Meta-Architecture Document

Lori Gildersleeve
Jennifer Martin
Cody Nelson

VER. 2014-3

Table of Contents

Architectural Vision 3

Principles 3

Styles 4

Patterns & Interconnection Mechanisms 7

Pattern [State-Logic-Display (Three-Tier)] 7

Interconnection Mechanisms 7

Philosophies & Preferences 9

Guidelines & policies 9

Additional Information 9

Meta-Architecture Document

[Gildersleeve, Martin, Nelson]

ARCHITECTURAL VISION

The architecture will be the guide by which the programmers will use to develop the Sudoku game. The vision is to create a concise architectural design clearly separating and defining each process of the Sudoku application thereby permitting the programmers the ability to study the architecture and design accordingly. The architecture will attempt to take future development considerations into account allowing for potential scalability and adaptation needs.

1. Clearly separates each process of sudoku app
2. allows devs to ~~study~~ follow design
3. allows for future scalability development
4. allows for future adaptation

PRINCIPLES

Principle Name	**Separation of Concerns**
Description	Creation of independent parts by subdivision of a problem
Rationale/Benefits	Aides in encapsulation of functionality and defining classes
Implications	User interface, business rules, and data access can be kept independent of one another
Counterargument	Parts of the problem could not be kept independent without significant trade-off of performance, functionality or appearance.

Any principles related to Scaling - ch12?
Ease of understanding?
Adaptation ch12, 14?

STYLES

Style [Object-Oriented]

Summary Description: States and functions encapsulated by objects

Components: Objects (AKA instances of a class)

Connectors: Method invocation (procedure calls to manipulate state)

Data Elements: Arguments to methods

Topology: Can vary arbitrarily; components may share data and interface functions through inheritance hierarchies

Additional Constraints: Commonly shared memory, single threaded

Qualities Yielded: Integrity of data operations; data manipulated only by appropriate functions; abstraction implementation details hidden

Typical Uses: Applications where the designer wants a close correlation between entities in the physical world and entities in the program; pedagogy; applications involving compiles, dynamic data structures

Cautions: Use in distributed applications requires extensive middleware to provide access to remote objects; relatively inefficient for high-performance applications with large, regular numeric data structures; lack of additional structuring principles can result in highly complex applications.

Relation to Programming Languages: Java, C++

Rationale: Allows the use of Java, which is familiar to all team members; application idea (Sudoku) can be encapsulated simply via object-oriented techniques (classes).

Style [Blackboard]

Summary Description: Global data repository (blackboard) through which independent programs communicate

Components: Blackboard and independent programs (AKA knowledge sources)

Connectors: Database query, method call, or direct memory reference can access the blackboard

Data Elements: Blackboard stored data

Topology: Blackboard is at the center of this Star Topology

Additional Constraints: Programs can ask blackboard if there is new data OR blackboard can push notifications of new data/updates

Qualities Yielded: Complex problems do not have to have preplanned solutions

Typical Uses: Artificial Intelligence experimental solutions to problems shared memory applications

Cautions: Not ideal if: a well-defined solution to a problem exists, complex regulations are needed between the independent programs, or data representation is subject to frequent change.

Relation to Programming Languages: Concurrency primitives required for ~~share blackboard in~~ versions that allow ~~for~~ concurrency ~~between integral programs~~ Synchronization are

Rationale for Not Using in our Solution: Sudoku requires constant updates and communication to the data layer. Also, this is a well-defined gamed at this juncture, and therefore, well-defined solutions are attainable.

Style [Pipe-and-Filter]

Summary Description: Streams used to pass data from one program to another; separate programs executed independently with option of concurrency

Components: Filters (AKA independent programs)

Connectors: Data router streams; operating system service

Data Elements: Linear data stream; not explicit

Topology: Pipeline

Additional Constraints:

Qualities Yielded: Simplicity of data streams (in and out) facilitates creation of unique combinations of filters for new programs; filters are autonomous

Typical Uses: Universal in operating system programming

Cautions: Not ideal if: programs need to interact or exchange of complex data structures is needed

Relation to Programming Languages: Unix shells

Rationale for Not Using in our Solution: If the Sudoku application was being built for command line prompts, this style could be useful. Since the application will be web based, this style is not ideal. The data passed between the classes is very complex and communication between classes is ideal.

Any styles that facilitate:

1. Scaling
2. Ease of Understanding
3. Adaptation
4. Separation of Concerns

Pattern [State-Logic-Display (Three-Tier)]

Summary Description: Three-tiered architecture where there is a data source behind set of logic rules, and the logic is accessed by a user interface.

Context of Use (Intent): To have a maintainable, simple architecture that keeps presentation, application processing, and data management functions separate.

Problem Statement: How will we use the three-tier pattern to formulate our program's architecture?

Solution Description: Carefully construct each tier by determining its responsibilities and functionality.

Variants and Related Patterns: Model-View-Controller

Known Uses: Business applications, multi-player games, and web-based applications

Consequences: Segregation of an application into tiers allows us to modify tiers individually without rework of the entire program.

Rationale: The three-tier pattern is a simple, flexible architecture that is maintainable and fits our project needs.

remove from here and put in Logical Arch.

Class:	Collaborator:
Responsibility:	

new page

Interconnection Mechanism

[Procedure Call Connector]

Summary Description: Model the flow of control among components through various invocation techniques, and perform transfer of data among the interacting components through the use of parameters and return values.

Rationale: Since the Sudoku application is a synchronous blocking process and method calls have a single source and destination, this connector type will be most useful.

[Event Connector]

Summary Description: Invoked by an event and passes messages/control to interested parties.

Rationale: This type of connector could be used in the UI sector of our application. For instance, a mouse click in a cell of the Sudoku puzzle would be the invoking event to send a message to the business logic section of the application to pop up an input box for the user to input their guess for the value of the cell.

[Data Access Connector]

Summary Description: Allows component access to data store

Rationale: Sudoku application has a mock database component housing puzzle values, guesses, locations, etc. Data Access connectors will allow the business logic tier of the application to gain and manipulate data housed in this component. For instance, a user inputs a guess into a cell on the Sudoku puzzle. The business logic tier will validate whether it's a valid input. If it is, a data access connector will be invoked to update the database guess value for the cell.

Hypothesize what connectors would be useful for:

1. Scaling
2. Adaptation

and just list them as options for future thinking.

PHILOSOPHIES & PREFERENCES

1. Construct a concisely detailed architecture for our project to solve design decisions.
2. Document architecture development thoroughly by creating these documents:
    a. Conceptual Architecture
    b. Interface Control
    c. Logical Architecture
    d. Execution Architecture
3. Complete all documents, processes and development required to submit our Sudoku application and receive a grade of A+. ☺

GUIDELINES & POLICIES

1. Reach weekly goals throughout the semester through teamwork / Individuals & team ~~have~~ set weekly goals
2. Utilize our professor's knowledge and skill as needed

ADDITIONAL INFORMATION

1. Our project will be stored and accessible to each team member on a Git repository.
   Where is info about the account recorded for us?
2. Our project will use additional libraries and jar files from:
    a. Java SDK 7
    b. Maven – What for? } Versions?
    c. Groovy – What for? } Intellij