 4.20 *– Media Player in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- Enable the project's playback – This pattern can be used to play back and review the project in standard time. The time-series data is represented with each passing moment.
- Rewind the project's playback – This pattern can be used to reverse the playhead's direction during the project's playback. Instead of moving forward in time, this pattern can be used to change the course of the playback and allow users to review the project backward, or move the playhead to a position earlier in time.

- Fast-forward the project's playback – This pattern can be used to alter the playhead's playback speed. As the users review a project's playback, users can use this pattern to increase or decrease the rate the project is played back.
- Loop a segment of the project – This pattern can be used to edit the playhead's playback sequence. The pattern can be used to loop, or replay a predetermined segment of the project indefinitely.

What does the pattern do?

The playback function represents the time-series data as it occurred with time. The playhead, often used to mark the moment the time-series data is represented during playback, is essential in helping users see the progression of the playback. The *media player* pattern alters the playhead's direction, speed, and behavior for playback purposes.

When should the pattern be implemented into a design?

Use this pattern when users need to review time-series data in the playback fashion. Time-series data can be represented in still form, but because it is indexed with time, the data can also be represented in sequence with time. This pattern helps users control the manner in which the playback occurs.

Why is this particular pattern so important to the user experience?

Time-series data like video can be represented in static form using small multiples of different frames. Audio can be represented statically as waves registered on

the decibel range. These states of representation, however, do not fully represent the nature of the time-series data because they exclude the temporal dimension.

The playback function includes the temporal dimension by representing the time-series data in sequence with time. As users review the time-series during playback, users may need to control the sequence. The *media player* can be used to automate the playback sequence. After enabling the playback function, the data can be represented consistently through a range of time without any need for user intervention.

How should this pattern be implemented into a design?

Interactions with the media player are typically afforded as a button group. One button, often known as the 'play' button, enables the playback function. Another, typically called the 'rewind' button, is used to reverse the direction of the playback. Another, typically called the 'fast-forward' button, alters the speed of the playback in the forward direction. Depending on the needs of the users, the media player may include more elaborate features. A common feature outside the basic play/pause, rewind, and fast-forward, is the 'loop' feature that lets users playback a segment of the project only. Users can mark the segment of the project using special annotation markers on the *timeline*. The playhead begins the playback progression on the first marker and restarts the playback process at this marker when it reaches the second marker. This feature often helps users review a specific segment of the project over and over. This automates the playback of the segment helping users examine the particular segment more closely without manually setting and resetting the playhead.

DESIGN PATTERN: Playhead Ribbon

Protools

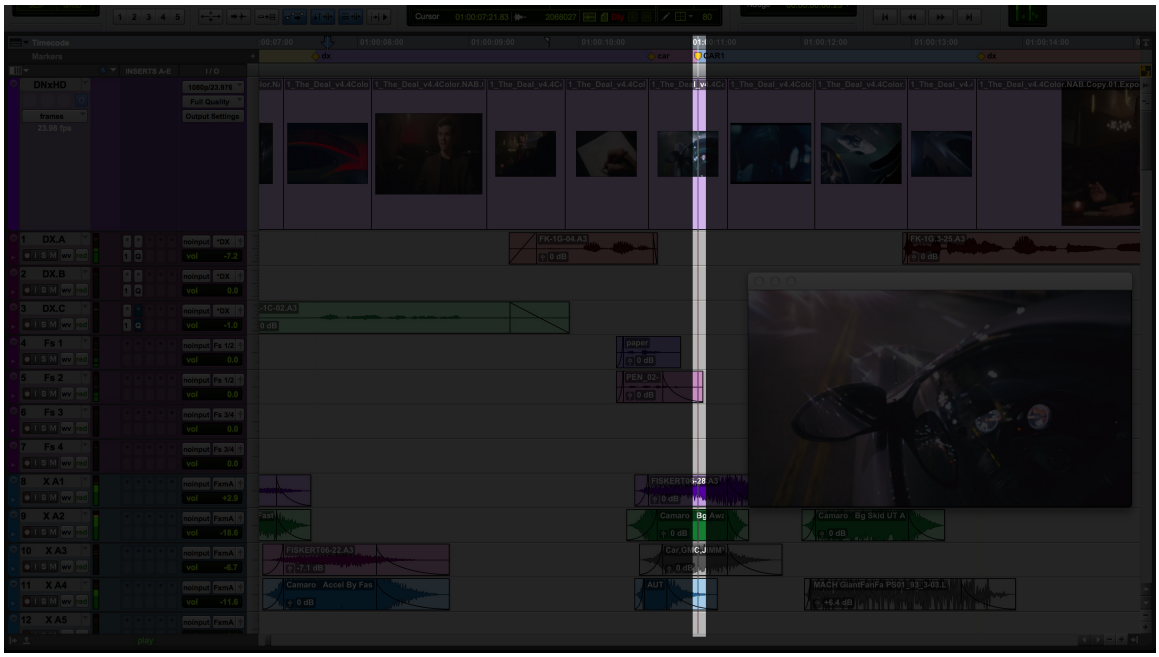

Figure 4.21 *– Playhead Ribbon in Protools* (avid_ProTools11_video.jpg, 2013)

Sportscode

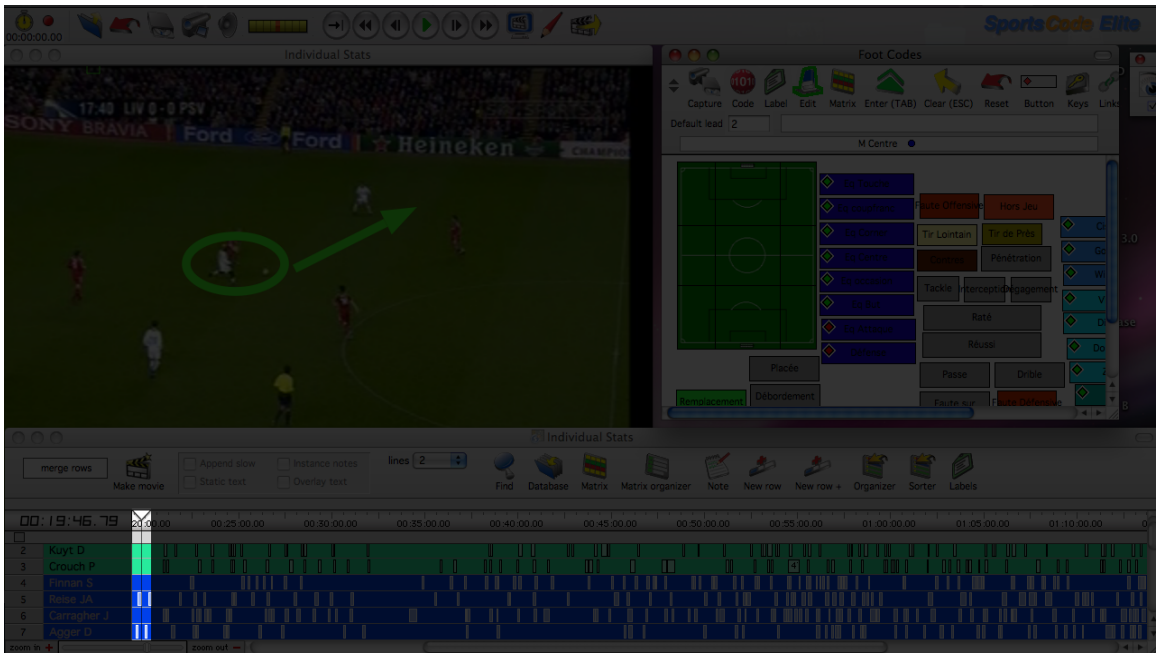

Figure 4.22 *– Playhead Ribbon in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

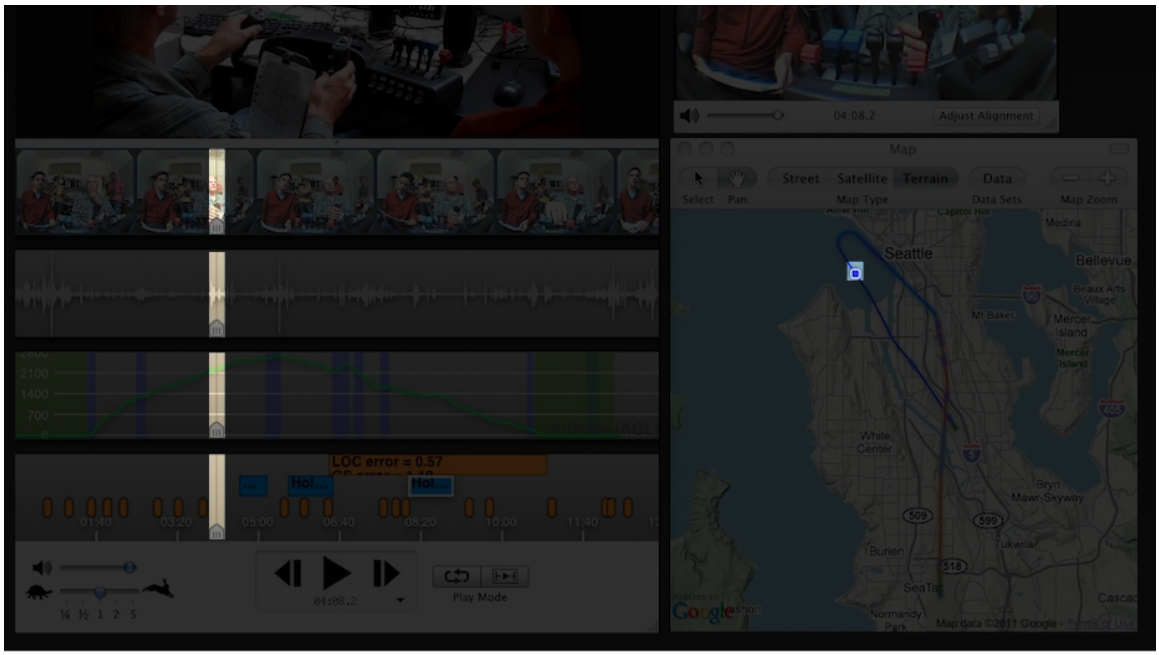

Figure 4.23 *– Playhead Ribbon and GPS Marker in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

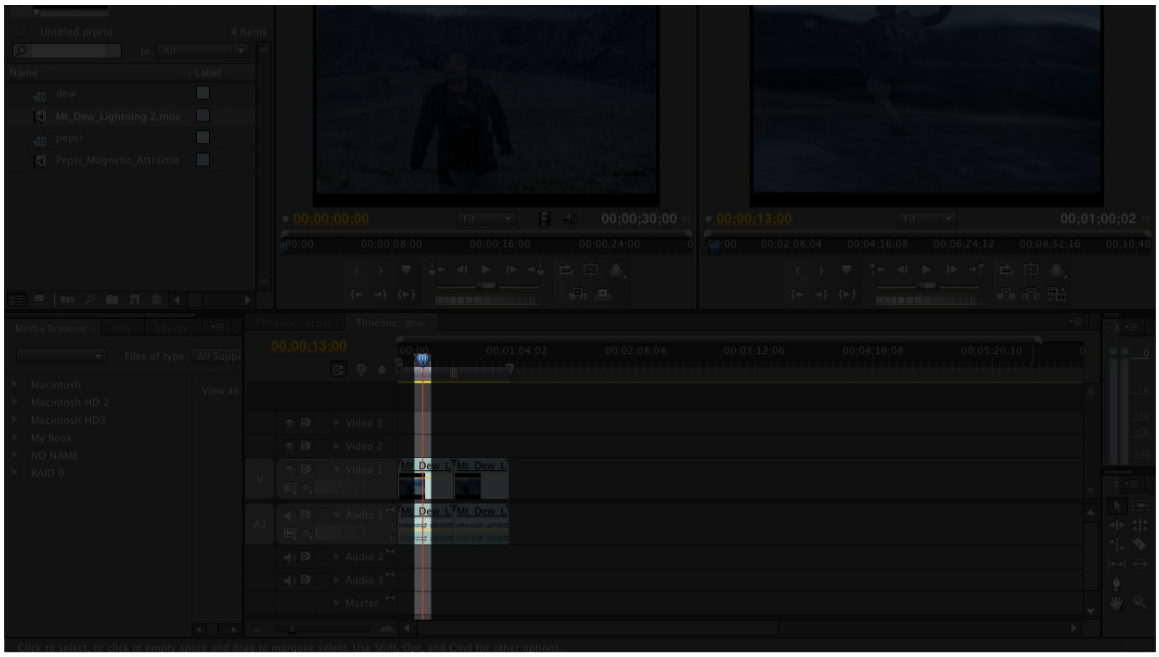

Figure 4.24 *– Playhead Ribbon in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- Position the playback's starting point – Users can drag this marker along the *timeline*. After pressing 'play,' this pattern moves from left to right in respect to time. The time-series data is represented in sequence as this pattern passes along it.
- Judge the speed of playback – Users can use the speed at which this pattern moves along the *timeline* to reference the rate the time-series data is being represented in sequence.
- Reference the *timeline* – Much of the time-series data in the project will be stacked below or above the *timeline.* This causes some time-series data to be located far from the *timeline*. Users can use this pattern to reference the timeline using the pattern's extended ribbon as a guide.

What does the pattern do?

This pattern marks the moment in which the time-series data is represented during playback. It can be dragged along the timeline to different temporal positions. During playback, this pattern moves according to the controls set by the *media player*. In *timeline navigators,* the standard *playhead* found in basic video or audio players is supplemented by a distinct line extending out over the time-series data representations. This characteristic, found in every application analyzed, differentiates this pattern from other standard playheads.

When should the pattern be implemented into a design?

Use this pattern when users are working with time-series data stacked on multiple y-axes. Because users may need to reference the *timeline,* this pattern can be used to accurately reference those time-series data located on the axes furthest from the *timeline.*

Why is this particular pattern so important to the user experience?

The playhead alone marks the exact moment of playback. This can help users understand their time-series data's current representation relative to the project's temporal range. A playhead near the left side of the *timeline* can help users understand how the current playback moment is near the beginning of the project, whereas a playhead on the right side means the playback is almost at an end.

Without this pattern, users would need to reference the numerical values found in the *time display.* This form of representation can be effective for absolute judgments, but users may take longer to understand the playback's current moment without any of the cues found with the *playhead ribbon* and *timeline's* inherently spatial properties.

Extending the playhead with a salient line provides users with two major benefits. They can position the playhead according to a specific event found in the time-series data. They can also reference the timeline to determine a particular event's timestamp. For example, users working in *Protools* may be working with many instruments. Due to the multi-y structure of the stacked *data swords* and the increasing spatial distance between them and the *timeline*, user eye gazes may need to traverse

over a large area of the screen to find a correspondence between an event and the *timeline.* The *playhead ribbon* alleviates this problem by providing a straight line that extends across every *data sword* from the *timeline.*

How should this pattern be implemented into a design?

Simple shapes like a circle or triangle can be used to mark the *playhead's* position on the timeline. Supplement the playhead marker with a visually salient line extending down over each *data sword*. The line typically stops at the *data sword* furthest from the *timeline.* Users can either click the line or marker to drag the entire pattern's position.

DESIGN PATTERN: Temporal Zoom

Protools

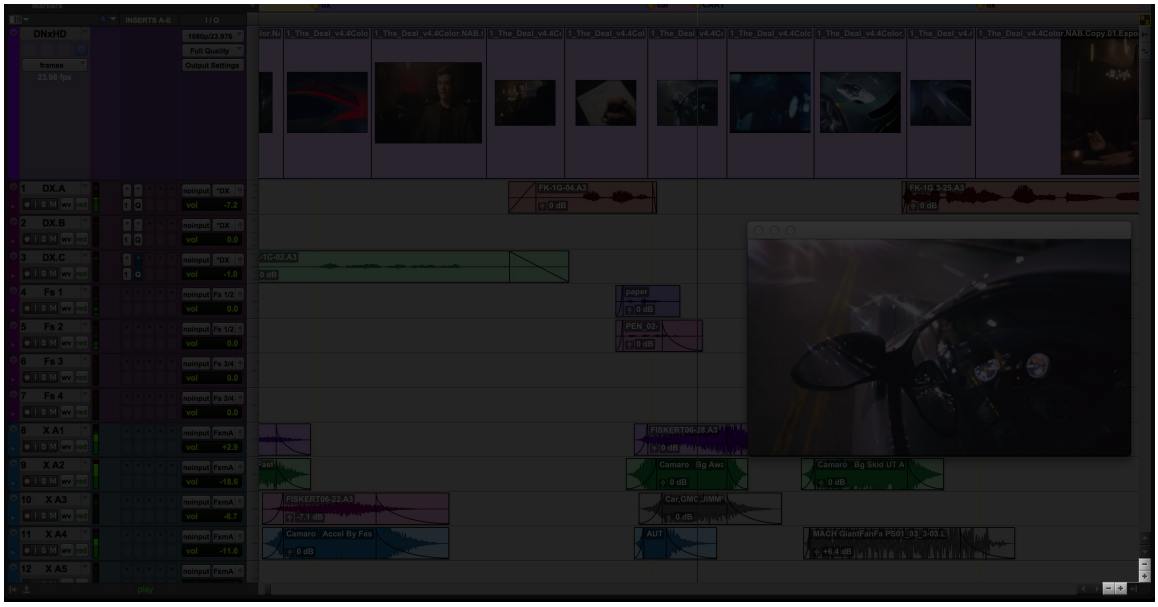

Figure 4.25 *– Temporal Zoom in Protools* (avid_ProTools11_video.jpg, 2013)

Sportscode

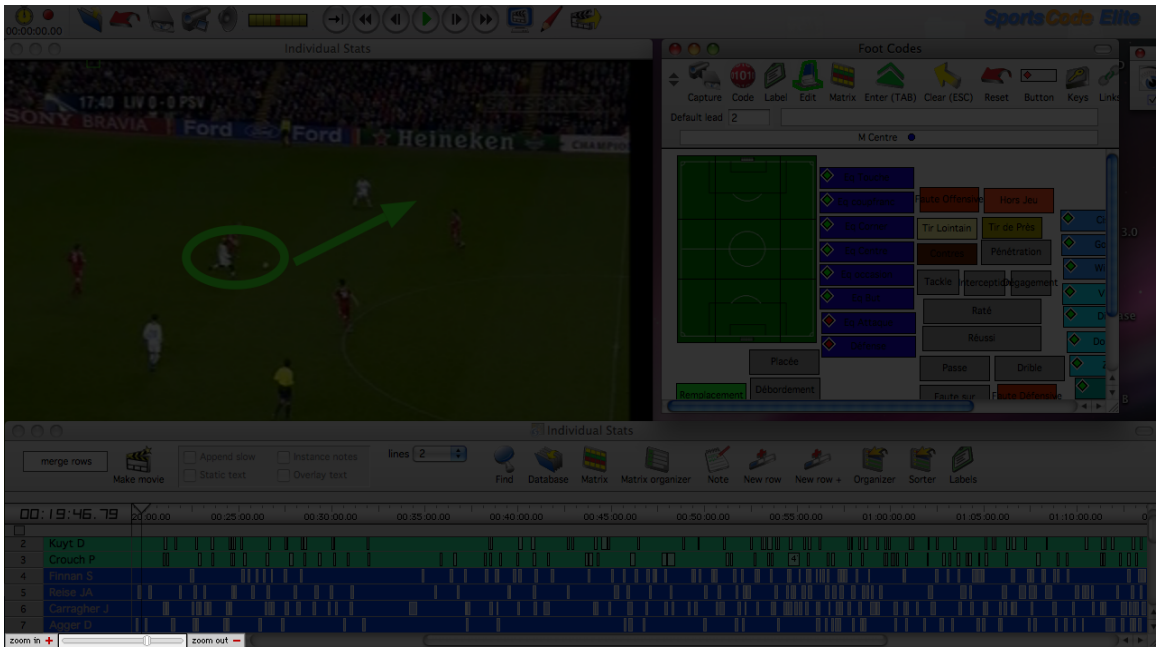

Figure 4.26 *– Temporal Zoom in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

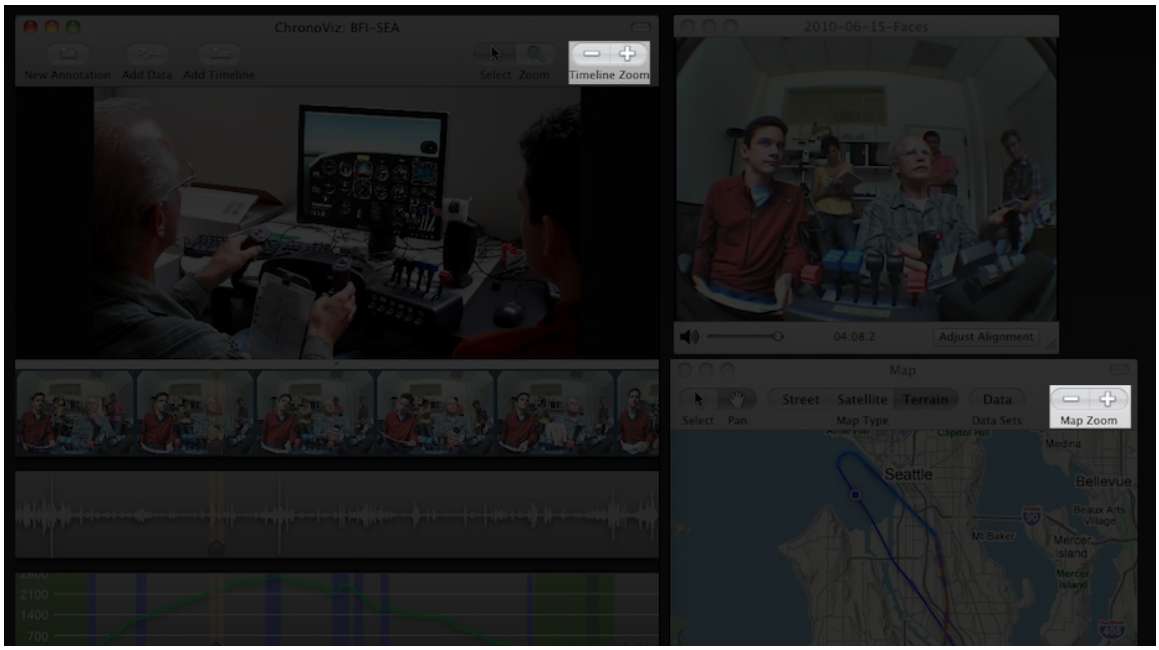

Figure 4.27 *– Temporal Zoom in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

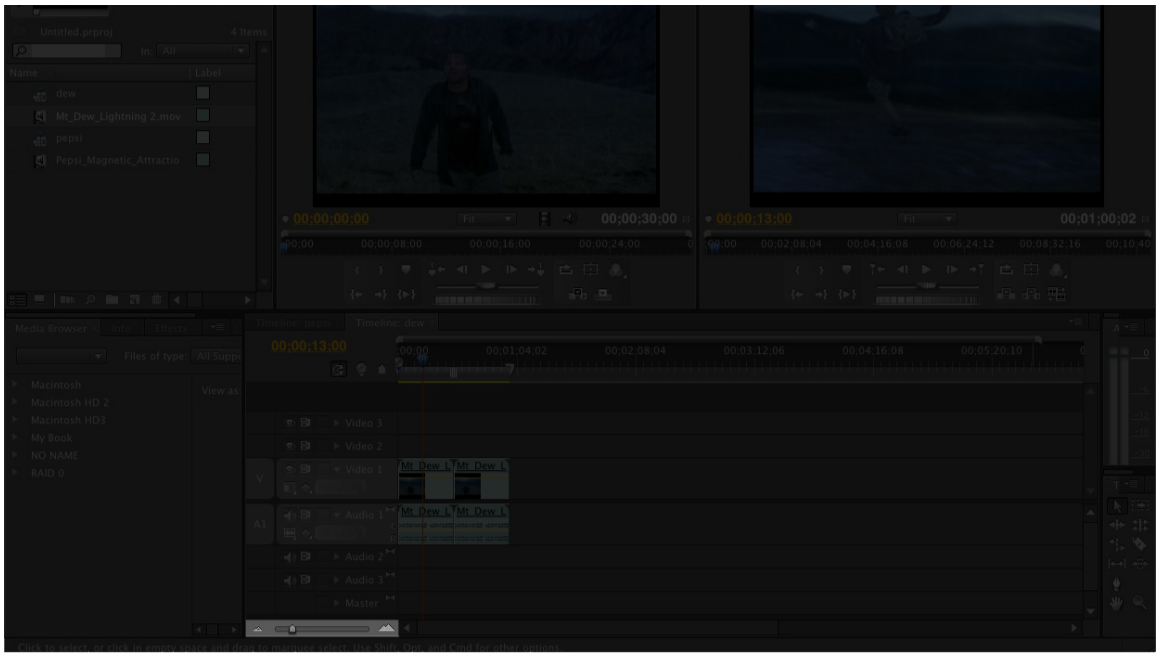

Figure 4.28 *– Temporal Zoom in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- View the time-series data in multiple temporal resolutions – This pattern can be used to adjust the temporal resolution of the time-series data.
- View and analyze the data in macro-resolutions – Users can “zoom out” and see the time-series data visualized in temporal resolutions where hours of data could potentially be represented in less than an inch of space on the screen.
- View and analyze the data in micro-resolutions - Users can “zoom in” and see the time-series data visualized in higher temporal resolutions where seconds of data could potentially be represented using the entire width of the screen.

What does the pattern do?

Like other zoom tools, this pattern includes two buttons with a range of resolutions in between. One button is used to "zoom in" or increase the resolution while the other decreases the resolution in which the figure, image, or data is visually represented. In the case of *timeline navigators,* the special type of zoom function alters the temporal resolution of the time-series data presented in the application. Users can use this pattern to view the entire project's range of time on a small space of the screen. On the other hand, users can also visualize small time-values across a large space of the screen and everything in between.

When should the pattern be implemented into a design?

Use this pattern when users need to see their time-series data visually represented at multiple temporal resolutions.

Why is this particular pattern so important to the user experience?

Multiple resolutions provide users with a greater range of understanding their data (Few, 2006). Macro-resolutions, where users can see all their data, may help users conduct a broad, more relative analysis. Moviemakers using *Premiere* can scan for specific scenes based on the great range of small multiples presented. Sports analysts using *SportsCode* can determine a specific event's occurrence, like scoring, relative to the entire project's range of time. In this macro-perspective, they may be able to determine that a team tends to score more in the second half of each game.

Micro-resolutions, where users can see their data closely, may help users conduct a more precise analysis of their data. Song composers looking to splice segments of audio to the millisecond may need to zoom in on their data using *Protools.* Researchers using *Chronoviz* may need to look at the exact language chosen by one of the commanders during a specific moment in the training scenario.

How should this pattern be implemented into a design?

This pattern does not need very much real estate on the UI. Some applications only provide the positive (increase in resolution) and negative (decrease in resolution) buttons to manipulate the data's resolutions. Others include a slider in between the buttons. This slider can also be used to manipulate the resolution by dragging the slider up and down the range.

Maintaining a consistent scaling effect is extremely important to providing accurate graphical representations of the data. Because all the data is indexed on time, any change in temporal resolution must apply to every source of time-series data presented on the interface. The scaling effect should also be applied to other items relevant to the user's understanding of time; items like the *timeline* and *playhead ribbon.*

DESIGN PATTERN: Cursor Editor

Protools

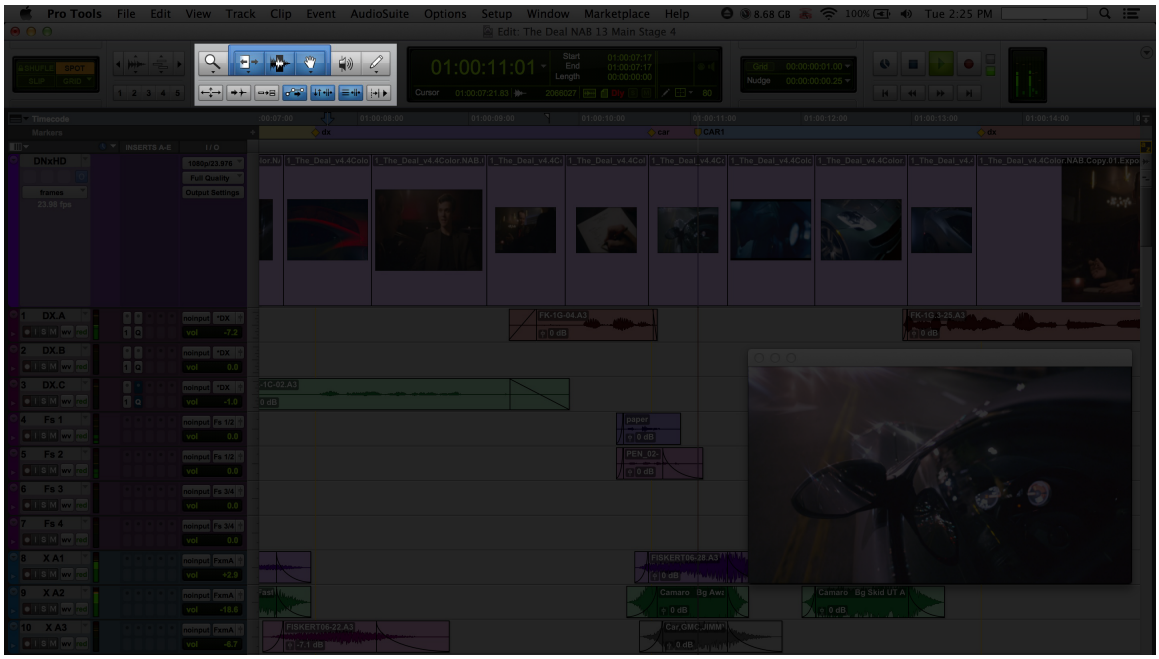

Figure 4.29 *– Cursor Editor in Protools* (avid_ProTools11_video.jpg, 2013)

Sportscode

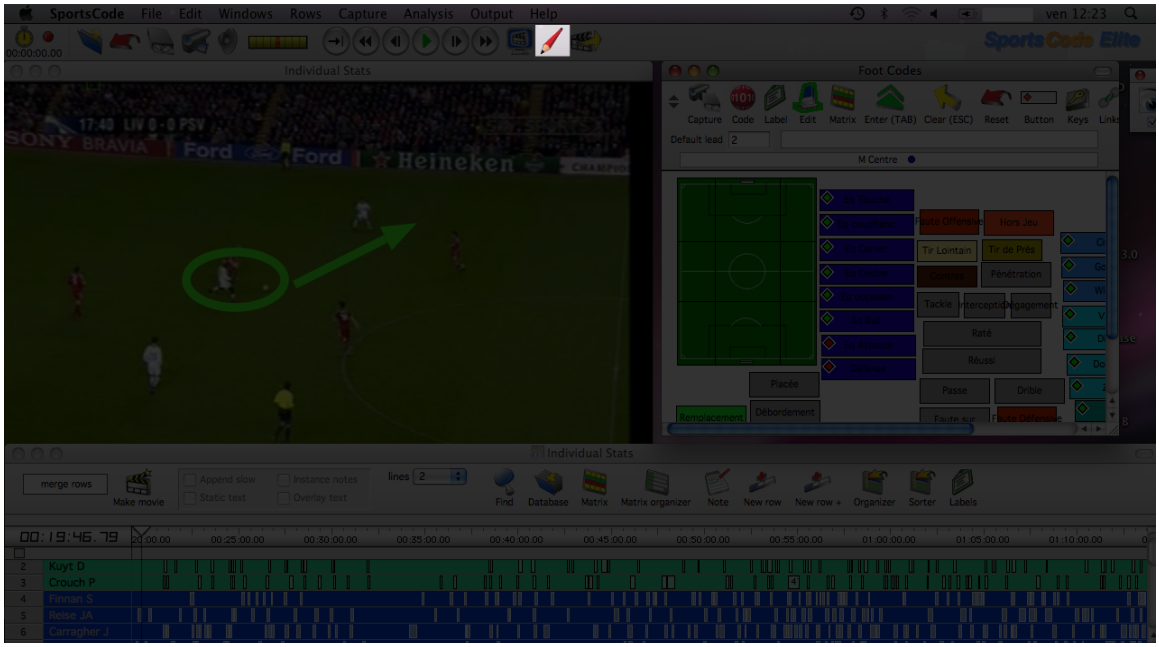

Figure 4.30 *– Cursor Editor in SportsCode* (PPT Sportstec.png, 2012)

ChronoViz

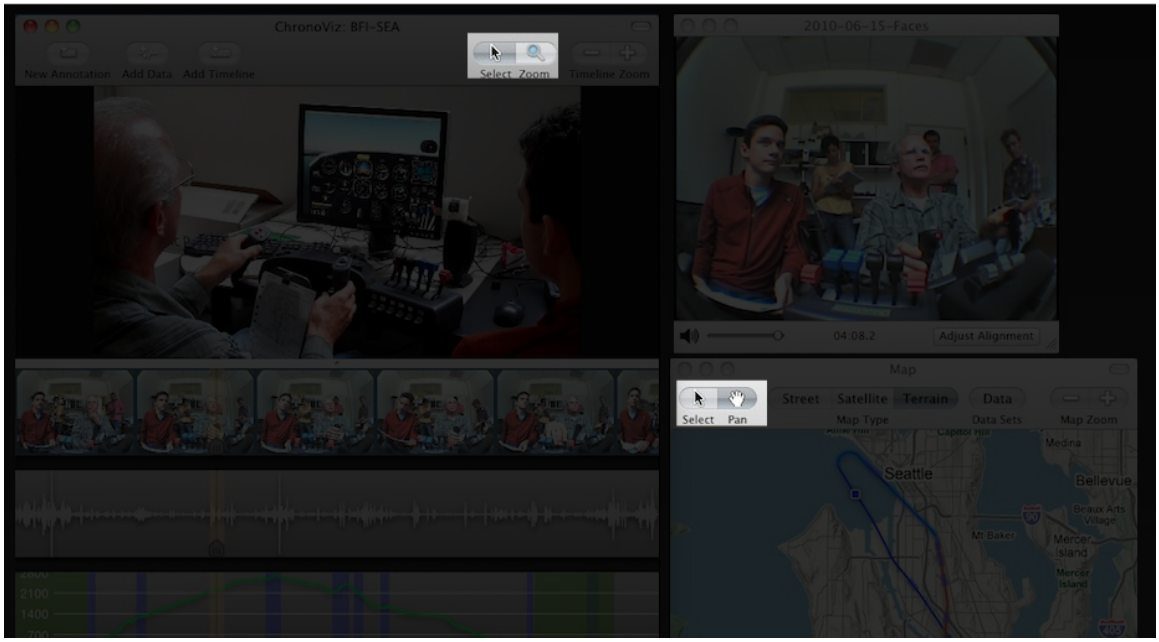

Figure 4.31 *– Cursor Editor in ChronoViz* (ChronoViz-United.jpg, 2012)

Premiere Pro

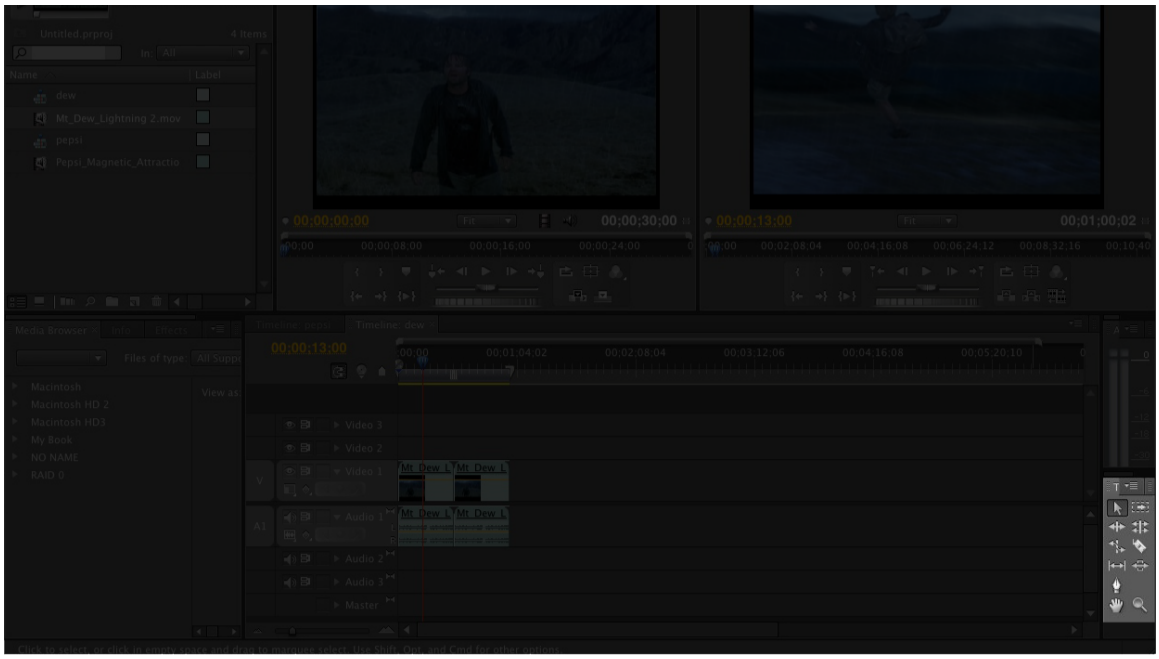

Figure 4.32 *– Cursor Editor in Premiere Pro* (premiu1.jpg, 2008)

TASKS SUPPORTED:

- Alter the mouse cursor's modality – The cursor can be used in different ways. Users can use this pattern to change the cursor's modality to perform different functions.
- Select items – The select function is the cursor's most common purpose. In this mode, users can click on an item and select it for further interactions.
- Zoom on a range of time-series data – In this mode, users can click and drag the cursor across a range of time-series data and increase or decrease that range's temporal resolution.
- Annotate – In this mode, users can click on the timeline to add annotation marks, click and drag on the timeline to define a loop range, or, in some special cases, draw annotations on videos.

What does the pattern do?

This button group is one of the most specialized patterns included in this analysis. Depending on the application, this pattern may contain a wide variety of modes and functions. These modes often depend on the needs specific to the user domain being served. *Sportscode,* while it provides users with the ability to annotate the time-series data, does not include nearly as many functions in its *cursor editor* as those found in *Premiere* and *Protools.*

When should the pattern be implemented into a design?

Use this pattern when users need extensive control over their cursor's modality. If users only need to click and drag an item, or basically perform the operations of the standard pointer, this pattern is not necessary. If users need to zoom, draw, erase, select, shape, scrub, etc., this pattern gives users a central place on the interface for selecting a cursor mode.

Why is this particular pattern so important to the user experience?

Due to the many functions needed to interact with time-series data, users need a way to control these functions in a centralized area. By grouping the various selectable modes, users can quickly change their cursor's function and continue with their next task.

How should this pattern be implemented into a design?

Place the most needed functions in one button group. This pattern does not need to take up a lot of space. The icons need to resemble their function accurately. The zoom function is commonly represented by a magnifying glass. In regard to this function, applications typically have users click and drag to zoom in, but pairing the same operation (click and drag) with another key zooms out. The ubiquitous mouse pointer or the gloved hand icon commonly represents the select function. A pen or pencil icon can be used to make annotations.

Provide each mode with a default shortcut key. Users may need to quickly change between cursor modes and the shortcut key dramatically decreases the time spent making these changes.

CHAPTER V

DISCUSSION

Properties found in the Elements

Approaching from the *Surface* layer of the UX, the analysis gleaned rich information from each interface. In addition to the screenshots, the accompanying instructional material in the form of paper documentation allowed the analysis to gain a more deep and insightful understanding concerning each application's user experience. Unfortunately, the analysis found some elements inaccessible or beyond the generalizations of a software genre definition.

The *Strategy* layer could not be accessed by the analysis because the element, by Garrett's definition, contains information available only to those intimately involved in the product's design process (Garrett, 2010). In regard to the *business objectives* side of the *Strategy* plane, one may consider how any business making a software product often has an insurmountable agenda. To define the element according to *business objectives* might have had the thesis chasing executives down for more undisclosed information. To define the element in regard to *user needs*, the analysis would have needed access to research conducted in the formative stages of each product's design process. This information often includes confidential user data. Empirically, the analysis was unable to draw any major similarities from the *Strategy* layer of each product.

The following element was accessible, but the nature of the element prevented the analysis from recognizing similarities between applications. Each application's

Surface layer played a major role in accessing deeper layers, but when the focus rested on the one layer constantly accessible to the five human senses, the element's definition forced the analysis to focus mostly on very unique *visual designs*. The visual design discipline shapes the user's experience by determining aspects like typography, color palette, and animations (Garrett, 2010). All applications analyzed utilized distinct color palettes, typographical applications, animations, and micro-interactions. The analysis could not draw general similarities from the *Surface* layer because the analysis found unique visual experiences on each layer analyzed.

With the *Strategy* and *Surface* elements absent from the analysis, the following discussion focuses on the similarities found in Garrett's three most central elements. In combining a visual analysis of four static desktop computer interfaces and accompanying instructional material, the following discussion sheds light on each application's *Scope*, *Structure*, and *Skeleton.*

To define the *timeline navigator Scope,* the discussion considers commonly fulfilled functional requirements. The analysis from Chapter 4 describes these requirements fulfilled as tasks supported. To define the genre *Structure,* the discussion considers how users may accomplish said tasks interacting with each interface. To accurately assess at least a singular way users may accomplish said tasks, the discussion references accompanying instructional material. To define the genre *Skeleton,* the discussion considers how aspects of the *interface, navigation,* and *information design* disciplines may be found in visually salient design patterns.

Similarities by Function

The analysis provided a list of tasks commonly supported by the applications analyzed. These listed tasks inform on the *Scope's functional requirements*. This discussion must note, however, that the results did not extensively list all the tasks supported by each application because each application, in serving its own user domain, supports far many more tasks than those shared by all applications. The number of *functional requirements* fulfilled by any one application exceeds the number of common *functional requirements* fulfilled by all four. The analysis only focuses on the tasks commonly supported by all four applications.

With this limitation in mind, the analysis still uncovered over 25 tasks supported by all four applications. In some form of language, the instructional material accompanying each application described the listed tasks and outlined at least one method for executing a singular task. While most of the tasks supported by *Premiere*, for example, were not supported by the other three, the analysis was still able to draw a fair amount of similarities between *Premiere* and other applications at the *Scope* layer. To cover the most basic and relevant *functional requirements*, those designing, developing, or evaluating an application within the *timeline navigator* genre may want to take special note of the tasks outlined. Because all four applications, in helping users capture, review, and manipulate time-series data, supported the listed tasks, a strong argument can be made for following their lead.

The *Structure* element of any user experience determines the user's steps through the product's content and functions. Because the structural plane focuses on

determining sequences in the user experience, disciplines like *interaction design* and *information architecture* play a vital role in making consequences to user interactions within the product consistent and predictable (Hogue, 2011). Designers often form these user steps using *scenarios,* hypothetical, goal-oriented task executions (A. Cooper, Reimann, & Cronin, 2007b). Scenarios help determine how the UI needs to behave in response to a user working toward accomplishing a task.

The products analyzed in this thesis serve very different user domains and were likely formed using different scenarios. Different teams working on the *Structure* plane of their respective *timeline navigator* may have used any number of screen sequences to help users accomplish a task. Finding similarities between two user experiences' interaction design can be a challenge when designers forming the interactions have so many options. To add new data sources, for example, one product may have the user click through multiple drop down-menus to import a file, like *SportsCode.* Other products, like *Protools* may only require the user to plug in the data-gathering device. These applications differ in their interaction design because they guide users down a sequence of screens very differently.

Interestingly enough, while many of the applications analyzed facilitate their respective tasks and scenarios uniquely, they still manage to use similar design patterns. The task of making an annotation, for example, requires *Premiere* users to drag the *playhead ribbon* to the desirable position on the *timeline* and press the default shortcut key 'M.' *SportsCode* users, to accomplish the similar task of adding an annotation, must

click on customizable buttons in a special panel during playback. Ultimately, both interactions manifest as *annotation* marks near the *timeline.*

In this case, the applications do not differ in *what* tasks are facilitated; both support the user's need to make annotations to their time-series data. The applications do not differ in *where* the tasks are facilitated; both interactions provide feedback as marks on the *timeline*. The most significant difference is observed in *how* the applications facilitate the task; one requires a shortcut key while the other requires a special control panel. Similar design patterns are implemented as different instances, but the instances, in their unique manifestation, still help users accomplish the same tasks.

The connection between *Structure* and *Skeleton* is interesting because a lot of the work developed on each plane often overlaps. Chapter one mentions how *information architects* may list out ways to form an application's *navigation design.* In other cases, *interaction designers* may be the ones forming UI layouts and compositions. Both disciplines have the capacity to structure an application's functions into perceivable, understandable human-computer interactions. It is not uncommon to see an individual product team member work on multiple planes.

This large overlap between the *Structure* and *Skeleton* elements may be connected by something Dan Saffer refers to as *microinteractions (Saffer, 2013)*. *Microinteractions* are "contained product moments that revolve around a single use case." The most effective *microinteractions* are composed of four main stages: 1) trigger, 2) rules, 3) feedback, and 4) loops and modes. Design patterns, because they are

dynamic and interactive, contain one or more *microinteractions*. The *playhead ribbon*, for example, uses the slider *microinteraction* also found in the *temporal zoom* to let users adjust a marker along a gradient line. Another example can be found in the color shifts observed when an interface indicates a button press. When clicked, virtual buttons provide feedback of the user click.

These moments technically occur on two of Garrett's elements. *Microinteractions* help support common tasks along a broader sequence of the user experience using familiar locations and patterns in the UI.

Similarities by Content

In serving different user domains, the *timeline navigators* analyzed presented dramatically diverse sets of content. *Protools* allows users to interact with time-series data intent for music production. Users working with *Protools* often use analog instruments, midi instruments, and microphones as their primary sources of data. The goals supported in *Premiere* primarily focus on video production. Users working with *Premiere* often import videos from prerecorded video files or capture video or audio from cameras and microphones plugged into the computer. Of the applications analyzed, *ChronoViz* supports the greatest variety of time-series data. In addition to audio and video, a *ChronoViz* user can import .cvs files. In and of themselves, .cvs files can present an incredibly wide variety of time-series data, like GPS data, financial data, weather data, etc. *SportsCode* users can capture or import video and audio from data gathering devices like video cameras or microphones.

Like the *functional requirements* section, the *content requirements* satisfied by all the applications are exceeded by *content requirements* satisfied by each individual application. While the applications presented diverse content, they tended to present the content in very similar ways. Users editing in *Premiere* could be working with birthday party clips and users analyzing data in *Chronoviz* could be working with videos captured in a laboratory, but both applications use *small multiples* to represent the time-series data along the *timeline.* During playback, this video content is often displayed in its own panel.

Timeline navigators visually represent other forms of time-series data like audio, too. This data type can take form as a line graph, bar plot, pie chart, and more. In every product considered by the analysis, audio data is displayed using the area chart graphic with the audio wave's ebbs and flows fluctuating in respect to the decibels registered.

While the *timeline navigators* selected by the analysis only share audio and video as common types of content, these two examples alone show how some of these products may visualize time-series data. Their respective *data swords*, harnessing the power of the *multi-y graph,* can visualize very diverse sources of information. *Protools,* for example, visualizes midi data using a very distinct stroking method in which hits from the midi instrument (typically a keyboard) are represented as horizontal bars indexed on the time the instrument key is hit and released. This data's y-values are determined by the key's position on the midi instrument with low notes typically represented at the bottom and higher notes represented at the top of the y-axis. *ChronoViz,* in visualizing GPS data, adopts a method similar to that of video presentation. During playback, a top

view perspective of a map is typically used as the background for the tracked unit(s) to move along according to their geospatial coordinates. Much like video content, this can be visualized on the *timeline* using *small multiples* to represent the sequence of frames captured in the GPS data.

These methods visualize multiple dimensions inherent to time-series data on a two-dimensional plane (the computer screen). This can be a difficult challenge as Edward R. Tufte notes in his book, *Envisioning Information (1990)*:

> *Nearly every escape from flatland demands extensive compromise, trading off one virtue against another; the literature consists of partial, arbitrary and particularistic solutions; and neither clever idiosyncratic nor conventionally adopted designs solve the inherent general difficulties of dimensional compression.*

In other words, the information as it exists in multiple dimensions, cannot be visually represented on a flat, two-dimensional surface without compromising the information's original, natural state. *Timeline navigators,* in attempting to visualize the three or more dimensions inherent to time-series data, face an intriguing and lofty challenge. Take the following example as testament:

> *Researchers conducting an A/B test with a new navigation system may want to look at driving routes taken by their participants in a*

study and compare them to routes taken by participants using another navigation system in the control group.

In the data gathered, researchers can expect at least three dimensions of information. The first is time. Participants will be driving from one point to another in time. As noted by the analysis, *timeline navigators* consistently reserve the x-axis for representing this dimension. This dimension helps researchers understand *when* participants demonstrate certain behaviors.

Timeline navigators typically reserve the y-axis for a dimension most often relevant to the data source in question. If the researchers are potentially concerned with data from the participant's heart monitor, for example, they can expect a line graph dictated by the parameters time (x-axis) and heartbeat velocity (y-axis). If the researchers are potentially concerned with data from the microphones inside the vehicle, they can expect an area chart dictated by the parameters time (x-axis) and decibels (y-axis).

If the researchers are potentially concerned with data from the vehicle's GPS monitor, the second most relevant dimension is space. To understand *where* their participants drove, researchers need appropriately visualized representations of time, space, and the data source. Sometimes these visualizations, to avoid the compromises that come with what Tufte refers to as *dimensional compression*, must go beyond the standard x and y axes plots. To support this need*, timeline navigators* often utilize

modal panels, small multiples, and/or *GPS balloon markers* to preserve the data source's multiple dimensions.

While the two first dimensions of information commonly visualized by *timeline navigators* are time and space, the third dimension can often be the data source itself. The two dimensions of time and space do not effectively represent data from multiple participants so a third must be considered. To accurately understand temporal and spatial data from multiple sources, researchers need a third dimension that preserves the distinction in each source of participant data. Visualizing multiple data sources, all indexed on the same timeline, often requires the temporally consistent *multi-y graph* design pattern. To allow for interaction and manipulation, *timeline navigators* typically use multiple *data swords* to preserve the two or more dimensions belonging to each data source.

In some cases, researchers may need access to more than three dimensions to further organize and structure their data. Oftentimes, the fourth dimension preserves the distinction between data types. The researchers, for example, may need to categorize their participant data according to experimental groups. The researchers looking to compare categories of time-series data introduce a fourth dimension to the information architecture by pivoting their analysis on data source types (experimental/control). A fifth dimension could easily be introduced by including annotations.

Timeline navigators represent time-series data using the *matrix* information architecture. According to Garrett, the matrix structure helps users navigate their

content by pivoting their navigation to one axis of the matrix (Garrett, 2010). If users want to shop for t-shirts on an e-commerce site, they may search by color, size, or price. *Timeline navigators* provide this same flexible way of consuming content using different axes to focus their navigation.

Consider how the following *timeline navigators* may help users pivot their navigation in respect to time. *Protools* users may need to edit their production and interpret their notes, sounds, and effects at the five-minute point. *Premiere* users may need to insert new video content near the beginning of the movie. *ChronoViz* users may want to analyze behavioral data at the point in which the independent variable was introduced to participants. *SportsCode* may need to assess athlete performance in the final minutes of the sporting event.

Consider how the following *timeline navigators* may help users pivot their navigation in respect to a second dimension relevant to the data source. *Protools* users may need to adjust the output levels of the entire product. In this case, the second dimension is characterized by decibel output. *Premiere* users may need to control the lighting from one of the scenes; the second dimension being lighting effects. *ChronoViz* users may need to look at participant galvanic skin responses immediately after introducing the experiment's independent variable. In this case, electrical conductance is reserved to the second dimension. *SportsCode* users may need to look at error frequencies in athlete performance. The second dimension could be a metric like the number of turnovers committed during the span of a basketball game.

In respect to the third dimension (often the data source itself), *Protools* users may only adjust the panning effect attached to the lead guitar track. *Premiere* users may want introduce a narration audio track while leaving the video footage unmodified. *ChronoViz* users may need to interpret data belonging to a singular participant. *SportsCode* users may want to assess the performance of a singular athlete. *SportsCode* actually features a function called the "Matrix" which allows users to view different types of annotations made for each athlete. A team of soccer players may commit various types of behavior like goal scoring, passing, assisting goals, etc. Users mark each of these events and *SportsCode's "*Matrix*"* displays them for trainers to evaluate.

Defining the Genre

Judging from the shared properties discovered in each application's *Scope, Structure, and Skeleton,* this thesis proposes a classification system for defining the software genre: *timeline navigator*. Figure 5.01 attempts to visualize the software genre classification system.

This figure represents four individual user experiences (A, B, C, and D) in relation to one another.

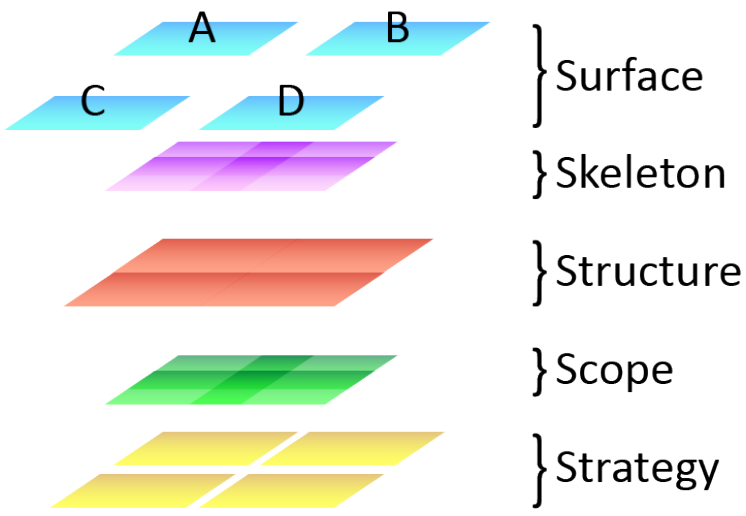

Figure 5.01 *– Classification System*

At the bottom, four *Strategies*, each represented by yellow quadrilaterals, occupy a conceptual space based on one central characteristic: help users interact with time-series data. This is a general characteristic determined by the author of this thesis. It does not reflect the *business objectives* or *user needs* determined and discovered by each application's product design team. Instead, this central characteristic provides the classification system with an excluding property that eliminates other applications from being considered in the software genre. Prieto-Diaz and Freeman (1987) states, "members of a group produced by classification share at least one characteristic that members of other classes do not." The analysis was unable to determine any classifiers from this layer so the *timeline navigator* genre's *Strategy* element may or may not accomplish the same *user needs* or *business objectives*. Until further analyses can be conducted, this excluding classifier unites the classification system proposed in thesis. Notice the close, but differentiating proximity between the yellow *Strategy* planes in the figure.

The following planes in green represent the conceptual space shared by the four applications in the *Scope* element. All four applications satisfied a vast amount of *functional* and *content requirements.* Despite their unique *Scopes,* the analysis uncovered similar properties in the form of tasks supported and content presented. A remarkable amount of overlap between the green planes in the figure represent the overlap in *functional* and *content requirements* satisfied by all four applications.

The red planes represent the conceptual space shared by the four applications in the *Structure* element. While the four applications supported many of their tasks with

unique interaction designs, they all managed to structure their information using the matrix model of information architecture. Of the commonalities discovered by the analysis, this particular quality, above all others, achieves the absolute 100% adoption rate. There are four possible ways one can structure information: sequential, hierarchical, organic, and matrix (Garrett, 2010). The applications analyzed all clearly adopt the matrix information architecture. Due to a great imbalance stemming from different interaction designs and identical information architectures, the figure shows no overlap or gaps between the applications. The *timeline navigator* software genre, in its *Structure* element, poses the most curious questions yet. More on this topic is discussed down further below in the discussion.

The remarkable quantity of shared properties discovered in each application's *Skeleton* almost mirror the quantity found in the individual *Scopes.* According to the analysis, each application shares eight distinct design patterns. These patterns provided each application with the necessary interface to facilitate the *functional* and *content requirements* set forth by each application's *Scope.* While there were significant differences in the way each application combined the patterns to guide users through various tasks, the various instances of these design patterns tended to play similarly consistent roles. In defining the *timeline navigator* genre, the purple shape in the figure above represents the notable coverage shared by each application's *Skeleton.*

Due to the very unique visual designs found on each application's *Surface,* the analysis was unable to include a significant amount of shared properties in the genre definition's *Surface.*

With these shared properties in mind, contemplate the following implications. Users form mental models over time (Gentner & Stevens, 1981). They become accustomed to reused stimuli presented on interfaces in the form of design patterns. They see a *media player* and they instantly know what to do with it because they have seen one before. These stimuli, from the user's end of the dialogue, are always accessed through the *Surface* first*,* but they can be sourced, as the analysis shows, deep down in the *Scope* or *Structure* layers*.* Affordances, interactions, microinteractions, content, patterns, data-visualization methods, all inherent to the product's conceptual model *surfaces* in the most accessible layer of all and if things go well users will recognize them and find them familiar (Jennifer Tidwell, 2011). If product teams wish to satisfy their users' needs first and foremost, their product needs to match their users' mental models. Fortunately, product teams designing and developing a new product can reference other products for conventional solutions; solutions users have long since adopted as the most familiar and intuitive.

In 1977, Christopher Alexander first introduced the concept of design patterns to the discipline of architecture (Ishikawa & Silverstein, 1977). Soon after, software engineers adopted a similar practice after learning the benefit of cataloguing and classifying solutions that targeted known problems (Borchers, 2001). Initially, these early adopters focused on communicating design solutions between product teams. A common "pattern language" helped teams struggling to maintain communication with different ways of verbally encoding the same information. Two engineers, for example, could be talking about the same concept, but communication between them would

often break down due to them using completely different words to refer to the same concept. “Pattern languages” fixed that.

Design pattern advocates did not move beyond the sole communication purposes of a “pattern language” until the “Gang of Four” published their pivotal book in 1995 (Jenifer Tidwell, 1999). This book introduced the concept of adopting solutions to problems someone had already solved (Freeman, Robson, Bates, & Sierra, 2004). The book documented flexible templates of code other software engineers could adopt and save time from reinventing the wheel.

This thesis follows the same principle advocated by the “Gang of Four” and since then, Jenifer Tidwell, who, in 1999, noted on the HCI discipline’s “badly needed benefits of such a pattern language” (Jenifer Tidwell, 1999). The applications themselves have been around for a while, but their classification into one genre of software has gone largely unrecognized until now. This thesis both recognizes the genre’s contribution to effective interaction with time-series data and classifies the genre according to properties discovered in industry-leading software. To help designers meet their users’ mental models, the software definition lists and describes the observed design conventions following the pattern language format.

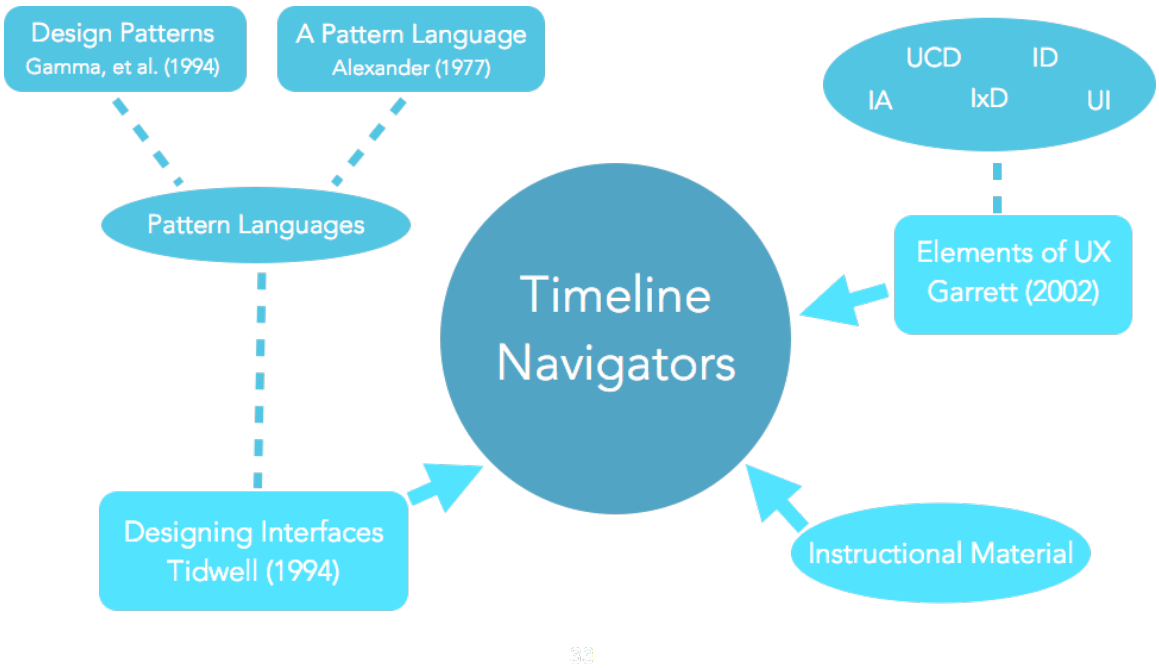

Figure 5.02 – How to Define a Software Genre

Limitations

Any product team set on developing a software application for the purposes of helping users capture, review, and manipulate time-series data can benefit from the genre definition proposed in this thesis, but they may want to supplement their process with other techniques if they want to produce a complete and coherent user experience. Using pattern languages may help designers identify or communicate conventions and best practices, but alone, they may not be enough to form entire user experiences. Great user experiences can be designed and developed using many different methods and philosophies. The research process alone includes dozens of techniques for discovering user needs. Product teams would be wise to keep proven UCD methods as part of their product design process and consider pattern languages and the defined *timeline navigator* genre as formative, supplemental techniques.

Future Work

The topic presented in this thesis could continue in one of two major directions. Since the analysis, discussion, and classification system described in this thesis does not include any user data, the claims are for the most part, unverified. To validate some of the claims made in this body of work, researchers could more closely investigate how users interact with time-series data using the software listed in this thesis. Research facilitated by eye-tracking and click-tracking tools could potentially highlight which design patterns users select for certain tasks. Researchers could, for example, verify if participants consistently reference the *timeline* to adjust segments of time-series data. Researchers could also clarify best practices for implementing each pattern. Depending on varying user contexts, the *media player* may need a play/pause button higher on the visual hierarchy, a loop button, or a special annotation button.

The topic of pattern languages and genre classification could also potentially be applied to other types of software. Emerging technologies like immersive gaming, online classrooms, and collaborative systems pose great and novel challenges for product teams and their users. Researchers could investigate these technologies and adopt the methodology presented in this thesis to identify shared properties across the UX elements. In defining the new genres, product teams designing these genres could benefit from identified conventions and practices. In either case, further research, operating with the methodology and analysis presented in this thesis, may lead product teams to designing better user experiences.

REFERENCES

Albert, W., & Tullis, T. (2013). *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Elsevier Science.

Allen, G., & Smith, R. (1994). After Action Review in Military Training Simulations. *Proceedings of the 1994 Winter Simulation Conference*, 845–849.

Alluisi, E. A. (1991). The development of technology for collective training: SIMNET, a case history. *Human Factors*, (33), 343–362.

Amanda Morrow. (2013, February 16). *The Fundamentals of User Experience*. Retrieved from http://www.youtube.com/watch?v=O8zmUJqxrng&feature=youtube_gdata_player

Bachmann, C., Bischoff, H., Bröer, M., Kaboth, C., Mingers, I., Pfeifer, S., & Schütte, B. (2012). *Operation Manual: Cubase 7, Cubase Artist 7*. Hamburg, Germany: Steinberg Media Technologies.

Bailey, B. P., Konstan, J. A., & Carlis, J. V. (2001). The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Proceedings of INTERACT* (Vol. 1, pp. 593–601).

Bohemia Interactive Australia Pty Ltd. (2012). *White Paper: VBS2*.

Borchers, J. O. (2001). A pattern approach to interaction design. *AI & SOCIETY*, *15*(4), 359–376.

Bosley, J. ., Onoszko, P. W. J., & Sevilla, E. R., Jr. (1979). *The role of the after action review leader in REALTRAIN: Research on training needs*. McLean, VA: Human Sciences Research.

Bosley, J. J., Onoszko, P. W. J., Knerr, C. S., & Sulzen, R. H. (1979). *Tactical engagement simulation training techniques: Two training programs for the conduct of after action review*. Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences.

Cooper, A., Reimann, R., & Cronin, D. (2007a). *About Face 3: The Essentials of Interaction Design*. New York: John Wiley & Sons, Inc.

Cooper, A., Reimann, R., & Cronin, D. (2007b). *About Face 3: The Essentials of Interaction Design*. New York, NY, USA: John Wiley & Sons, Inc.

Cooper, R. G. (2008). Perspective: The Stage-Gate® Idea-to-Launch Process—Update, What's New, and NexGen Systems*. *Journal of Product Innovation Management*, *25*(3), 213–232.

Crosby, O. (2000). Usability engineer. *Occupational Outlook Quarterly*, *44*(4), 48.

Davenport, T. H., & Prusak, L. (1997). *Working knowledge: How organizations manage what they know*. Cambridge, MA: Harvard Business School Press.

Eppinger, S. D., & Ulrich, K. (1995). *Product design and development* (1st ed.). New York, NY: Mcgraw-Hill College.

Eskow, G. (2001). The language of digital audio workstations. *Broadcast Engineering*, *43*(10), 67–71.

Few, S. (2006). *Information Dashboard Design: The Effective Visual Communication of Data*. Sebastopol, CA: O'Reilly Media, Incorporated.

Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). *Head first design patterns*. O'Reilly Media, Inc.

Garrett, J. J. (2010). *Elements of User Experience,The: User-Centered Design for the Web and Beyond*. Berkeley, CA: Pearson Education.

Gentner, D., & Stevens, A. L. (1981). *Mental Models*. Mahwah, NJ: Lawrence Erlbaum Associates.

Gibson, J. J. (1977). The theory of affordances. *Hilldale, USA*, 56–64.

Gothelf, J., & Seiden, J. (2013). *Lean UX: Applying Lean Principles to Improve User Experience*. O'Reilly Media, Incorporated.

Gubler, J. C. (1997). *Unit simulation training system after action reviews (AAR): A novel approach to achieve effectiveness* (Unpublished master's thesis). University of Central Florida, Orlando, FL.

Hawes, L. C. (1972). Development and application of an interview coding system. *Communication Studies*, *23*(2), 92–99.

Hay, S. (2013). *Responsive Design Workflow*. San Francisco, CA: New Riders.

Hoffman, D. D. (2000). *Visual Intelligence: How We Create What We See*. New York, NY: W.W. Norton.

Hogue, D. (2011). *Part 1 - Five Essential Principles of Interaction Design*. Adobe TV. Retrieved from http://tv.adobe.com/watch/classroom-five-essential-principles-of-interaction-design/part-1-five-essential-principles-of-interaction-design/

Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford, UK: Oxford University Press.

Johnson, C., & Gonzalez, A. (2008). Automated After Action Review: State-of-the-Art Review and Trends. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, *5*(108).

Keynote Alan Cooper at TNW2012. (2012). Retrieved from http://www.youtube.com/watch?v=Qb1umw8IMBw&feature=youtube_gdata_player

Kissane, E. (2011). *The Elements of Content Strategy*. New York, NY: A Book Apart.

Koger, M. E., Long, D. L., Britt, D. B., Sanders, J. J., Broadwater, T., & Brewer, J. D. (1996). *Simulation-Based Mounted Brigade Training Program: History and lessons learned*. Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences.

Laughlin, D. (2011). *Initial Report*. Virtual Reality Applications Center: Iowa State University.

Laughlin, D., Peterson, A., & Camou, J. (2011). *OmniScribe: Functional Requirements*. Virtual Reality Applications Center: Iowa State University.

Le Poidevin, R. (2011). The Experience and Perception of Time. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2011.).

Mao, J.-Y., Vredenburg, K., Smith, P. W., & Carey, T. (2005). The State of User-centered Design Practice. *Commun. ACM*, *48*(3), 105–109. doi:10.1145/1047671.1047677

Meliza, L. L., & Brown, B. (1996). Increasing the speed flexibility of feedback systems for PIS exercises. *Proceedings of the 18th Interservice/Industry Training Systems and Education Conference*, 132–139.

Meliza, L. L., & Tan, S. C. (1996). *SIMNET Unit Performance Assessment System (UPAS) Version 2.5 User's Guide*. U.S. Army Research Institute for the Behavioral and Social Sciences.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81.

Morrison, J., E., & Meliza, L. L. (1999). *Foundations of the After Action Review Process*. U.S. Army Research Institute for the Behavioral and Social Sciences.

Norman, D. A. (1999). Affordance, Conventions, and Design. *Interactions*, *6*(3), 38–43. doi:10.1145/301153.301168

Norman, D. A., & Draper, S. W. (1986). *User Centered System Design; New Perspectives on Human-Computer Interaction*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Patnaik, D. (2009). *Wired to Care: How Companies Prosper When They Create Widespread Empathy*. Pearson Education.

Peterson, A., Gilbert, S., Winer, E., Welch, J., de la Cruz, J., & Gonzalez, H. (2012). OmniScribe - Enhancing AAR in an LVC Environment. *Interservice/Industry Training, Simulation, and Education Conference*.

Portigal, S. (2013). *Interviewing Users: How to Uncover Compelling Insights*. Rosenfeld Media, LLC.

Prieto-Diaz, R., & Freeman, P. (1987). Classifying software for reusability. *Software, IEEE*, *4*(1), 6–16.

Quarles, J., Lampotang, S., Fischer, I., Fishwick, P., & Lok, B. (2008). Collocated AAR: Augmenting After Action Review with Mixed Reality. *7th IEEE/ACM International Symposium on Mixed and Augmented Reality 2008*, 107 – 116.

Raij, A., & Lok, B. (2008). IPSVIz: An after-action review tool for human-virtual human experiences. *Proceedings of The IEEE Virtual Reality Conference (2008)*, 91–98.

Roam, D. (2008). *The Back of the Napkin: Solving Problems and Selling Ideas with Pictures*. London, England: Penguin Group.

Saffer, D. (2013). *Microinteractions: Designing with Details*. O'Reilly Media, Inc.

Sikora, J., & Coose, P. (1995). What in the world is ADS? *PHALANX*, *28*(1), 6–8.

Slamecka, N. J., & Graf, P. (1978). The generation effect: Delineation of a phenomenon. *Journal of Experimental Psychology: Human Learning and Memory*, *4*(6), 592.

Sullivan, G. R. (1995). *Gordon R. Sullivan: The collected works 1991-1995* (No. CMH Publication 70-44). Washington, DC: U.S. Army Center of Military History.

Sullivan, G. R., & Harper, M. V. (1997). *Hope is not a method: What business leaders can learn from America's Army. New York:* New York: Times Business, Random House.

Tidwell, J. (1999). Common ground: A pattern language for human-computer interface design. Retrieved from http://www.mit.edu/~jtidwell/common_ground.html

Tidwell, J. (2011). *Designing Interfaces* (2nd ed.). California: O'Reilly Media, Inc.

Tufte, E. R. (1990). *Envisioning Information*. Connecticut: Graphics Press LLC.

U.S. Army Training and Doctrine Command. (1997). *The Standard Army After Action Review System (STAARS): After Action Review (AAR) Handbook* (No. Version 2.1). National Simulations Center.

Urban, G. L., Hauser, J. R., & Dholakia, N. (1987). *Essentials of new product management*. Prentice-Hall Englewood Cliffs, New Jersey.

UX Week 2009 | Jesse James Garrett | The State Of User Experience. (2009). Retrieved from http://vimeo.com/6952223

Weinschenk, S. (2011). *100 Things Every Designer Needs to Know About People*. Pearson Education. Retrieved from http://books.google.com/books?id=yODWdMUhkFwC

Word, L. E. (1987). *Observations from three years at the National Training Center*. Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences.