CLASSIC
SYSTEM SOLUTIONS, INC.

Implementing Visual Design Patterns

By James Hobart, President, Classic System Solutions
August 2002

The next evolution in capturing and implementing design knowledge will be the use of visual design patterns (VDP). VDPs offer a powerful new way of focusing design solutions, based on context, by telling the developer when, why, and how the design solution can be applied successfully. Applying VDPs to complex systems is both challenging and potentially very rewarding. Countless designs have been deployed to very large user communities only to result in poor usability. The proprietary nature of these types of applications, coupled with the proliferation of deployment platforms, makes it difficult for organizations to compare successful outcomes among teams and to cooperatively develop VDPs to solve common design problems. VDPs can provide massive benefits to users, as the repetitive nature of patterns can translate into significant cost reductions in training and support for both internal and external users. Implementing a way of capturing, managing, and delivering design standards and VDPs across the enterprise can maximize the use of the design assets, improve the efficiency of the development process, increase the productivity of the development teams, and achieve consistent and usable designs.

Implementing Visual Design Patterns

Implementing VDPs into a development process can be challenging. Although developers may be familiar with traditional software engineering design patterns, they may not be accustomed to using this approach to create new visual interfaces and may be skeptical of their value to the coding process. Gaining consensus among a project team on which VDPs work best for a complex transactional system requires a great deal of iterative design work, team facilitation, and usability testing. Taking the time to document and validate the VDPs requires discipline on the part of the project members and a constant vigilance to identify, refine, and test the VDPs as they evolve.

What should be in a Visual Design Pattern Repository?

We view the VDP repository as a dynamic, growing source of design knowledge supported by your existing expertise. Capturing, refining, documenting, and re-using VDPs is a continuous journey requiring an ongoing commitment. Ultimately, VDPs should be shared across enterprises, thus leveraging the effort and work of the human factors teams behind them. As the collection of VDPs grows, the task of organizing them for use by your development teams grows. We have found that the best way of organizing VDPs is by type of usage. The developer looking for a VDP typically has a specific design issue and translates that issue into a question. For example: *How do I allow very experienced keyboard-oriented users fast, efficient entry of a full calendar date in a web application?*

Ideally, the developer should be able to easily browse a set of categories related to the issue to find a VDP. As the repository grows, a non-categorized search and a search by problem will be needed to find the right VDP.

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

A partial list of VDPs that we have developed for transactional systems includes the following.

Data Entry	Navigation
Simple form	Menus
Complex form	Repetitive transactions
Transactional save / validation	Tabs
Dates (calendars)	List management
Grid / table entry	Object / sub-object

Providing Usage-Centered Design Context to VDPs

By employing usage-centered design techniques, we can provide a great deal of context to a VDP that provides guidance to new team members on when, how, and why to use the VDP. VDPs are very dependent on the models employed during the usage-centered design, like the user role, task, and operational models. As these models are developed and refined for a project, an enterprise repository of VDPs can be defined. For instance, if a project is directed towards a mobile executive user who performs infrequent tasks like scheduling appointments, a VDP for a PDA calendar can be identified. The developer can then study the VDP for its applicability and, eventually, use the supporting code bases, design guidelines, and case studies to help solve the issue. Feedback from implementation of the VDP can then be sent back to the repository and associated models.

Developing User Role Models

When designing systems, we often start our role models with three basic user types: customer, employee, and supplier. These roles can often be easily mapped to VDPs based on assumptions about their usage. For instance, the three macro roles fold nicely into the B2C (business to consumer), B2B (business to business), and B2E (business to employee) genre of VDPs now being employed by enterprise resource software vendors like SAP and PeopleSoft.

A Multi-Column List VDP Examined

Let's take a problem that a developer may face and apply a VDP to see how it works. Let's presume that the developer is creating a web-based transactional application that allows the user to display, manipulate, and act upon line items in a multi-column list. Upon building a user role model, the developer determines that the primary users of the list will be semi-trained employees with occasional need to perform the self-service human resources tasks from a web-based desktop either at home or work. Further development of use cases and their success scenarios help us further choose from a few variants on the available VDPs. Based on the design problem being addressed, the developer can then access the repository of VDPs and quickly filter down to those related to list management with a web-deployment for semi-trained users who perform the task infrequently. The developer then can quickly scan a list of available visual examples of the VDPs and select the most applicable one. Here is an example of a possible layout for this type of VDP.

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

Below: A web-based list, semi-trained user, limited actions on the row

Help

Dependents Personal Data

Below is a list of all your dependents and beneficiaries. Click on a name to review personal information.

Simon Schumacher

Name	Relationship	Type		
Kala Hari	Daughter	Dependent and Beneficiary	Edit	Delete
Milit Hari	Spouse	Beneficiary Only	Edit	Delete
Dirti Hari	Son	Dependent and Beneficiary	Edit	Delete

Add Click the Add button to add a new Dependent or Beneficiary.

The same problem can be addressed with a variation on this VDP if the user role model indicates a high-volume transactional worker within a human resources department who is comfortable using a client/server-based application and who needs to perform a variety of actions on the items in the list.

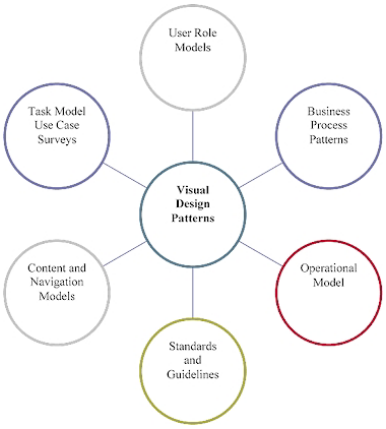
Below: A web-based list, expert user, many actions on the row

Name	Relationship	Type	Action
Hari Kari	Daughter	Dependent and Beneficiary	Add
Milit Hari	Spouse	Beneficiary Only	
Dirti Hari	Son	Dependent and Beneficiary	Add
			Delete
			Suspend
			Review

VDPs Supported by Models

As the usage-centered models evolve in an enterprise, the associated VDPs will be continually validated by use cases early in the design and actual usage in production. The VDPs will be refined with usability and field testing, thus preserving the hard work performed by the design teams. The essential models of usage-centered design become the context that supports the growing repository of VDPs.

Below: Visual design patterns supported by contextual models



101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

What Does a Complete VDP Look Like?

A full implementation of VDPs requires more than just the visual layouts. Our experience has shown that VDPs supported by a robust, yet concise, base of design knowledge are much more likely to be implemented by developers. Developers who can evaluate the VDP to see how it can be used in their design problem will be more likely to use the VDP rather than forging ahead with new designs. VDPs should be easy to update and share across teams and should be stored in a web-based repository on the corporate intranet. In addition, VDPs should be supported by known good and bad examples. Examples can include information about deployment, experience and ratings from usability tests, and interactive items delivered via animation, so that the developer can quickly see how a VDP would behave "in action". The VDP should also be supported by one or more design guidelines, adding rigor to why the VDP is presented as a best practice.

Interestingly, we have found that this approach is excellent for getting developers to actually read guidelines. Their interaction normally starts with the VDP and follows to reviewing the supporting guidelines once they feel that the VDP has merit for solving their design problem. Experiential data such as case studies, quality assurance and design checklists, and code templates are also linked to the VDP to provide the "what, why, and how" total solution to the design problem. During the usability testing of our repository, we found that many developers were not familiar with the term "pattern", so we substituted the word "solutions" for increased clarity. The VDP example below is a from our repository product called GUIguide.

Below: Example of a multi-column list VDP from GUIguide

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

Solutions - Multi-column list

Back | Edit

Multi-column list

Name:	Multi-column list
Problem:	The user needs to view and manipulate multiple lists.
Principle:	Overview
Context:	The user has several lists to manage. The items in each list are typically ordered and may be quite numerous. There may be multiple columns of data in a list.
Forces:	- The user wants to see as many of the items in the list as possible at one time. - There is often limited display space. - The user may want to sort the columns of data. - Some operations can be performed on a single item in the list. - Some operations can be performed on more than one item in the list.
Solution:	Provide access to the total list and sorting when possible. Allow intermediate and advanced users to size and order the columns in the list. Provide visual feedback on the number of items displayed relative to the total number of items in the list.
Rationale:	By seeing an overview of the list items, the user can more easily scan the available options, detect patterns in the data, and more readily make a selection.
Known Uses:	http://www.google.com ; http://www.yahoo.com

Examples: Add

Good | Del |

List - Column sort indicator

A small visual cue in the column header that indicates the column currently being used to sort the data and the direction of the sort.

View full-size image

Bad | Del |

List - Overload

The use of too many lists or list choices cognitively overloads the user.

View full-size image

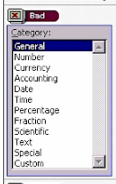
Good | Del |

List - Column headings

List with the column headings aligned to the left.

View full-size image

Bad | Del |



List - Scroll bars

Don't show a scroll bar on the right side of the list if scrolling is not available.

Good | Del |

List - HTML table

List presented in an HTML table. Does not allow column sorting.

View full-size image

Related Solutions: Add

No solutions associated

Related Guidelines: Add

CSS1096 - Choosing number of items in list boxes	Del
CSS1523 - Identifying columns in grids	Del
CSS1526 - Showing columns in grids are resizable	Del
CSS1546 - Using list views	Del
CSS1553 - Aligning column headers in grids	Del
CSS1558 - Making grids guide users	Del
CSS1559 - Aligning data in grids	Del
CSS171 - Using grids	Del
CSS4 - Choosing height of column headings in tables	Del
CSS43 - Avoiding scroll bars in grids	Del
CSS499 - Filtering list boxes with over 40 items	Del
CSS659 - Arranging controls in windows	Del

Related Case Studies: Add

Table Hyperlinks - How can I make my table's hyperlinks less confusing? - Navigation	Del

Related Checklists: Add

What to verify on list boxes	Del

Related Resources: Add

HTML Tags - List elements	Del
Cascading Style Sheet Tags - Table properties	Del

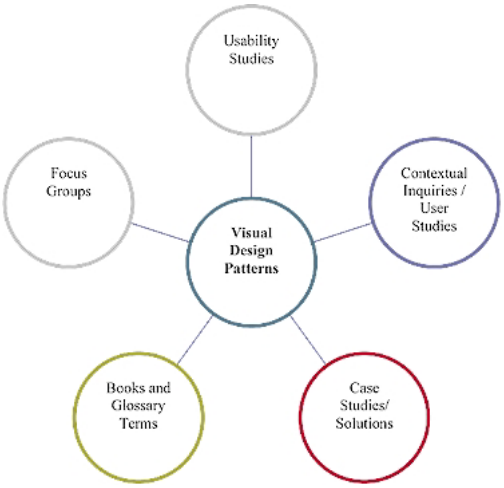
Submit Exception

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

VDPs Supported by Team Experience

Design teams learn best by sharing their experiences. An hour in the observation room of a usability lab provides a developer with vast amounts of information on users attempts to use their newest software creation. In addition to providing real-time experiential learning, online tools designed to deliver expertise on-demand in a developer-friendly format should also be provided. Developers can access these tools during the design process to help them learn how to create usable, consistent, and intuitive user interfaces. Examples of such tools include case studies, glossaries of terms, usability studies, and book reviews that point developers to useful reference information that support their design decisions and the VDP repository.

Below: VDPs are supported by experiential data



Summary

Employing proven VDPs in the development process is the key to improving the consistency of user interfaces for complex transactional web applications. And, consistency is the key to delivering ease-of-use. Supporting VDPs with rich contextual information provided by the models developed by a usage-centered design approach helps ensure design teams that the VDPs are relevant and suited to solving their design problems. VDPs that include code templates, navigation widgets, and other design assets increase the efficiency of the development process by encouraging the reuse of those assets within and across design teams. Capturing existing design assets and documenting design expertise in a central, accessible repository allows enterprises to retain their investment in staff development, while simultaneously reducing the time wasted on redundant design efforts.

Investing in VDPs is essential to delivering consistent user interfaces. However, traditional approaches of implementing VDPs often fall short of expectations because of difficulties in managing, using, updating, and enforcing the VDPs after adoption. Fortunately, products are now being developed to offer a complete, flexible, and scalable solution for capturing, managing, and delivering design knowledge across the enterprise. Using their corporate network, enterprises can create a convenient, searchable repository of VDPs, guidelines, case studies, checklists, and other resources that the entire design team can access on demand. A collaborative system with open access, like GUIguide, allows everyone in the enterprise to contribute their design knowledge to the repository.

101B Sand Creek Road • Suite 209 • Brentwood, CA 94513
925.250.2550 • 925.516.9658 • www.classicsys.com

VDPs allow enterprises to improve the usability of their software and web applications by leveraging design knowledge and enabling efficient development practices. More than just standards, patterns-based solutions offer great hope in dealing with the increased complexity of designing systems. Investment in the development and deployment of a solutions-driven approach must provide a way of capturing, managing, and delivering design knowledge, so that enterprises can develop easy-to-use software on time and on budget.

About the author:

James Hobart is an internationally recognized user interface design consultant based in California, USA. He specializes in the design and development of large-scale, high-volume client/server and web applications. He is an expert in GUI design for transaction processing systems and strategies for migration to thin-client graphical user interfaces. He can be reached at jimh@classicsys.com

Find more articles on Usability at www.classicsys.com