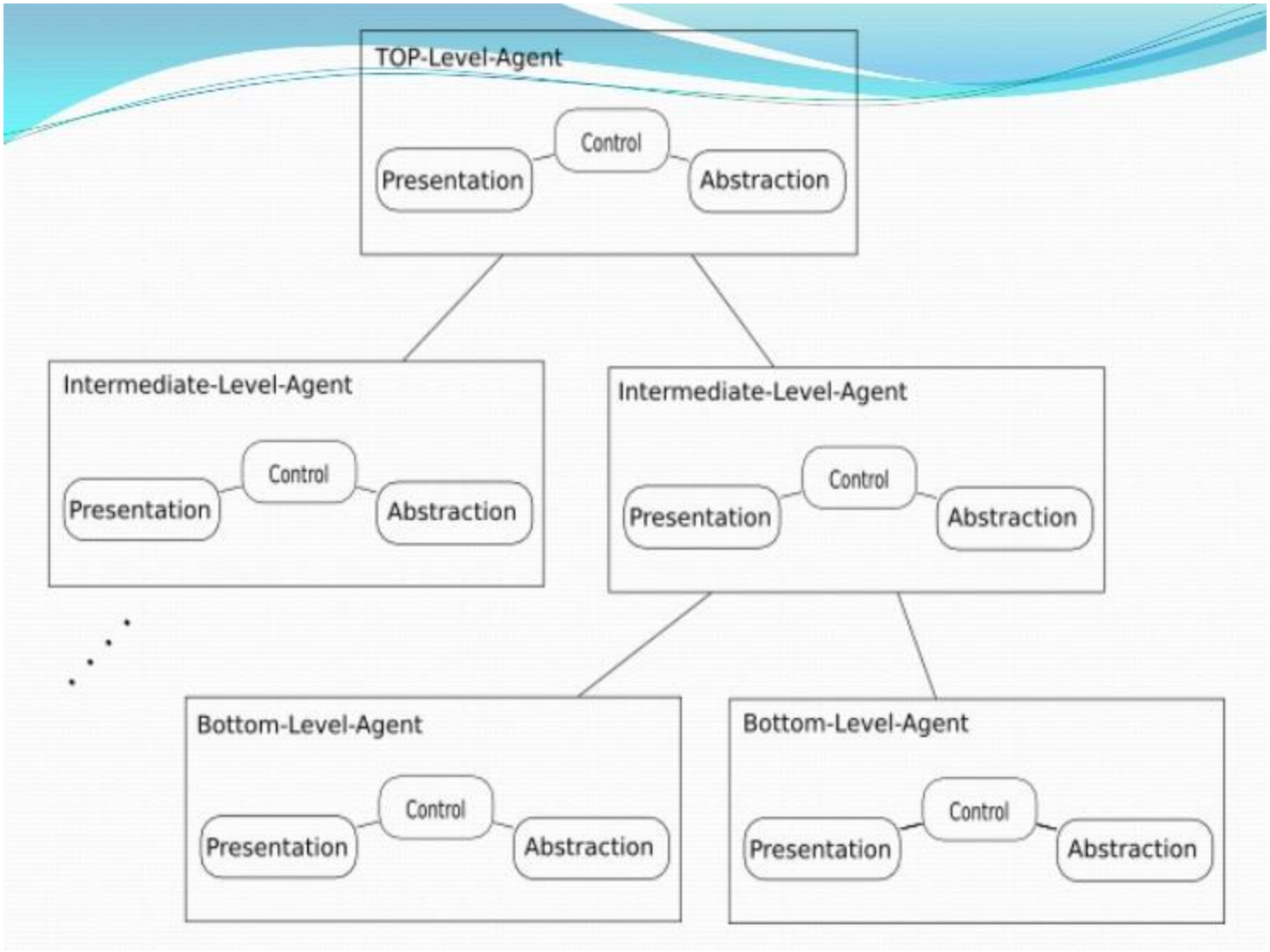
Software Design and Architecture

Presentation Abstraction Control (PAC) Architecture

Presentation-Abstraction-Control (PAC)

- The PAC architecture is similar to MVC but with some important differences.
- The PAC was developed from MVC to support the application requirement of multiple agents in addition to interactive requirements.
- In PAC, the system is decomposed into a hierarchy of many cooperating agents.

- Each agent has three components (Presentation, Abstraction, and Control).
- The Control component in each agent is in charge of communications with other agents.
- The top-level agent provides core data and business logics.
- The bottom level agents provide detailed specific data and presentations.
- A middle level agent may play a role of coordinator of low-level agents.
- There are no direct connections between Abstraction component and Presentation component in each agent.

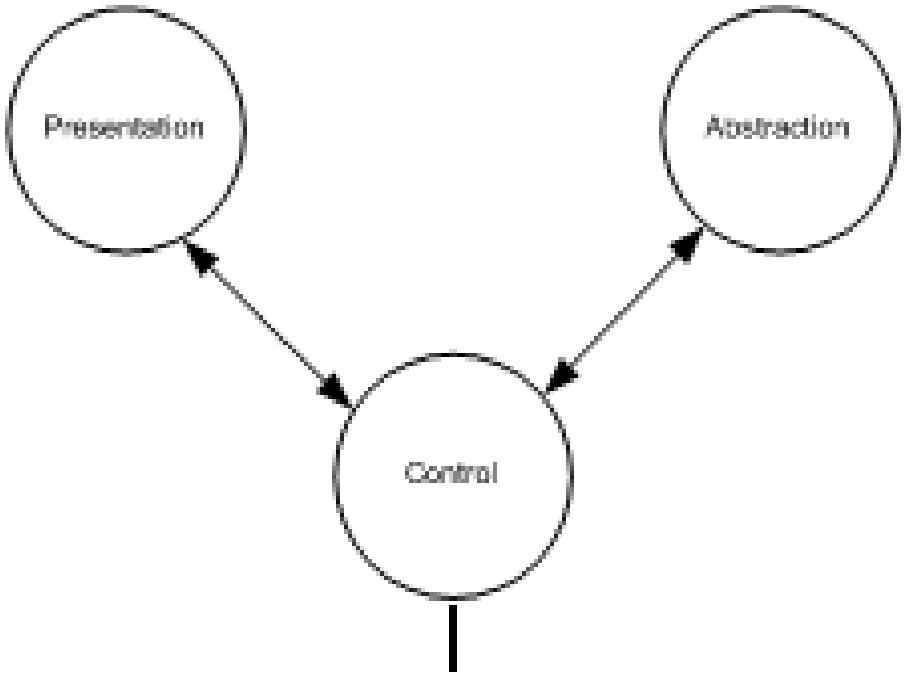

TOP-Level-Agent
Control
Presentation
Abstraction
Intermediate-Level-Agent
Control
Presentation
Abstraction
Intermediate-Level-Agent
Control
Presentation
Abstraction
Bottom-Level-Agent
Control
Presentation
Abstraction
Bottom-Level-Agent
Control
Presentation
Abstraction

- It is very suitable for any distributed system where all the agents are distantly distributed and each of them has its own functionalities with data and interactive interface.

- In such a system, all agents need to communicate with other agents in a well-structured manner.

- The PAC is also used in applications with rich GUI components where each of them keeps its own current data and interactive interface and needs to communicate with other components.

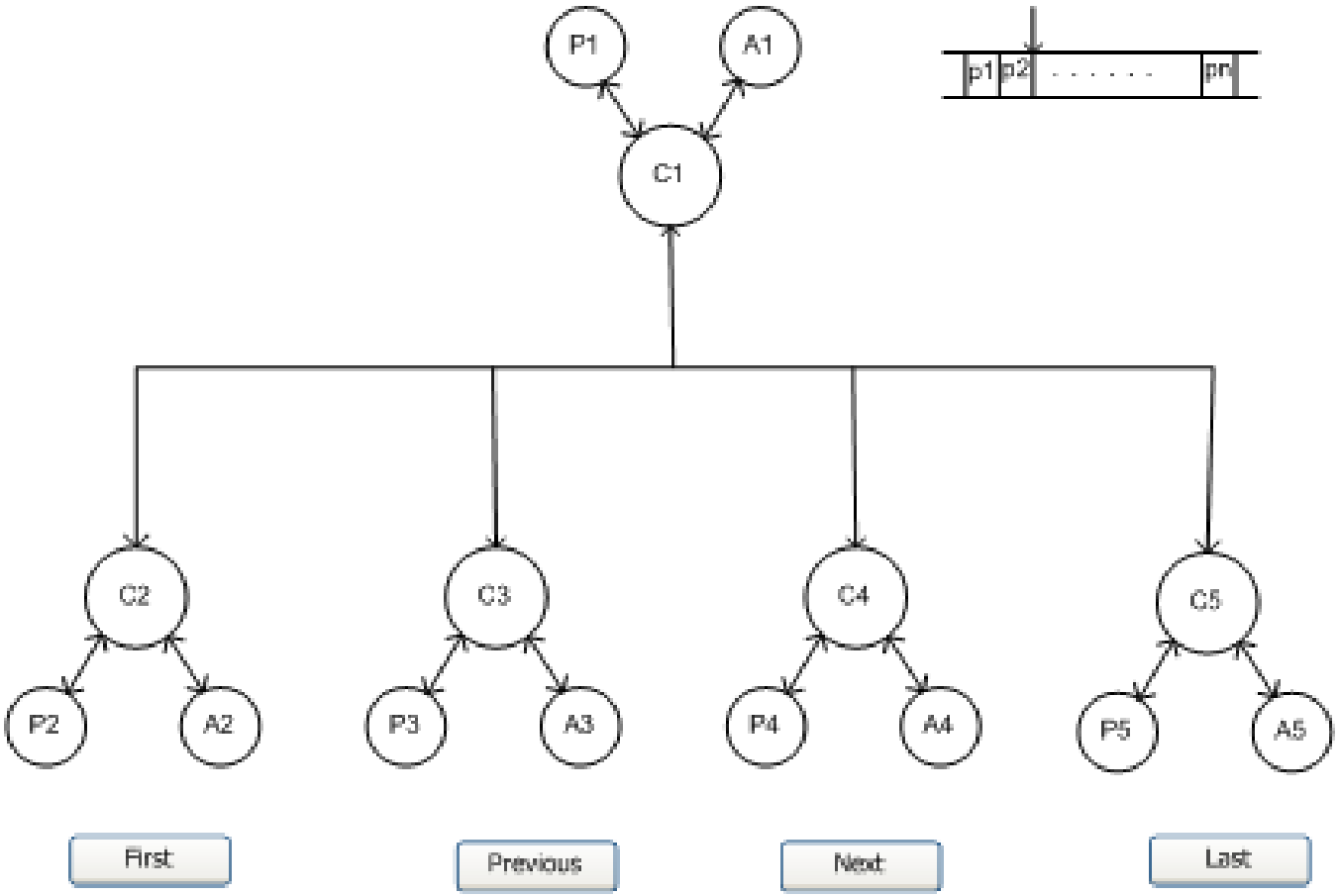
- The Control component is a mediator between the Presentation component and the Abstraction component within the agent, and also a bridge between the agent itself and other agents as well.
- The Presentation component and the Abstraction component are loosely coupled.
- The Presentation component is responsible for both data input and data output in GUI interfaces where the data come from the Abstraction component.
- The Abstraction component is responsible for providing logical data concepts and services and encapsulating all detailed data manipulation.



A single Agent in PAC

Example of PAC

- We design a desktop presentation application that can browse and display presentation pages in a graphic document one at a time.
- The application provides four buttons to show first page, last page, previous page and next page.
- The agent A1 has a connection to the data repository which is the document file.



Multiple agents in PAC

- Assume that the current page is the second to last page in the document at this time.

- If the user clicks on the next button, the control agent C4 informs agent P4 to update its presentation, in this case, it also hides the next button since there is no next page after last page.

- Agent C4 also informs agent A4 to update the data on next button.

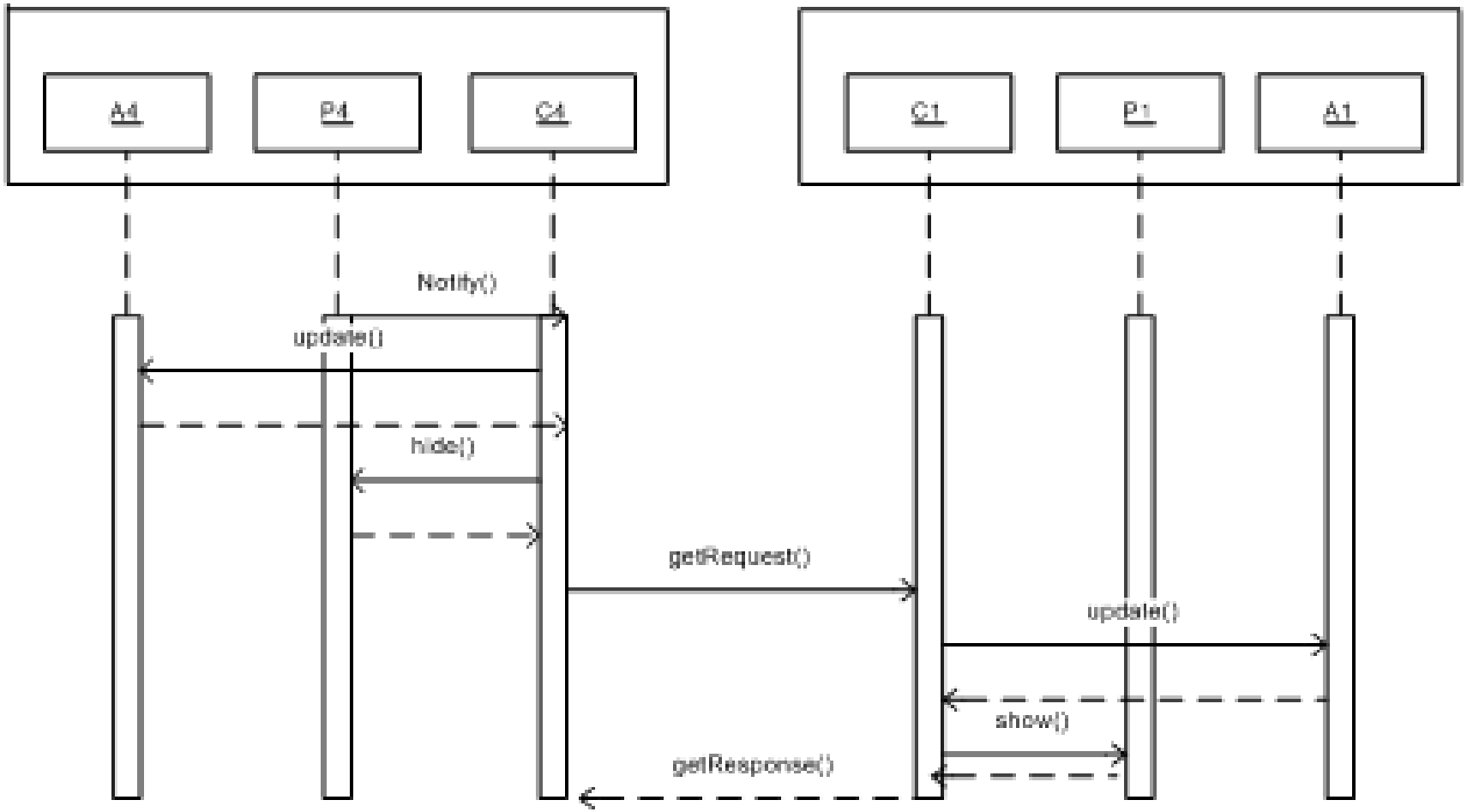
- After C4 handles all local processing, it contacts its parent agent, C1, to let it take over.

- After C1 gets the message from C4, it tells A1 to move the next page, which is the last page in the document, and then asks P1 to display that page.

- C1 also informs C5 to hide the last button since the current page is the last page (or let the last button stay based on the requirement specification).

- We can see that all the agents communicate via the controls.

- Since PAC2, PAC3, PAC4, and PAC5 are all buttons.



Sequence Diagram of PAC Architecture

- The sequence diagram shows the interaction sequence in the example we discussed above.

- When the next button is pressed to display the last page in the document PAC4 and PAC1 react as follows:

❖ P4 informs C4 that the "next" button was pressed,C4 sends update to A4

❖ C4 informs P4 to update its presentation or shape, C4 contacts C1 (a top level agent).

❖ C1 sends update to A1 to move the pointer to next (last page) C1 instructs P1 to display the last page.

Applicable domain of the PAC architecture

- Suitable for an interactive system where the system can be divided into many cooperating agents in a hierarchical manner.
- Each agent has its own specific assigned job.
- Suitable when the coupling among the agents is expected to be loose so that changes on an agent does not affect others.

Benefits

- Support of multi-tasking and multi-viewing.
- Support agent reusability and extensibility.
- Easy to plug-in new agent or replace an existing one.
- Support for concurrency where multiple agents are running in parallel in different threads or different devices or computers.

Limitations

- Overhead due to the control bridge between presentation and abstraction and the communication of controls among agents.
- Difficult to determine the right number of the agents due to the loose coupling and high independence between agents
- Complete separation of presentation and abstraction by control in each agent generate development complexity since communications between agents only take place between the controls of agents.

Summary

- Introduce PAC Architecture
- Applicable domain of PAC
- Benefits and Limitations of PAC