The Impact of Rich Application Frameworks on User Experience Design

Tobias Komischke

Infragistics, Inc., 2 Commerce Drive, Cranbury, NJ 08512, USA
tkomischke@infragistics.com

Abstract. Rich internet and desktop application frameworks provide advanced features like direct object manipulation and advanced animations that can enhance the user experience. This paper traces the history and evolution of rich applications and in front of the background of the T.O.T.E. framework discusses the benefits that rich application capabilities provide for the human-system interaction.

Keywords: User Experience, Design, Rich Interactivity.

1 Introduction

When the first graphical user interfaces (GUIs) entered the market in the 1980s the use of pictorial and metaphoric user interface (UI) elements were revolutionary and significantly contributed to higher computer adoption rates. Direct manipulation of UI elements was realized through clicking and dragging and was performed on basically the same set of interactive UI elements that we still have today, such as push buttons and scrollbars. The behavior of these elements was mainly binary: on/off, show/hide, open/close.

Today, the input methods (clicking, dragging) are still the same for mouse-based applications (obviously, we now have multi-touch capabilities on more and more systems), yet the output behavior that the user interface is capable of showing is much richer. Features like drag & drop, fading, sliding, and transitions allow for more nuanced system output. This can add to the user experience both from a pragmatic and a hedonic standpoint and will be explained in the sections below.

It's worth looking at the history of rich interactivity as it shows that two areas of software have been influencing and boosting each other: the desktop world and the browser world. Before web browsers were invented in the 1990s, personal computing has been done based on desktop applications such as the Microsoft Office suite. The first browser-based user interfaces were static web pages with hardly any option to directly manipulate UI elements other than entering ASCII onto forms. Later Adobe Flash made it possible to provide multimedia content that could also be interacted with. It took about 15 years until the first browser-based applications were available that could match the interaction capabilities of desktops. These so-called Rich Internet Applications (RIAs) were realized through technologies such as Flash and Java and were driven by the need of optimizing the appeal of web-based content presentation (multimedia integration, sophisticated animations, etc.). The typical websites that

pushed this development forward were marketing and advertising sites. Since then, more technologies like Ajax, Adobe Air, Microsoft Silverlight were brought to the market and allowed an ever growing number of designers and developers to create RIAs whose interactivity went beyond the desktop world.

With RIAs more and more becoming the gold standard for the way user interfaces should look and feel, the desktop world in recent years adopted the concept of rich interactivity under the name of rich desktop application (RDAs).

As of today, desktop and web applications have converged and now are hardly distinguishable. Development platforms like Microsoft Silverlight allow running web applications outside of a browser like a desktop application. Because of this, for the rest of this paper the term “rich applications” is used instead of distinguishing between RIAs and RDAs.

There is a fine line between boosting and wrecking the user experience of rich applications. Rich development platforms together with 3rd party controls enable programming highly sophisticated interactive features with a minimum of effort. Yet, even if something is easy to program and rich in its presentation, is it really useful, usable and desirable?

2 Benefits of Rich Interactivity

2.1 Cognitive Psychology Background

Human-system interaction can be considered as a sequence of problem-solving tasks. One of the earliest and most fundamental models is T.O.T.E. which stands for “Test – Operate – Test – Exit” (Miller, Galanter, Pribram, 1986). As a first step an assessment (test) about the current system state is made. If the system is not in the targeted state already, an operation (e.g. a user input) is executed to change the current state towards the desired state. Another test is made to see if the targeted state has been reached or not. More recurrences are performed until the test result is positive. Only then the sequence ends.

Each T.O.T.E. unit can be decomposed into sub-processes:

- Test:

- *Perceive Status*

On GUIs, the status is typically perceived through the visual sense, i.e. something needs to be displayed on the screen. The more prominently this signal is shown against other UI elements, the faster it can be found and seen. Rich interaction features such as animations can help make status elements more salient.

- *Understand Status*

Once data has been perceived, the next step is to understand it. On a user interface, for example, a status message may be clearly visible, but its meaning may be ambiguous to the user.

- *Compare w/ Goal*

Once the meaning of a current status is understood, it has to be put in relation to the envisioned goal. How near or far am I away from that goal?

- *Plan Operation*

In order to bridge the gap between current and envisioned state a strategy has to be developed.

- *Anticipate Outcome*

An operation is selected based on the expectation that it will reduce the gap between current and envisioned state. So one way to select from several operations is to anticipate its outcome, i.e. its effectiveness.

- Operate:

- Initiate Operation

Even if the current state has been correctly assessed and an effective operation has been planned, it can be a challenge to start executing on that operation. In the area of user interfaces an example is a user who knows that s/he has to increment a value on a virtual gauge, but doesn't know how to do it.

- Monitor Operation

For operations more complex than a single click, the actual execution has to be monitored. This can be viewed as a nested T.O.T.E. within the Operate T.O.T.E. unit. For instance, at the end of a drag & drop action the user has to monitor whether the dragged element is on the target drop zone, otherwise the action won't be successful.

- Test:

- see above –

- Exit:

- Re-Orient

After successfully ending a T.O.T.E. sequence the attention has to be shifted to the next activity. On user interfaces this may involve scanning the screen for other information, other visual targets that allow initiating the next T.O.T.E. sequence.

The drill-down into the sub-processes of T.O.T.E. units helps to understand the complexity behind seemingly simple user activities within an application. It also helps understanding how rich interactivity can support the human-system interaction. This will be explained in the next sections.

2.2 Nuanced Feedback

For human-system interaction that includes direct manipulation of UI elements, the analysis of T.O.T.E. above showed that during longer interactions like for Drag & Drop the user needs to monitor the operation, i.e. making sure that the object to be relocated is dragged on to a valid target zone. De-composing a Drag & Drop action into more atomic events leads to what Bill Scott (Scott, 2009) calls “interesting moments”, incl.:

- Mouse hover
- Mouse down
- Drag initiated
- Drag leaves original location
- Drag re-enters original location

- Drag enter valid target
- Drag exits valid target
- etc.

Scott lists 15 events for Drag & Drop. With the user input being the same (mouse down – dragging of element to target zone – mouse up), rich interactivity provides several alternatives on how the system output during the Drag & Drop action can be presented (see Figure 1):

- The appearance of the drop zone can be changed.
- The appearance of the drag object itself can be changed.
- The appearance of the mouse cursor can be changed.
- Any combination of the above.

The last bullet calls attention to the fact that combinations of rich behavior can be detrimental for the user experience as too many different stimuli have to be perceived and understood at the same time.

Fig. 1. Rich system behavior for Drag & Drop: mouse cursor indicating that the target zone has not been reached yet

Traditionally, in UI Specs standard mouse events have been documented, for example what happens on Mouse hover, Mouse down, Mouse up, etc. With the capabilities of designing Scott's interesting moments, the documentation effort is significantly higher as with rich interactivity different system outputs can be defined for different interesting moments. Even when these are documented in a specification document, consumers of that specification may find it hard to envision what the rich interaction looks and feels like. Building mock-ups for demonstration conveys rich concepts better than static words and images, but the effort to create them is significantly higher as well.

2.3 Keeping Users in the Same Screen Context

One of the biggest challenges in user interface design is to shorten navigation paths and de-cluttering screens – both at the same time. Rich interaction concepts allow keeping users on the same screen longer by providing means that go beyond the traditional tabs, accordions and blinds that only have to states (on/off, expanded/collapsed, shown/hidden, etc.):

- Microsoft uses a composition model in WPF and Silverlight that allows placing any UI element into any other UI element. For example, a rich tool tip on a grid cell displaying last year's total sales in the U.S. can host a visualization of the regional sales numbers that make up the total number – this contextual information is provided right where it is needed without requiring the user to navigate somewhere else in order to find it.
- User experience patterns such as in-line editing or drill-down charts enable the user to keep working on an item without leaving the current screen context. Navigation steps are minimized and consequently the perceived complexity of an application is reduced.
- Oftentimes, data objects and functional objects that are only infrequently needed are positioned in fly-out menus that can be brought into view on demand. Rich interaction patterns can help reduce the effort in terms of user for accessing these “additional” spaces through more seamless ways of access (interesting moments, see above).

2.4 Education

Interaction concepts can be challenging for users to learn and understand. Rich interactivity can be used for educating users. For example, when a user clicks on a UI element which in turn is replaced by another object (e.g. drill-down charts, see above) animations showing how the new object superimposes on the initial object help communicating that switch that would otherwise be binary and instant. In that respect animations can support the concept of location consistency, i.e. that each object has its place on the UI, even though it may not be visible at all times.

Another example of how a rich interaction feature contributes to a better understanding of a UI is the so-called UX pattern “Magnetism”. It specifically helps making affordances explicit. Magnetism denominates the behavior of an interactive UI element when the mouse pointer comes close. Imagine a fly-out menu that is collapsed with only few pixels sticking out of the side of the screen in order to maximize screen real estate. Once the mouse pointer gets close, the menu will move inward by a couple of pixels. By that, it is communicated that the element is interactive (since it reacts to the mouse pointer being close by). Furthermore, its intended behavior is indicated through the movement.

An important aspect of user education is feedback. Especially on web sites the user oftentimes does not perceive and understand feedback, e.g. after input validation has returned an error on a form. This can be explained by change blindness (Simons, Ambinder, 2005). Through rich interactivity the feedback message or alert can be asynchronously loaded and be animated to draw attention.

2.5 Entertainment

Entertainment is an integral part of many web sites to help attract and retain site visitors. The convergence of desktop and web worlds has also brought with it the deployment of entertaining features into line-of-business applications. For these applications it feels counter-intuitive to incorporate entertainment, as productivity is one of the biggest goals. And still, entertainment serves its purpose and richness adds to its quality. Entertainment in line-of-business applications constitutes all those features of a front-end that serve the purpose of diverting the users and holding their attention. Entertainment features are not required for the accomplishment of user objectives. In most cases, the goal is to bridge waiting time for the user during the initial start-up of the application, during times when more data and content is loaded as part of using the application, and during processing times. In all these situations the user is put into a passive mode where he or she has to wait before relevant information is displayed or inputs can be made.

The start-up of an application, for example, can be lengthy. During this time, so-called splash screens are typically shown, initial screens containing information like product name, company logo, version information, etc. In recent years splash screens have become richer. Animations and multimedia see to it that the perceived waiting time is shortened. The richness also generates an expectation towards the interaction quality of the product once it has been loaded.

3 Outlook

User interface design is constantly evolving and advancing. New concepts and new technologies build the basis for novel interactive systems. The role of user experience design remains the same – to center the design of a software product on the needs, capabilities and limitations of the users. User-centered design processes and methods remain crucial in order to exploit the strengths and to mitigate the weaknesses of new technologies.

References

1. ISO 9241-210, Ergonomics of human-system interaction – Part 210: Human-centered design for interactive systems. Geneva: International Organization for Standardization (2010)
2. Miller, G.A., Galanter, E., Pribram, K.A.: Plans and the structure of behavior. Adams-Bannister-Cox, New York (1986)
3. Scott, B., Neil, T.: Designing Web Interfaces: Principles and Patterns for Rich Interactions. O'Reilly Media, Cambridge (2009)
4. Simons, D.J., Ambinder, M.S.: Change blindness: Theory and consequences. *Current Directions in Psychological Science* 14(1), 44–48 (2005)