Smart Kanban Workflow Manager

A Qt6 / C++17 desktop application demonstrating maintainable OO design and design patterns

CMPE 202
Team: Model Builders
Alekya Gudise
Michael Kennedy

Presentation roadmap

Aligned with the class presentation rubric: application overview, architecture, design principles, design patterns, and demo.

1	**What the app does**	Create, move, filter, delete, undo, redo, and track work items.
2	**Architecture and UML**	Four layers: UI, service, pattern support, and domain model.
3	**Design principles**	SRP, OCP, dependency inversion, encapsulation, and separation of concerns.
4	**Design patterns**	Command, Observer, State, Strategy, Factory Method, and Chain of Responsibility.
5	**Demo plan**	Show visible features and connect each one back to the design.

Application overview

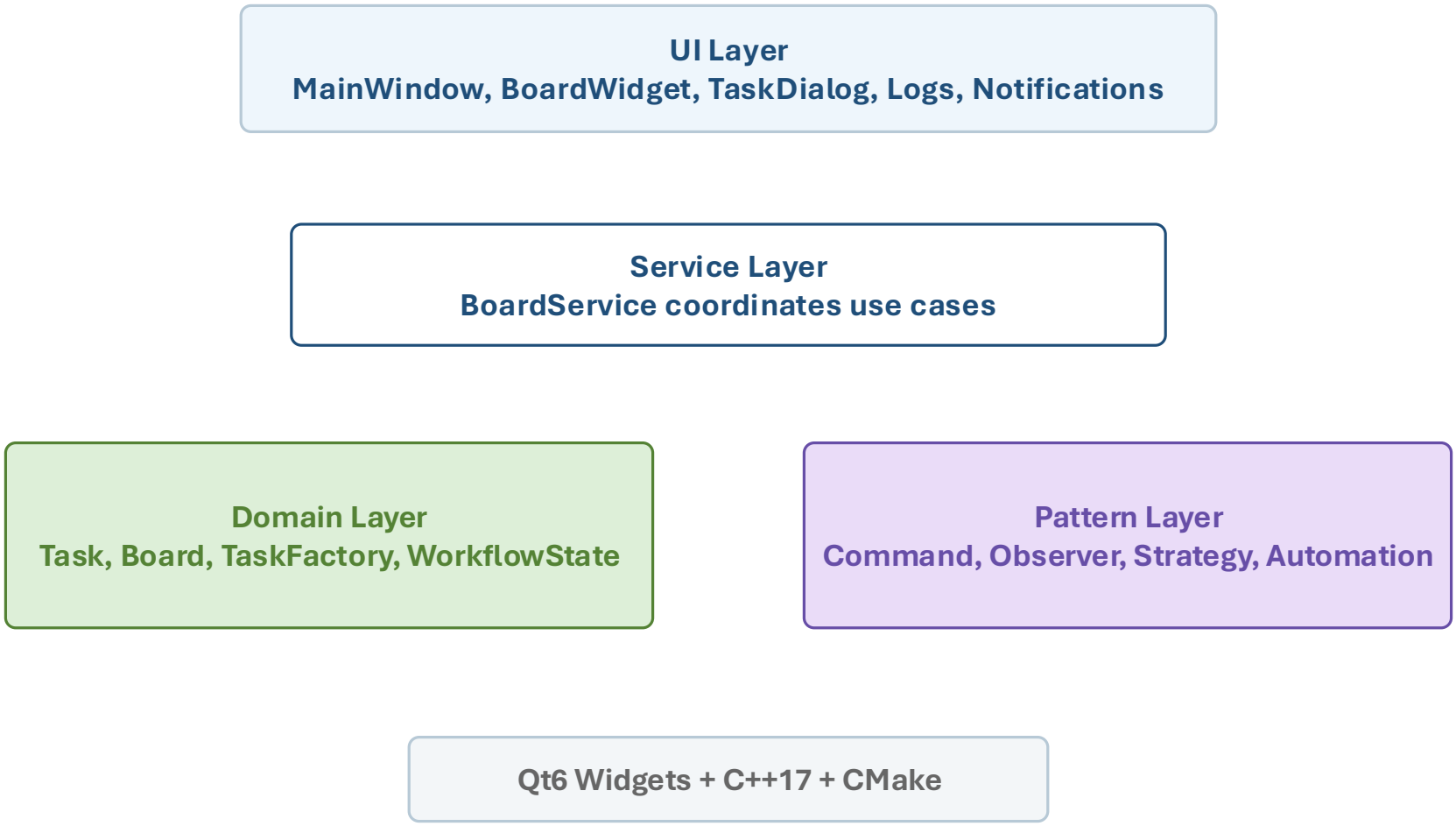
A desktop Kanban workflow app with visible features tied directly to design patterns.

Core user actions

- Add tasks
- Drag across workflow
- Filter board
- Undo/redo
- View activity log
- Get automation notifications

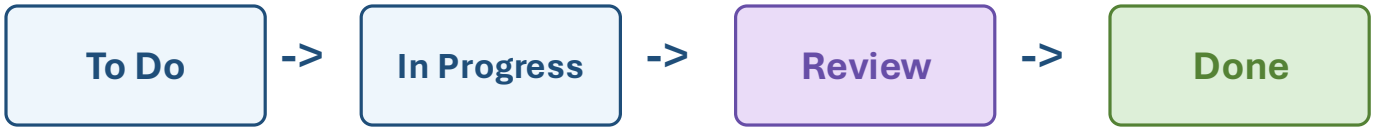
High-level architecture

The UI talks to BoardService; domain objects and pattern components stay behind clear interfaces.



Workflow model: states guard movement

The State pattern keeps legal transitions in the domain layer instead of the drag/drop UI.



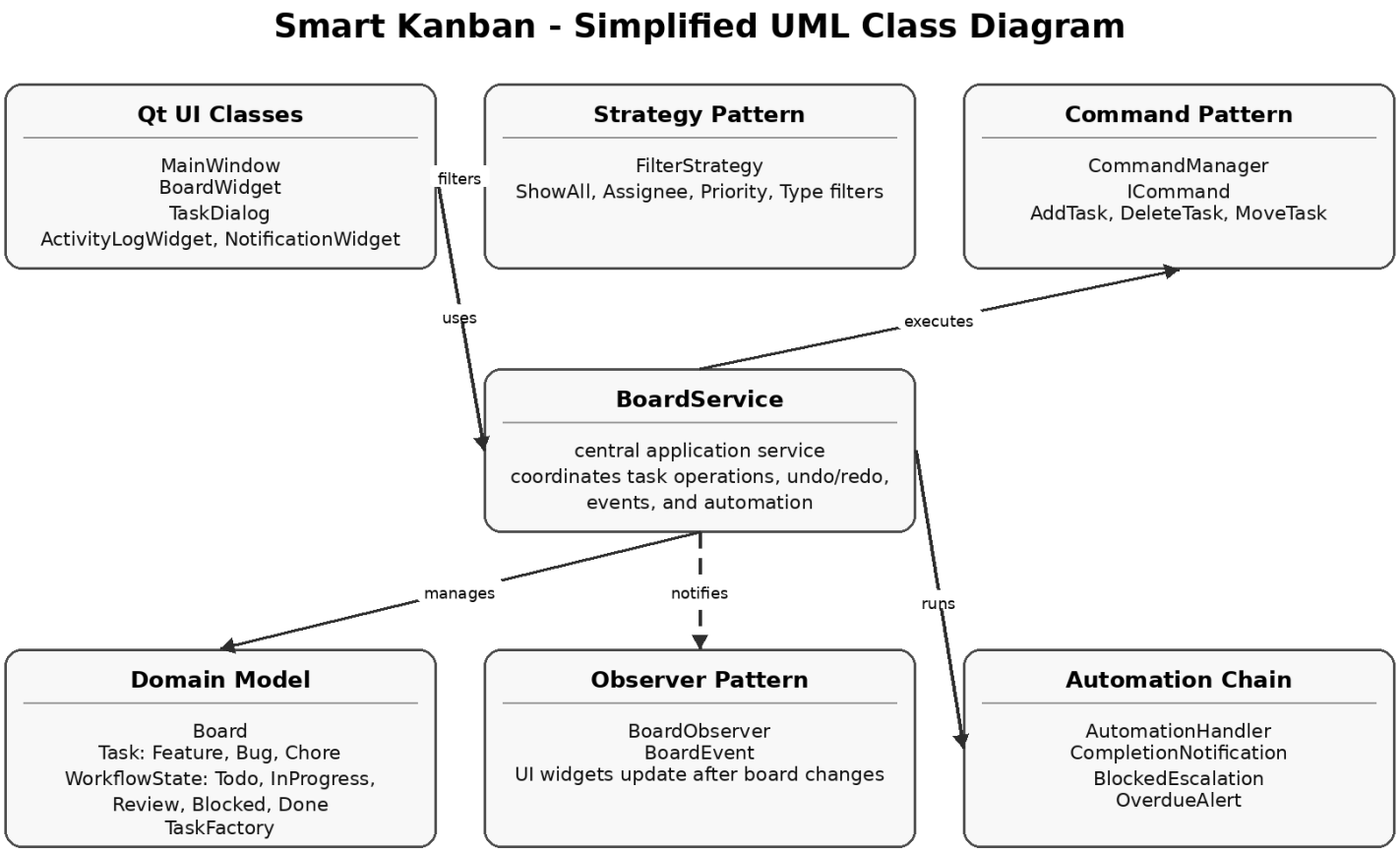
Blocked can be entered from active stages and returned to In Progress

Blocked

Invalid drops are rejected by Task::canTransitionTo()

UML class diagram

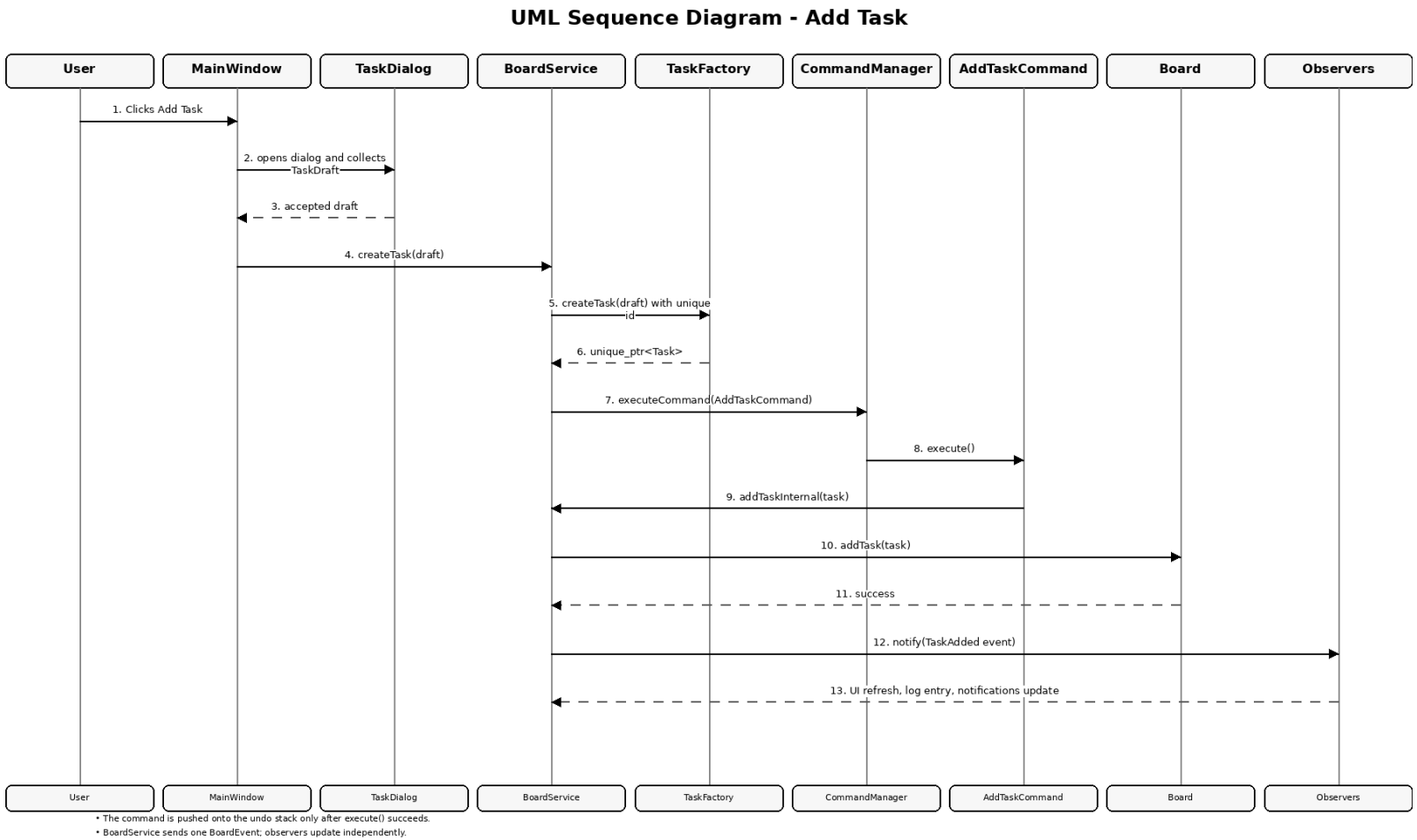
Major class groups and pattern roles in the project.



This diagram groups concrete subclasses under their parent pattern or domain role to reduce visual clutter.

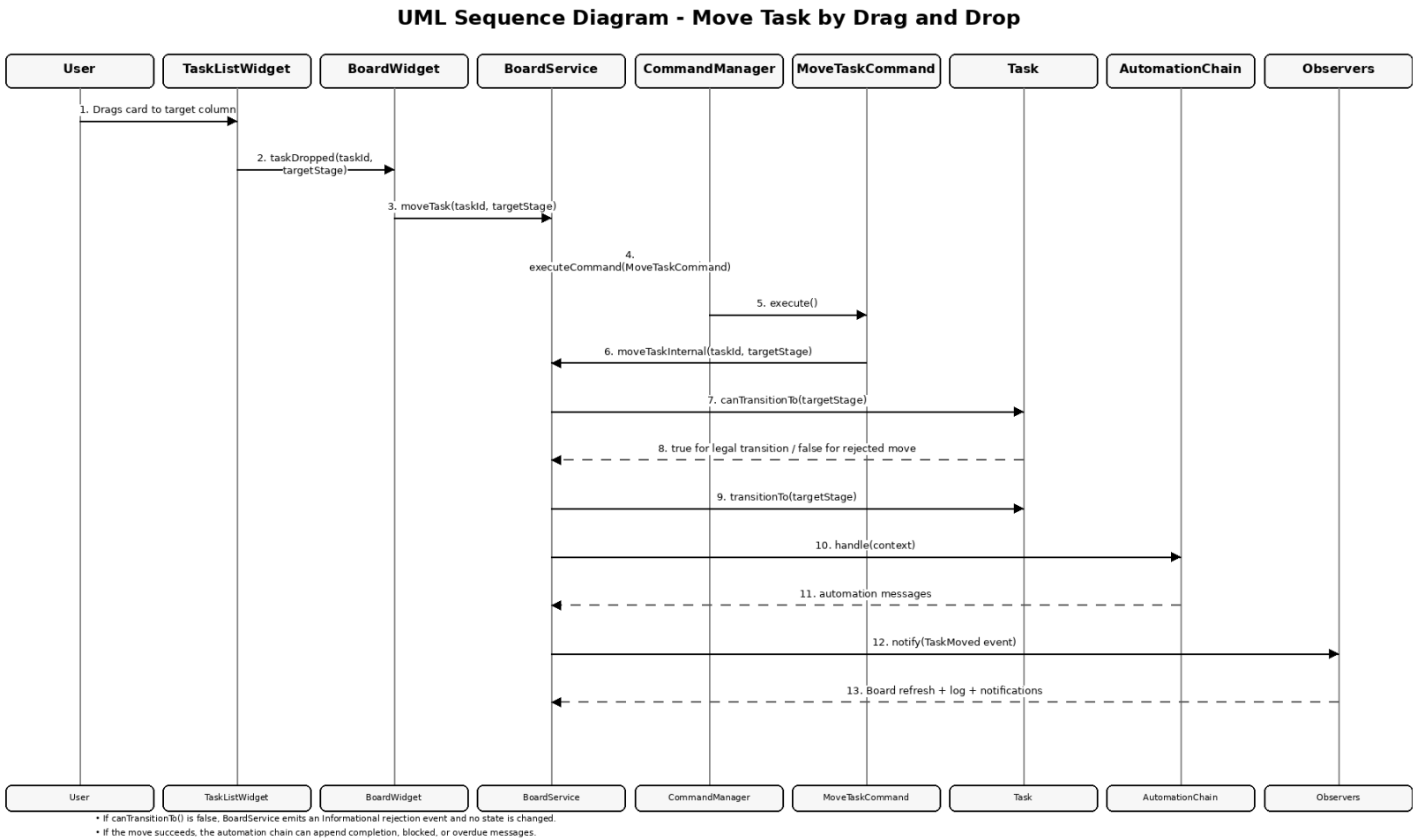
Add task sequence

Factory, Command, and Observer work together when a task is created.



Move task sequence

Drag/drop movement is validated by State and then broadcast through Observer.



Design principles used

The codebase is organized to keep classes cohesive and loosely coupled.

Single Responsibility

Board stores tasks; TaskFactory creates tasks; CommandManager manages undo/redo stacks.

Open/Closed

Add a filter, command, automation rule, or task type through new classes instead of editing core loops.

Dependency Inversion

UI and services depend on interfaces like BoardObserver, ICommand, and FilterStrategy.

Encapsulation

Board data and internal mutations are hidden behind BoardService and command-controlled APIs.

Separation of Concerns

UI detects user actions; service coordinates use cases; domain enforces rules.

Design patterns mapped to features

Each pattern supports a feature the class can see in the demo.

Pattern	Visible feature	How it helps
Command	Undo / redo	Operations are objects with execute and undo behavior.
Observer	Auto-refresh UI	BoardService notifies widgets through BoardObserver.
State	Legal workflow moves	Each WorkflowState controls allowed transitions.
Strategy	Runtime filters	BoardWidget swaps FilterStrategy objects.
Factory Method	Typed tasks	TaskFactory builds Feature, Bug, or Chore tasks.
Chain of Responsibility	Automation messages	Handlers independently react to task moves.

Demo script

A concise flow to show the app and connect behavior back to the design.

Launch	Show seeded tasks in the five workflow columns.
Add task	Open TaskDialog; Factory creates a typed task; Command records the action.
Move task	Drag/drop; State validates the move; Observer refreshes widgets.
Automation	Move to Done or Blocked to trigger chain handlers.
Filter	Switch to Assignee, Priority, or Type strategy.
Undo/redo	Reverse and replay add/delete/move commands.

Build: cmake -S . -B build -DCMAKE_PREFIX_PATH="/path/to/Qt/6.x.x/lib/cmake" | Run: ./build/SmartKanban

Conclusion

The project demonstrates patterns as real product features, not just extra classes.

What the design gives us

Maintainable separation between UI, service, domain, and pattern logic

Extensible filters, automations, task types, commands, and workflow stages

Visible behavior backed by reusable OO designs

Future extensions

- Persistence to disk
- More automation rules
- Additional task types
- Custom workflows
- Exportable reports

Questions?