# PAVE

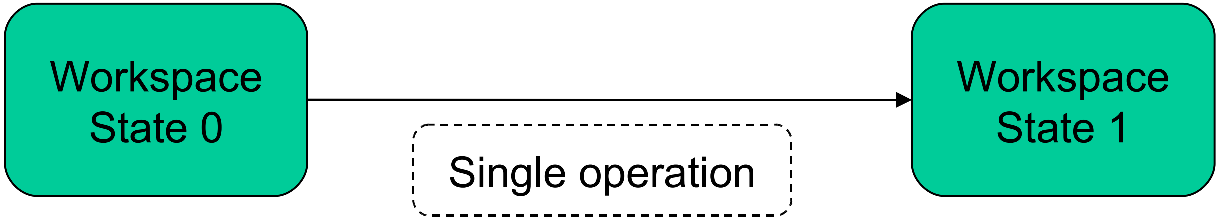
## Framework for executing patterns

Dimitar Giormov, SAP Labs Bulgaria

# What is “pattern”?

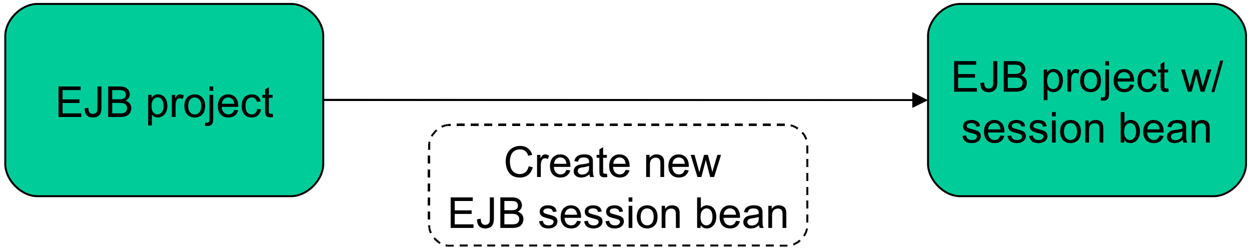
- Patterns are “well defined templates”
- Patterns change the workspace’s state by applying one or more operations
- Patterns can be triggered only for resources with the required contextual environment

# Single operation



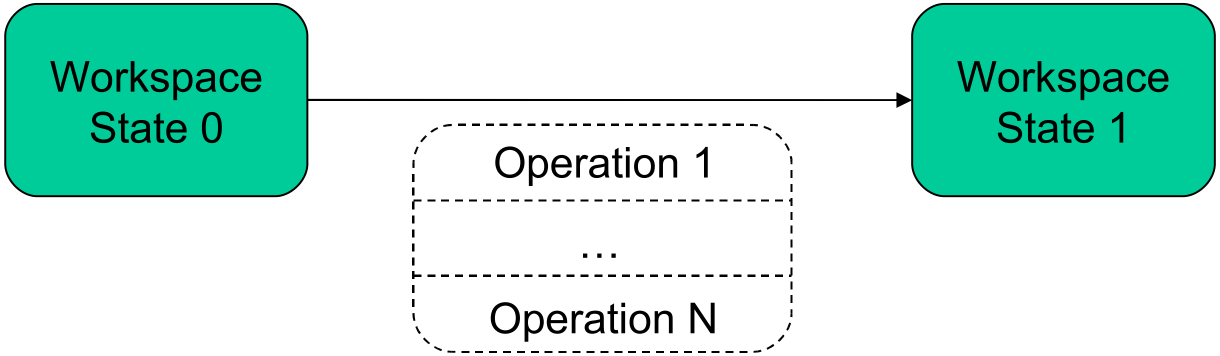
- Changes the workspace’s state by executing a single operation

# Single operation – example



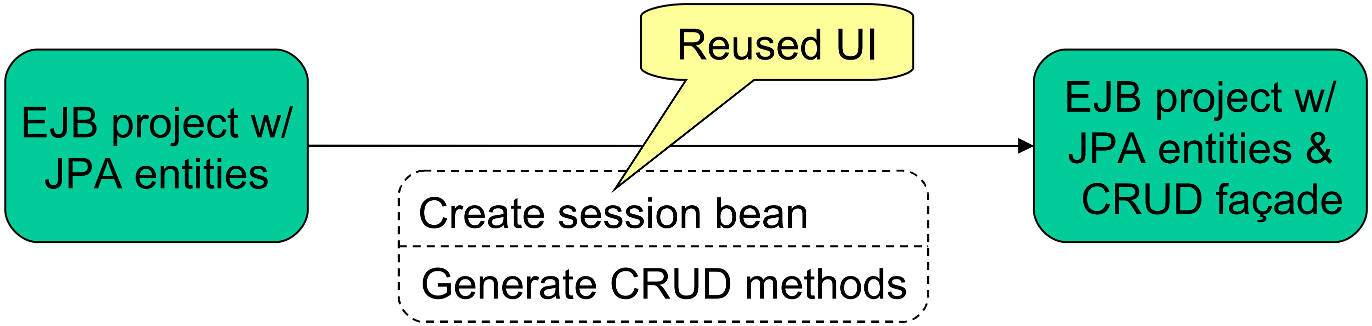
- Adds a new EJB session bean to an EJB project

# Pattern



- Changes the workspace’s state by executing a sequence of operations
- The output of the previous operation might be the input of the next operation
- Existing operations and/or their UI could be reused by the pattern
- Assigned only to workspace resources with the required contextual environment

# Pattern – example



EJB project w/ JPA entities

Reused UI

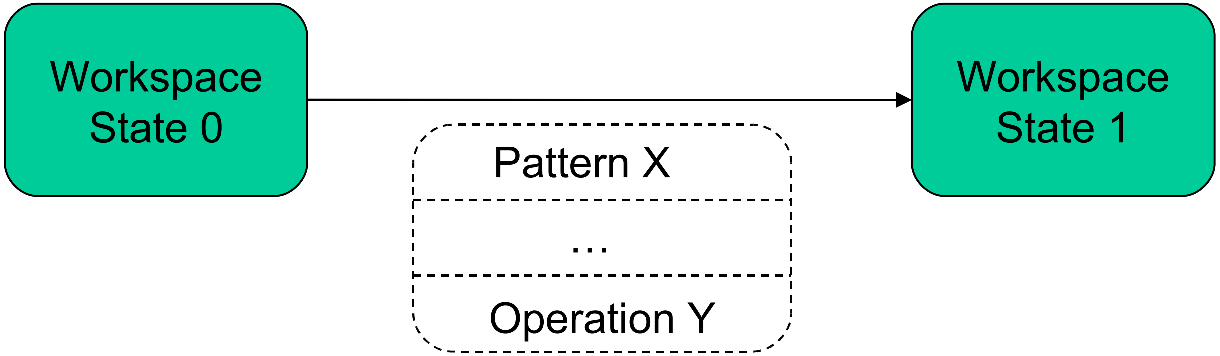
Create session bean

Generate CRUD methods

EJB project w/ JPA entities & CRUD façade

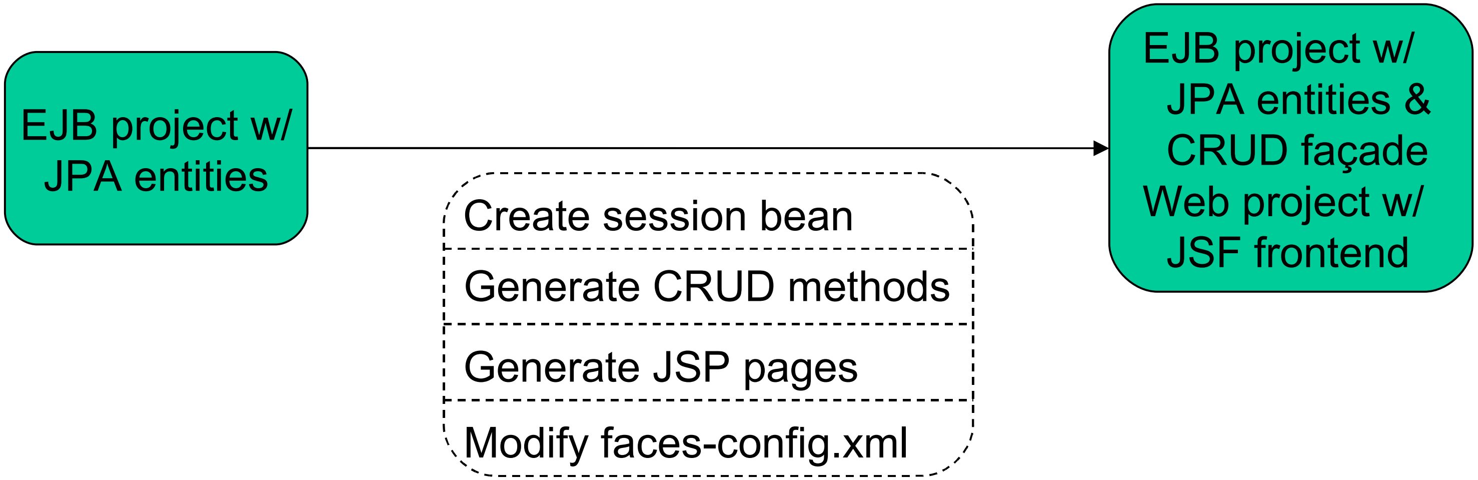
- Create CRUD façades from JPA entities
- Reuses the wizard pages and operations from the EJB Session Bean wizard
- Can be called only for projects with JPA facet

# Extended pattern



- Changes the workspace’s state by executing a pattern that extend an existing pattern with other operations and/or patterns

# Extended pattern – example



EJB project w/ JPA entities

Create session bean

Generate CRUD methods

Generate JSP pages

Modify faces-config.xml

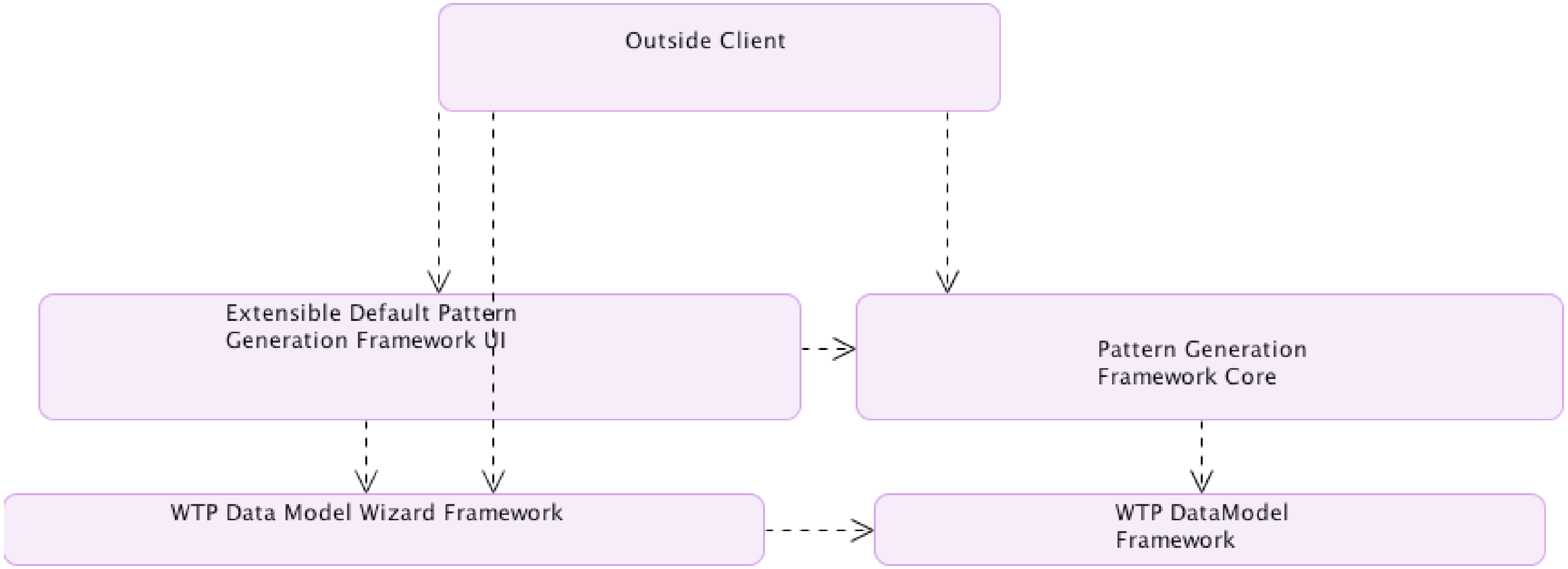
EJB project w/ JPA entities & CRUD façade
Web project w/ JSF frontend

- Create the complete skeleton of a Java EE application from JPA entities
- Extends the “CRUD façade” pattern with two more operations

# Project goals

- Make generic extensible, easy to use atomic operation based framework
- Framework built on top of WTP Data Model Framework
  - Data Model – for data collection
  - Data Model Operations – undoable operations
- Patterns contributed through extension points
- Each Pattern consists of one or more operations
- Standalone operations that can be reused in different patterns or outside the framework.
- Execution of sequence of operations
- Extending patterns
- Business logic separated from the UI
- Extensible default pattern wizard

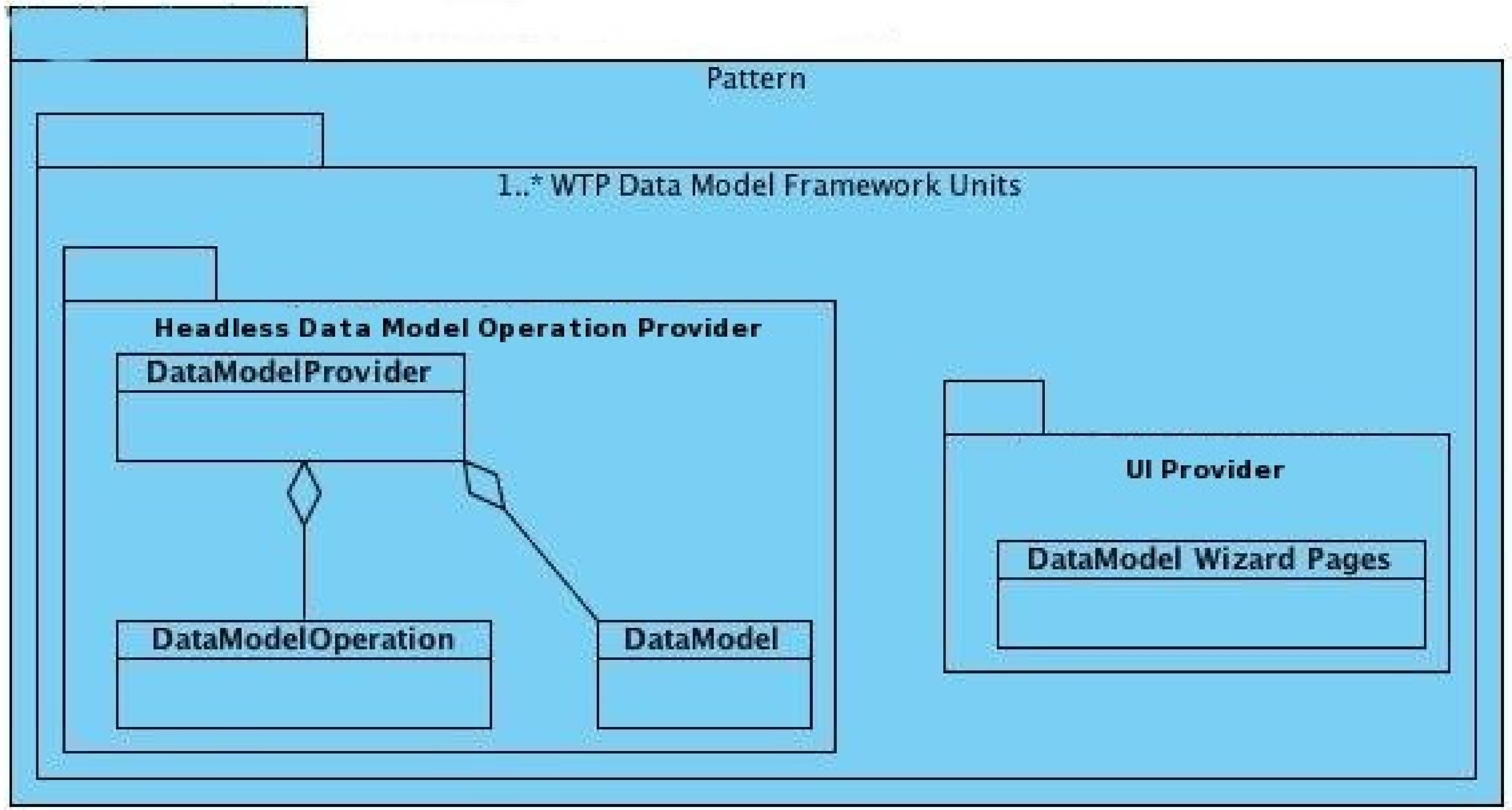
# Architectural outlook


Outside Client
Extensible Default Pattern Generation Framework UI
Pattern Generation Framework Core
WTP Data Model Wizard Framework
WTP DataModel Framework

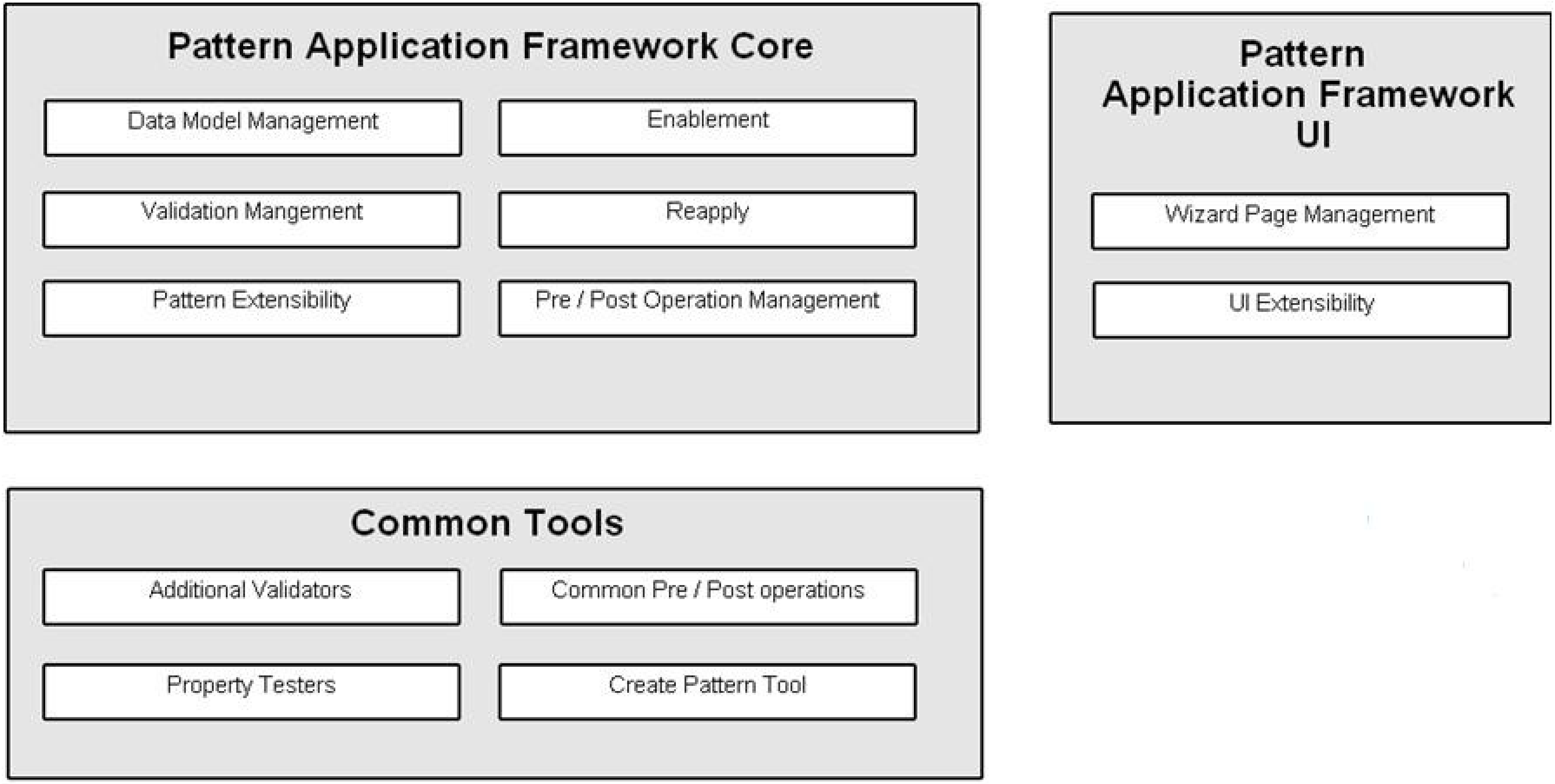
- Framework Core based on WTP Data Model Framework – it will care for pattern execution
- Extendable / replaceable default pattern wizard – contributes first page on which the appropriate patterns can be chosen.
- Outside client could create its own UI using standard WTP Data Model Wizard Framework and Framework Core or by extending the default pattern wizard.
- Headless execution (drag and drop)

# Relationship to WTP Data Model Framework

- PAVE is heavily based in Data Model Framework
- Data Model Framework operations (units) can be used directly in PAVE
- The units can be combined in sequences – patterns



# Components



## Pattern Application Framework Core

- Data Model Management
- Enablement
- Validation Mangement
- Reapply
- Pattern Extensibility
- Pre / Post Operation Management

## Pattern Application Framework UI

- Wizard Page Management
- UI Extensibility

## Common Tools

- Additional Validators
- Common Pre / Post operations
- Property Testers
- Create Pattern Tool

# PAVE features explained

- Core
  - Enablement – using org.eclipse.expressions define on which object the pattern is applicable.
  - Data Model Synchronisation – mechanism for sharing data model properties without modifying the original data model providers.
  - Additional validation added to the pattern (not the Model Providers).
  - Validation Override – override validation in used Data Model Providers
  - Pre / Post operations adaptation in patterns
  - Patterns are extensible
  - Reapply concept.
- UI
  - Wizard Page Management based on patterns
  - UI extensibility (roadmap).
- Common Tools
  - Common validators / property testers / operations
  - Pattern for creating patterns.

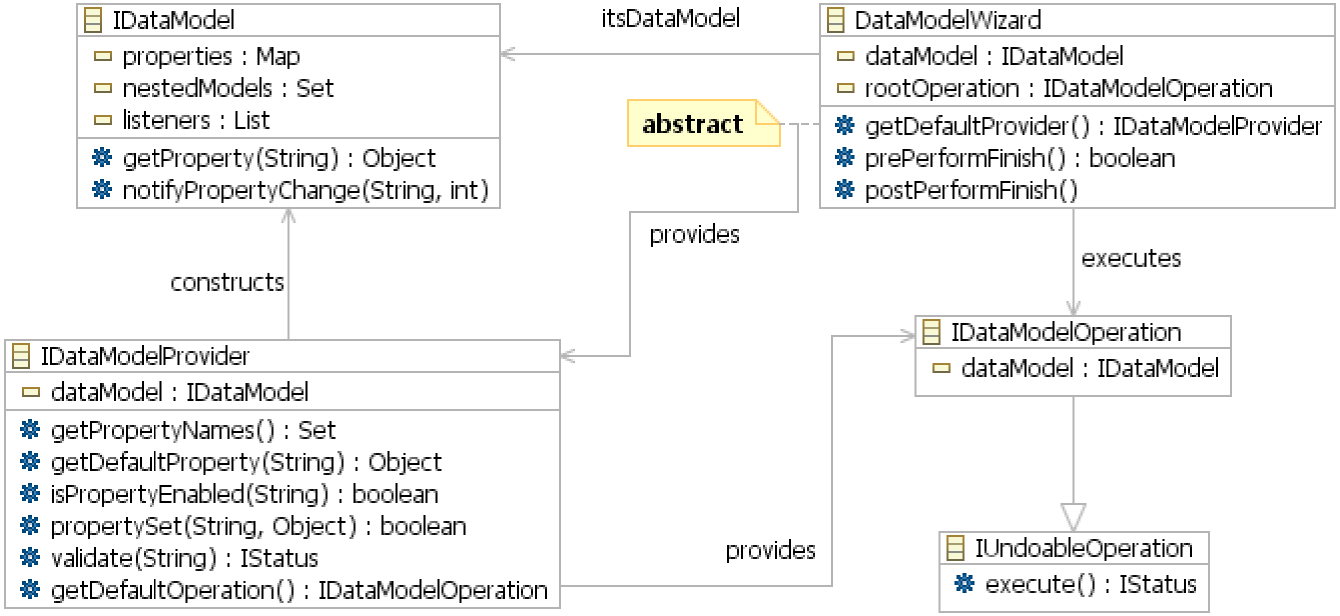
# Java EE Patterns

- Session CRUD façade pattern
  - Enabled on JPA entity
  - Pattern consists of 2 operations.
    - Create Session Bean operation in ejb.tools project
    - Customize Session Bean operation, which generates CRUD methods and queries (static or dynamic)
  - Pattern will elaborate on Entity Relations and will generate one Session Bean CRUD Façade for them.
- Java EE application pattern
  - Extends CRUD Façade pattern (enabled again on entity and has the 2 operations as a starting point)
  - Generates Manage Beans, JSF Pages, Creates Relations in faces configuration file

# Appendix

# WTP Data Model class diagram

- Data Model Providers create the Data Model instances and supply associated default Data Model Operations
- Data Model is property based model.



IDataModel
properties : Map
nestedModels : Set
listeners : List
getProperty(String) : Object
notifyPropertyChange(String, int)

itsDataModel

DataModelWizard
dataModel : IDataModel
rootOperation : IDataModelOperation
getDefaultProvider() : IDataModelProvider
prePerformFinish() : boolean
postPerformFinish()

abstract

provides

executes

constructs

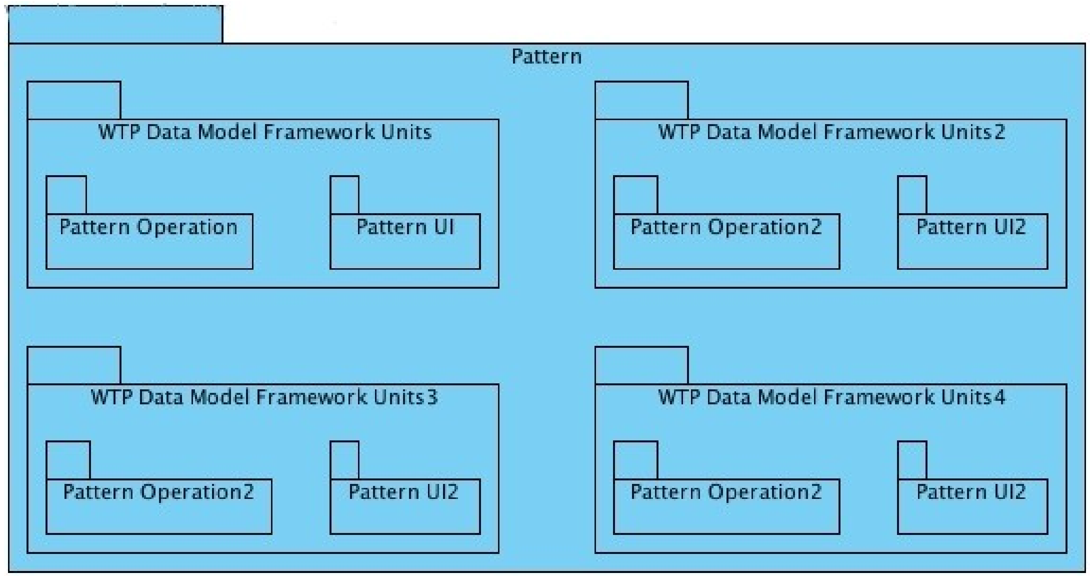
IDataModelProvider
dataModel : IDataModel
getPropertyNames() : Set
getDefaultProperty(String) : Object
isPropertyEnabled(String) : boolean
propertySet(String, Object) : boolean
validate(String) : IStatus
getDefaultOperation() : IDataModelOperation

IDataModelOperation
dataModel : IDataModel

provides

IUndoableOperation
execute() : IStatus

# Template Framework



Pattern

WTP Data Model Framework Units

Pattern Operation

Pattern UI

WTP Data Model Framework Units2

Pattern Operation2

Pattern UI2

WTP Data Model Framework Units3

Pattern Operation2

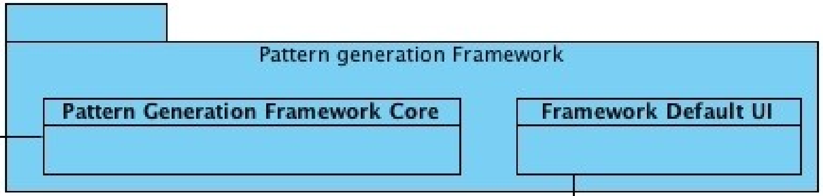
Pattern UI2

WTP Data Model Framework Units4

Pattern Operation2

Pattern UI2

Pattern Greneration Framework Core is based on the same WTP Data Model Framework, It will gather all "generators" from the pattern and it will nest their models. Nesting the models allows communication between units, all units have the model with all the data.



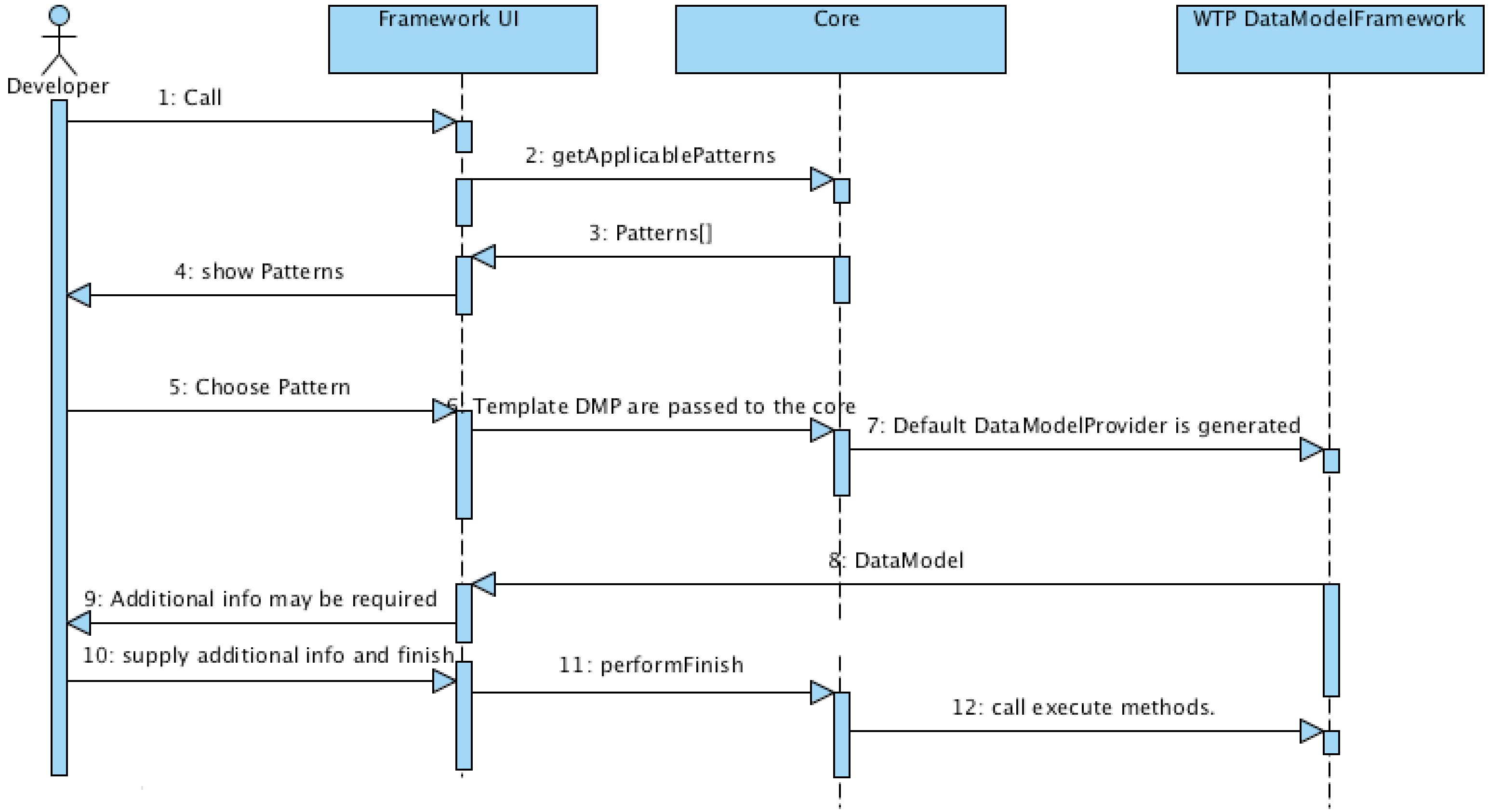
Pattern generation Framework

**Pattern Generation Framework Core**

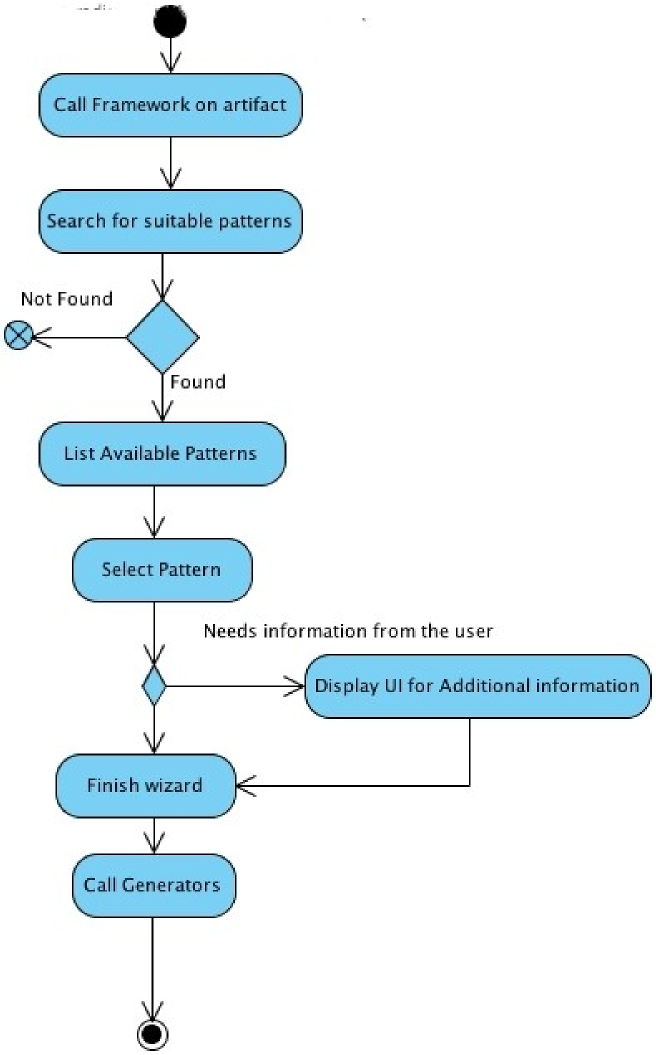
**Framework Default UI**

Default UI will combine the ui defined by the units
in a Default Wizard, however. The UI is entirely
decoupled and can be substituted.

# Action sequence


Developer
Framework UI
Core
WTP DataModelFramework
1: Call
2: getApplicablePatterns
3: Patterns[]
4: show Patterns
5: Choose Pattern
6: Template DMP are passed to the core
7: Default DataModelProvider is generated
8: DataModel
9: Additional info may be required
10: supply additional info and finish
11: performFinish
12: call execute methods.

# Pattern Generation Framework Action Sequence



Call Framework on artifact

Search for suitable patterns

Not Found

Found

List Available Patterns

Select Pattern

Needs information from the user

Display UI for Additional information

Finish wizard

Call Generators