

SOS POLITICAL SCIENCE AND PUBLIC ADMINISTRATION

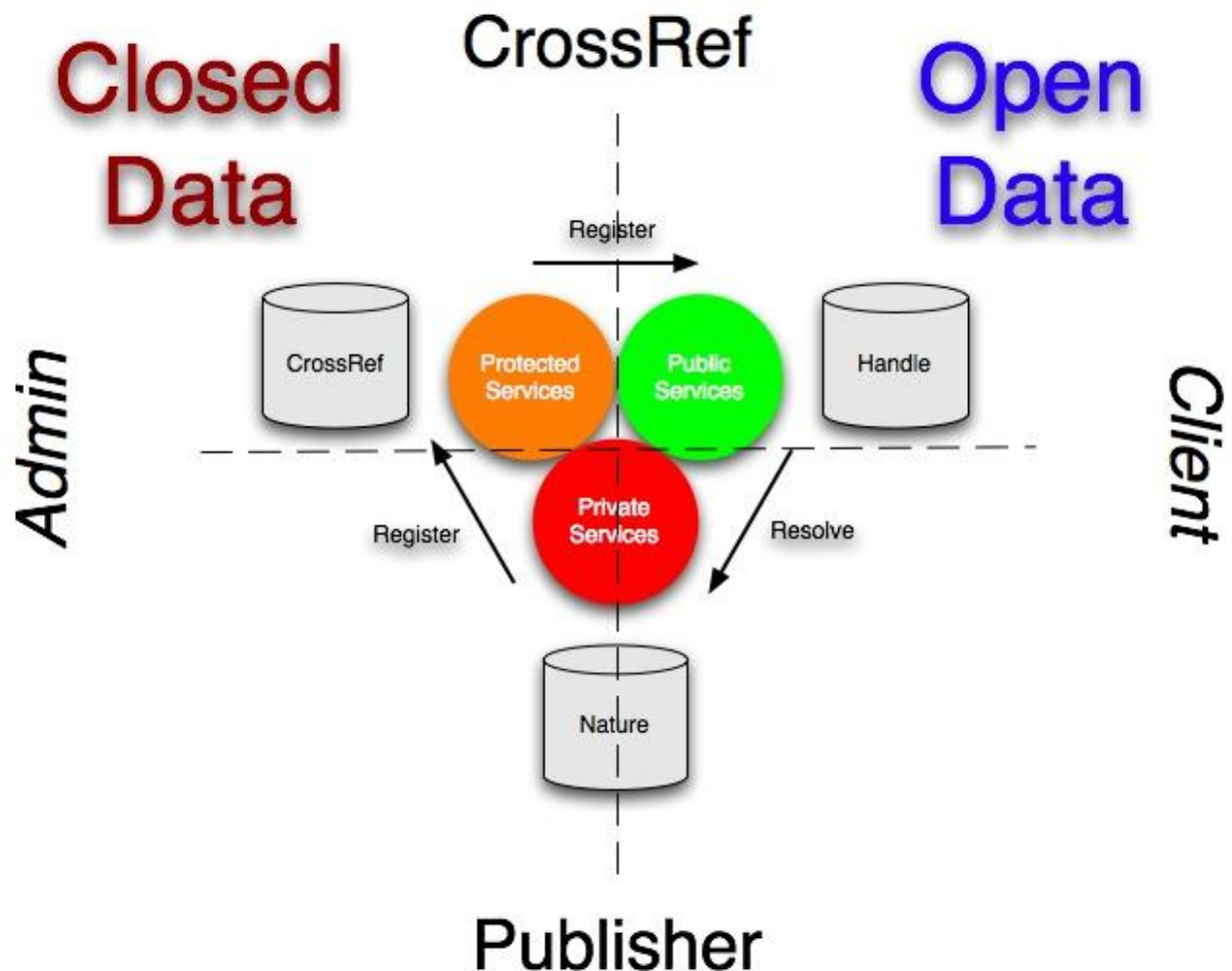
MBA FA 402

SUBJECT NAME: E- BUSINESS AND CYBER LAWS

UNIT-V

**TOPIC NAME: OPEN VS CLOSED ARCHITECTURE AND
CONTROLLED VS UNCONTROLLED CONTAINED**

OPEN VS CLOSED ARCHITECTURE:



OPEN ARCHITECTURE:

- Open architecture means an architecture whose specifications are public. This includes officially approved standards as well as privately designed

architectures whose specifications are made public by the designers. The opposite of open is closed or proprietary.

- The great advantage of open architectures is that anyone can design add-on products for it. By making architecture public, however, a manufacturer allows others to duplicate its product. Linux, for example, is considered open architecture because its source code is available to the public for free. In contrast, DOS, Windows, and the Macintosh architecture and operating system have been predominantly closed. Many lawsuits have been filed over the use of these architectures in clone machines. For example, IBM issued a Cease and Desist order, followed by a battery of lawsuits, when COMPAQ built its first computers.

CLOSED ARCHITECTURE:

- Closed-Architecture Systems. Loosed-architecture systems are systems for which access to vital software and hardware configuration is restricted.
- A system whose technical specifications are not made public. Such systems restrict third parties from building products that interface with or add enhancements to them.

OPEN ARCHITECTURE SYSTEM VS. CLOSED ARCHITECTURE SYSTEMS IN COMMERCIAL REAL ESTATE:

But how does this apply to the commercial real estate industry? Once they have created an acceptable roadmap for processes, managing brokers should then ask themselves several questions. The first, and the most critical issue is from a strictly financial perspective – should you create a customized system, or would it be better to utilize one of the real estate-centric products currently available in the market?

Regardless of what your ultimate decision on the matter is, industry best practices encourage the utilization of open architecture software over closed software architecture.

Simply put, an open architecture system is more favorable because it will allow a more efficient and cost-effective migration of your data with financial systems and CRM's systems that are utilized by your agents and administrative staff.

WHY YOU SHOULD CONSIDER AN OPEN ARCHITECTURE?

Open architecture is an application integration process that can be created, that only identifies and migrates transaction specific information from your agents' private customer relationship management software directly to the company's commission tracking platform.

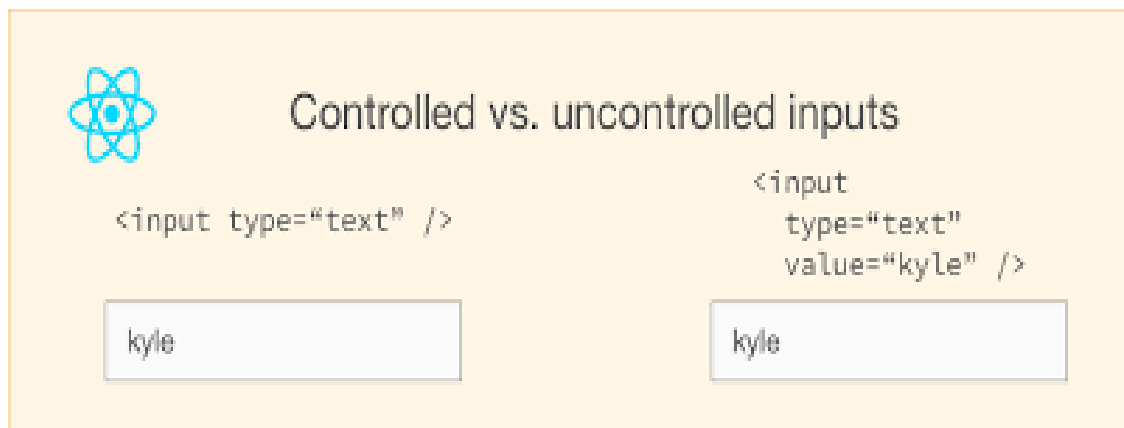
As a managing broker, it is crucial to understand that by utilizing an open architecture platform, you will enhance the ability to save your agents and administrative staff time by avoiding double data entry – which leads to an increase in productivity typically followed by increased profitability. Achieving these increases in productivity and profitability lead to an increase the acceptance and adoption of new technology in general – and that's half the battle.

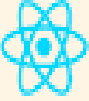
Any system you are considering should simplify your back office by managing receivables, commission split plans, invoicing and agent distributions all from one place. The system should streamline and remove the headaches, known as excel hell, by managing brokerage commissions and bookkeeping in a simplified and automated process.

The platform should provide simplified workflows and accounting journal entry automation for everything that happens after the deal closes. We suggest, an open architecture system because it will allow for an easier integration directly with other open systems, such as QuickBooks, general ledger accounting systems, and CRM's typically used by CRE agents and brokers.

CONTROLLED VS UNCONTROLLED CONTAINED:

CONTROLLED CONTAINED:



 Controlled vs. uncontrolled inputs

```
<input type="text" />
```

```
<input  
  type="text"  
  value="kyle" />
```

kyle

kyle

In a controlled component, the form data is handled by the state within the component. The state within the component serves as “the single source of truth” for the input elements that are rendered by the component.

THESE 2 POINTS TELL YOU THAT THIS IS A CONTROLLED COMPONENT:

- We don't need a form element on the page for the component to be a controlled component.
- When changes are made to any of the input elements that have an event handler, the handler is fired.
- The handler calls `setState ()` as you can see in line 9 above. This updates the state within the component.
- Updating the state in this way will not cause a re-render of the component and the changes made by the user will not be displayed in the UI.
- When a state update occurs via `setState ()`, it causes the component to re-render and the newly entered value is displayed in the element.
- The data flow is uni-directional from the component state to the input element.
- Working with controlled components can be a bit cumbersome. If there are a large number of input elements on the page each element requires setup with a `value` attribute and an event handler.

UNCONTROLLED COMPONENTS:

Uncontrolled components act more like traditional HTML form elements. The data for each input element is stored in the DOM, not in the component. Instead of

writing an event handler for all of your state updates, you use a *ref* to retrieve values from the DOM.

Wrap Up and Key Takeaways:

- Use controlled components whenever possible.
- Controlled components do not require a form element in order to be considered a controlled component.
- If a component has an input element that has a value attribute bound to state and an event handler to update said state, it is a controlled component.
- For pages that have a large number of input elements, it can be cumbersome to work with controlled components.
- Data flow is uni-directional in controlled components with the state within the component acting as the single source of truth.
- All state changes within a controlled component should be made via the `setState` function.
- Uncontrolled components store their data in the DOM like a traditional HTML input element.
- `React.createRef ()` is used to create instance variables within uncontrolled component constructors. These variables are then associated with input elements via the *ref* attribute.
- Refs cannot be used on functional components as there is no instance.
- Refs can be used inside functional components.