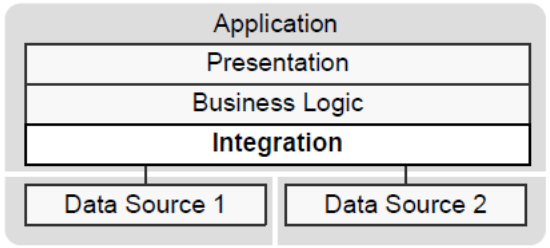
UI Integration

CIS 8020 Systems Integration

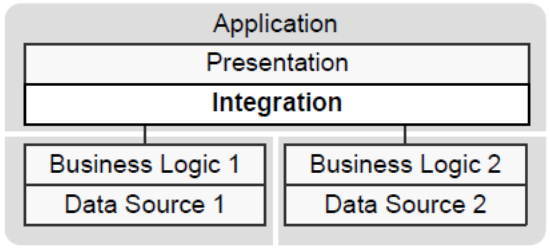
Jack G. Zheng

October2009

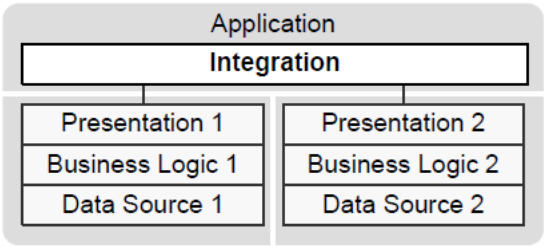
UI Integration



(a) Integration of different data sources



(b) Integration of distributed business logic elements



(c) Integration of two autonomous applications

Florian Daniel, et al., "Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities Export", IEEE Internet Computing, Vol.11, No.3 (2007), pp. 59-66.

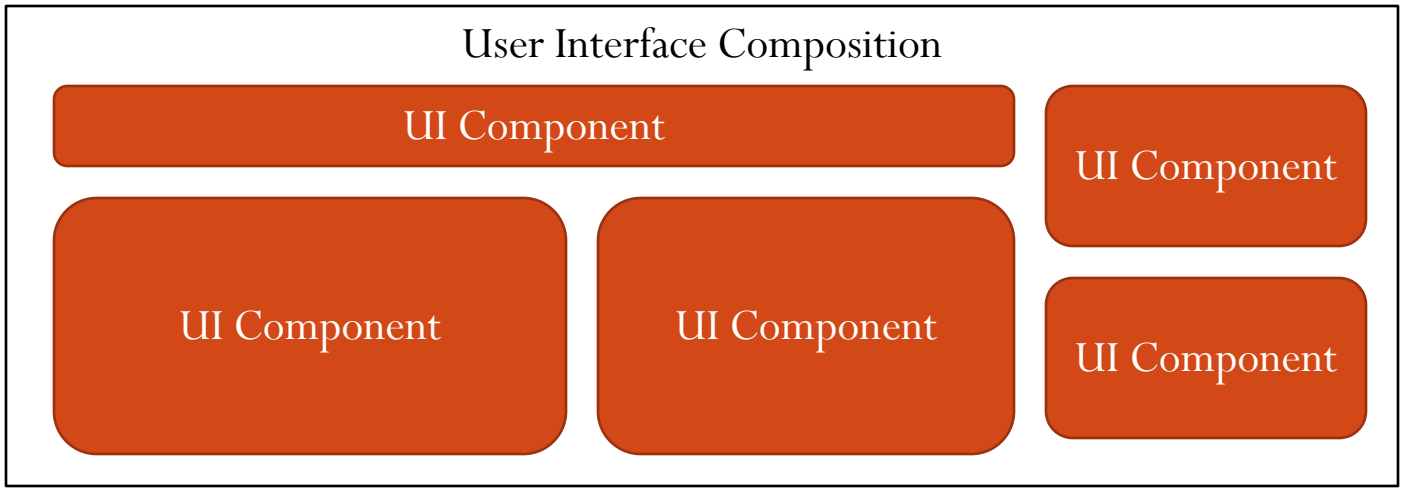
UI Integration

- UI integration is particularly applicable when
  - application or data integration just isn't feasible (such as when applications don't expose business-level APIs)
  - developing a new UI from scratch is too costly (such as when the component application changes frequently or its UI is overly complex – e.g. maps, charts, etc.)
  - there is no need for further integration except for displaying purpose

Florian Daniel, et al., "Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities Export", IEEE Internet Computing, Vol.11, No.3 (2007), pp. 59-66.

General Strategy for UI Integration

- The general integration strategy is to use UI components
  - Components are good for re-use
  - Components are flexible to be composed together

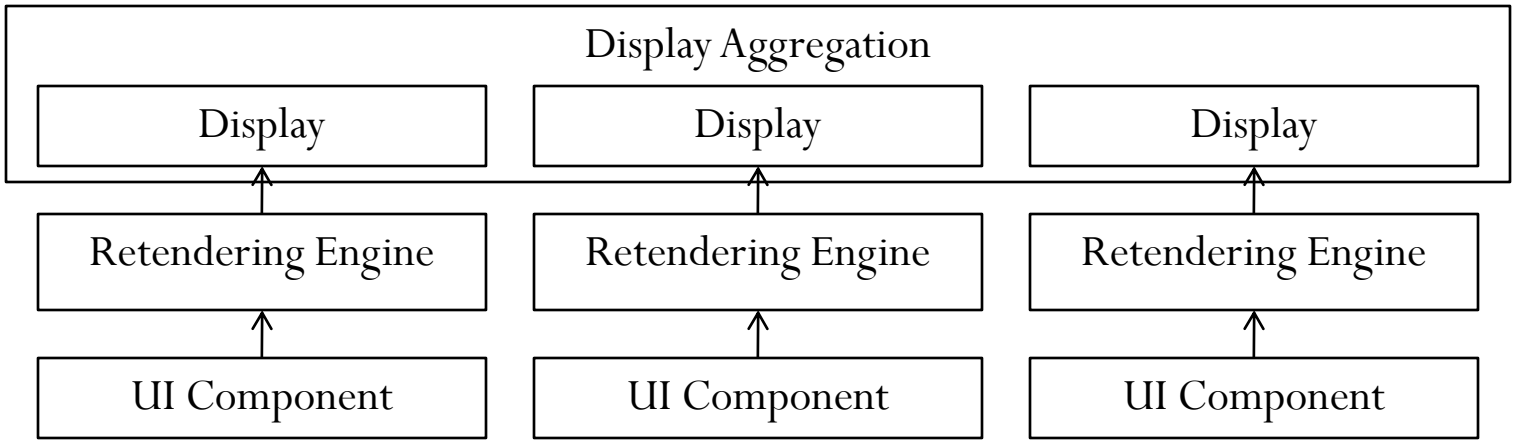


UI Component Styles

- Loosely defined components
  - UI components do not necessarily conforms to a common model or standard
    - Requires more ad hoc composition processing
    - Web pages
- Strongly defined components
  - Generally conforms to a common model set by the composition framework (container)
    - Composition processing is handled by the platform
    - Portlets

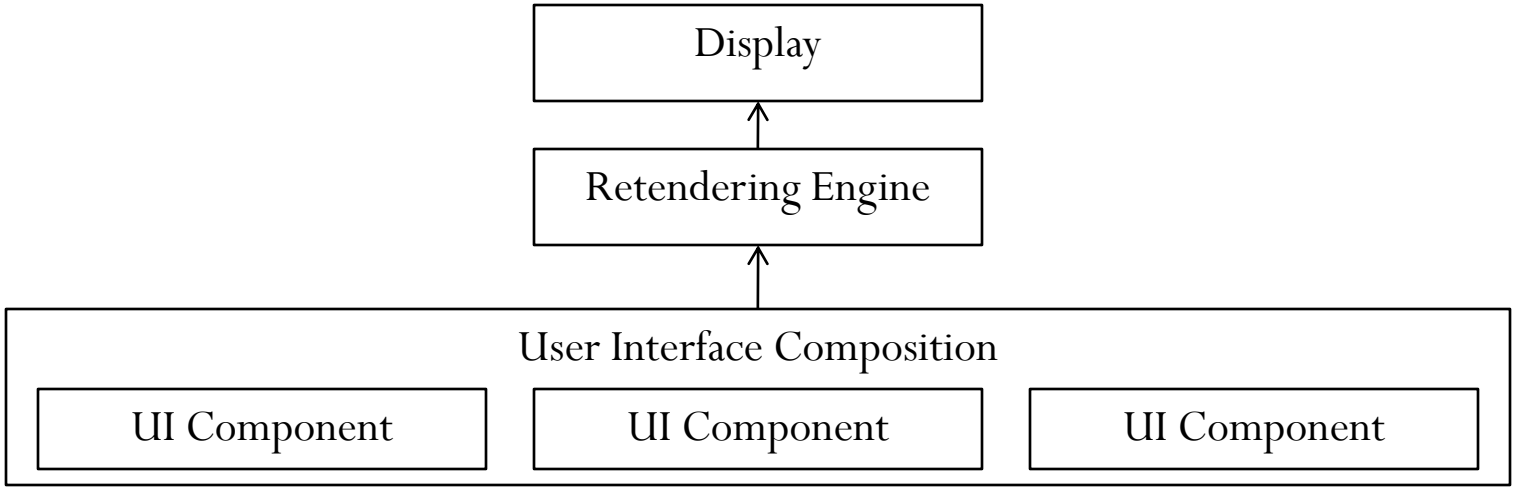
UI Component Rendering at the Component Level

- Each component handles its own UI rendering and display
- The composition is an aggregation of displays
- Components are usually compiled binary modules that directly utilize hardware or software platforms (operating systems or application frameworks)



UI Component Rendering at the Composition Level

- Component does not handle UI rendering and display
- The composition is an aggregation of UI components that are usually represented by declarative language (markup language), which is interpreted and rendered by the platform



UI Components and Composition Styles

- Visually aggregated, container style
  - Iframe, frames
- Closed/embedded components style
  - Desktop
    - ActiveX Object
  - Web components
- Standard components
  - Generally conforms to a common platform model
- Declarative (markup) UI

Visual Aggregation

- Frames
  - <iframe>
- Easy and simple
- Limited interaction/integration with other components

Using iFrame

Closed/Embedded Components

- Desktop
  - ActiveX objects (Windows)
- Web
  - Image
  - ActiveX
  - Java applets
  - Flash
  - Silverlight
  - Specialized plug-ins: video, PDF, etc.

Standard Components

- Within platform
  - Portlets (Java)
  - Web parts (.Net)
  - Desktop sidebar gadgets
  - User controls (.Net)

- Cross platform
  - Web Services for Remote Portlets?
  - Web slice?
  - Declarative UI?

Declarative UI Components

- Usually uses markup language to represent user interface elements
  - May be treated as "presentational data"
- Popular languages
  - HTML/XHTML
  - XUL
  - XAML
- Examples
  - Microsoft Gadgets
  - Google Map
  - Google AJAX Search

UI Composition on the Web

- Server side
  - User controls
  - Screen scraping handling
  - Portlets
- Client side
  - Frames
  - Client side objects: requires downloading
  - AJAX

Another View of UI Integration Approaches

Table 1. Comparison of current user interface (UI) integration approaches.

	UI component model and external specification	Composition language	Communication style	Discovery and binding	Component visualization
Desktop UI components	Published, programmable API	General-purpose programming language	Centrally mediated and component-to-component communication could be supported	Static and dynamic binding	Component rendered
Browser plug-in components	Published, basic interface (startup configuration parameters)	Document markup code and JavaScript	Centrally mediated; very limited intercomponent communication via ad hoc JavaScript	Static binding	Component rendered
Web mashups	Hidden interface; published API	General-purpose programming language	Centrally mediated	Static binding	Typically markup based
Web portals and portlets	Standard interface based on public API; interface wrapped as a Web service	General-purpose programming language	Centrally mediated (interportlet communication under development)	Static and dynamic binding	Markup based

Florian Daniel, et al., "Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities Export", IEEE Internet Computing, Vol.11, No.3 (2007), pp. 59-66.