Differentiate Application Look, Feel and Function with Autonomous User Interfaces

Robi Karp, Fluffy Spider Technologies

The earliest embedded systems – industrial automation and scientific instruments communicated with operators using physical actuators: knobs, dials, gauges, indicator lights, etc. Today's intelligent devices, especially in consumer electronics, feature more sophisticated operator interaction and boast user interfaces (UIs) comparable to desktop applications. Leading this trend are mobile devices like smartphones and web pads, in-vehicle infotainment (IVI) systems and home entertainment (media players, HDTV, DVRs and STBs).

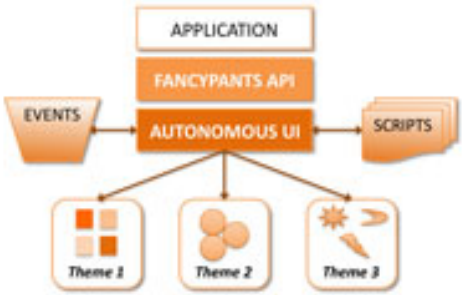


Figure 1. Different presentations of the same application.

Traditional UI design methods give application code "ownership" of particulars of UI implementation: type, orientation, placement and other attributes of objects on the display (buttons, widgets, etc.), and the flow of element use. Such application-centric UI design fixes device "personality" limiting downstream customization by channel partners, third-parties and end-users. While some frameworks support theming – changes to color schemes, text styles, window frames, widgets, etc. - the fundamental structure and flow of an application UI remains a closed box as conceived by its original designers.

This article considers an alternate UI design paradigm – the Autonomous User Interface. This is a concept to let OEMs specify general purpose presentation of controls, widgets and even content, but also gives downstream developers the freedom to brand and customize.

Decoupling UI and Application

Autonomous UI design goes beyond custom themes, icon sets and color schemes common on many mobile phones and other intelligent devices. It lets downstream developers to:

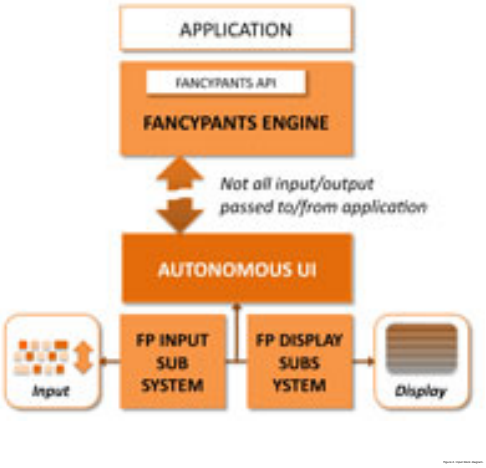
•Associate custom functionality to individual UI elements.

•Add or remove items from UIs, including images, videos, and widgets without modifying application code.

•Support interaction with new device events and capabilities, like shaking and orientation (accelerometer), location and movement (GPS), and definable data and network events and handle events from after-market devices (e.g., game controllers).

•Work with integrators, operators to add new UI personalities.

Autonomous UI Architecture

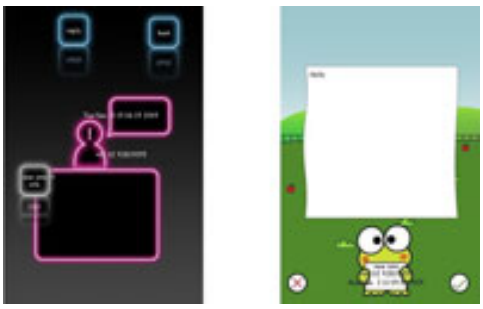


Separating application and presentation code doesn't break core application design. The application design team must determine, however, which interface functions reside within the application itself and which belong to an autonomous UI.

An autonomous UI does require support from the underlying graphical/multimedia framework and an open, high-level API for developer use. At Fluffy Spider Technologies, we chose C in building FancyPants and for API libraries. For autonomous UI implementation, we bind to the Lua scripting language, taking advantage of Lua features and rapid prototyping capabilities.

Below are some examples.

SMS on Mobile Phones



[1]

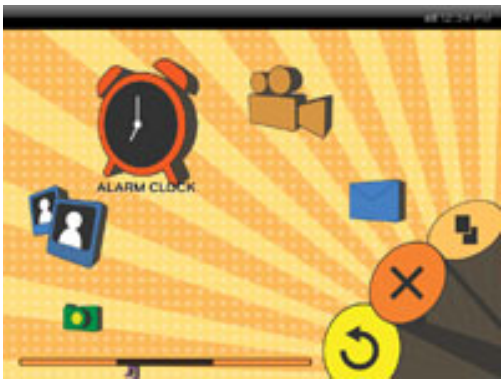
Handset manufacturer typically include an SMS (Short Messaging System) application on devices. SMS applications usually feature a straightforward (uninspired) display of messages and addressees. Mobile network operators (MNOs) and other channel partners have few options for customizing or branding, instead passing mundane software through to users “as is”.

An SMS client designed with an autonomous User Interface would give MNOs and other ecosystem players more options for customization and differentiation. For example, MNO developers could augment addressee information with status and location-based data from the operator’s network. Similarly, a software supplier could offer alternate look-and-feel package to that same SMS client, adding new functions like accelerometer input or GPS coordinates, all without changing original application code.

Other Consumer Devices

The same paradigm applies to devices like set-top boxes and DVRs, internet TVs, automotive GPS systems and media players. Autonomous UI capability lets cable and satellite operators, automotive manufacturers and car rental companies – virtually any channel participant – customize, brand and extend device interface functionality without access to application source code.

Multiple Product Families



Depending on the industry, a new product (not just a new product version) can

require between 2 to 10 or more man-years of engineering effort to reach the market. Consumer devices, with short life cycles, also need to hit short market windows.

A large part of device engineering lies in the creating compelling, differentiated user interfaces. For manufacturers creating broad product families, being able to deploy the same application code base with different user interfaces saves time, money and also helps focus development on truly differentiating features. In new product iterations, an autonomous UI helps new products arrive to market more quickly and confidently.

Supporting OEM Operating PlatformsM

Manufacturers often choose an off-the-shelf OS like Android or WinCE to save on core engineering, and to leverage ecosystems that revolve around those platforms. However, these OSes leave little room for branding and customization. Unless OEMs invest in significant incremental engineering, users will be greeted with the same UI as on every other Android-based gadget, commoditizing new devices. When OEMs do invest in differentiating, they will need to repeat that effort with each new OS release.

Benefits of Autonomous UI

Autonomous UI design is not just another application architecture fad. It offers developers benefits that emerge directly from decoupling UI and application code.

Additional benefits include:

•Speeds development for creating new, unique user interfaces without modifying application code.

•Enhances branding with unique look and feel, even on common commodity platforms.

•Enables adding intelligence to existing UI code, e.g., to make decisions and process events not in the original design.

•Shortens Q/A time by integrating pre-tested/integrated application code.

•Helps OEMs create product families based with a single code base.

Conclusion

Autonomous UI support is central to my company’s FancyPants graphical and multimedia platform, but embodies core principles that can be applied to using other frameworks and toolkits. Decoupling UI and application design is fairly new concept, but it handily accommodates today’s complex consumer electronics

ecosystem and channel. In today's dynamic landscape of multiple applications OSes and burgeoning device SKU counts, it's important to build a strong base product that can be easily tailored for different packages, channels and markets.

Robi Karp is CEO of Fluffy Spider Technologies, www.fluffyspider.com

Source URL (retrieved on *03/09/2014 - 7:18pm*):
http://www.wirelessdesignmag.com/articles/2010/07/differentiate-application-look-feel-and-function-autonomous-user-interfaces?qt-most_popular=0

Links:
[1] http://www.wirelessdesignmag.com/sites/wirelessdesignmag.com/files/legacyimages/1006/wd_FST_Fig3_lrg.jpg