Extra work
Evaluate Architectural Styles (implicit invocation)
Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu
November 22, 2021

Key Word in Context

Implicit invocation style

Student name: Reem Ezeddin

ID: 3079336

Mini-Project
Evaluate Architectural Styles (Implicit Invocation)
Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu
November 22, 2021

Contents

1- Identification of Problem 3
2- Make or Break Issues Driving Design Decisions 3
3- Significant Features of the Architecture 3
4- Compare Alternatives-Strengths and Weaknesses 5
Conclusions 5
Performance Comparison: 6
Summary 6

Mini-Project
Evaluate Architectural Styles (Implicit Invocation)
Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu
November 22, 2021

1- Identification of Problem

The KWIC system takes text lines of input, processes the line, first by outputting it in its original format, then shifting the last word to the beginning of the text line and outputting the adjusted text line. This process is repeated until all words have been shifted to the beginning position. The entire set of text lines will be alphabetized providing an alphabetized list of all words in the first position of the line, making it easy to identify all words and phrases in the text file.

In this document, I will introduce the Implicit Invocation style for solving this problem.

It will compare these styles across six categories:

- Change in algorithm
- Change in data representations
- Change in functionality
- Performance in time
- Performance in space
- Reusability.

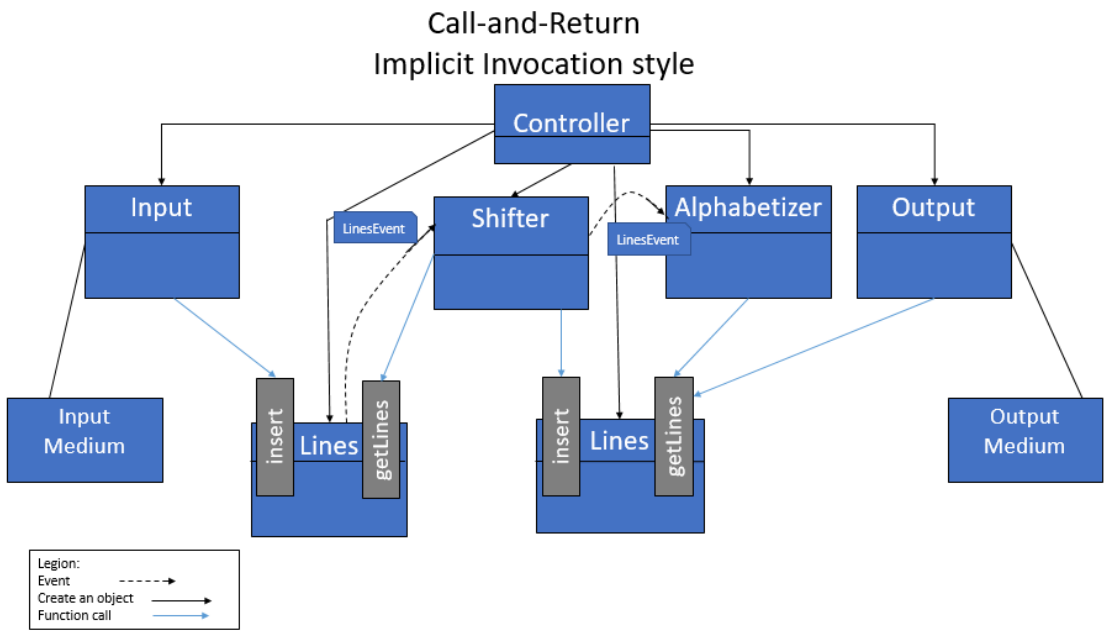
2- Make or Break Issues Driving Design Decisions

system is structured around event handling. Instead of directly executing a process, implicit invocation allows a component to announce (or broadcast) one or more events. By connecting a process with an event, other components in the system might register an interest in it. When the event is announced, the system automatically calls all of the procedures that have been set up for it. As a result, an event 'implicitly' causes procedures in other modules to be invoked.

3- Significant Features of the Architecture

The architectural style combines both Shared Data and Abstract Data Type. However, the interface to the data is abstract. A key feature of this style is that computations are invoked implicitly as data is modified as a result of this interaction is based on an active data model.

Mini-Project
Evaluate Architectural Styles (Implicit Invocation)
Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu
November 22, 2021



Components: objects that register interest in “events” and objects that “signal events”. Components have a dual role: announcer, it posts events to a medium, and listener which registers with a medium to receive certain events of interest

Connectors: automatic method invocation and event announcement.

Even: a message or signal that can be sent across a media. Because either of these factors allows interested parties (components) to register to receive the event, the event must include an identifier for "event type" or "event source." Data may be present in the event. A timestamp is frequently included in an event.

Data elements: will be provided via parameters into methods.

Topology: arbitrary.

Typical use: distinguish or differentiate between the intent to take a certain action from the actual implementation of that action.

Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu

4- Compare Alternatives-Strengths and Weaknesses

Strengths:

- This system supports functionality enhancement.
- Insulate computations from changes in data representation. This will bring easy modification of individual modules.
- If some components malfunction, the system might still perform some functions.
- Loose coupling.
- Components are easy to be replaced/added/used again

Weaknesses:

- Difficult to control the processing order.
- Most natural implementations of this kind tend to use more space.
- Large amounts of data are difficult to transfer from events.
- Component has no idea what other components will respond to against an event
- Global Performance and Resource Management become a serious issue since all components share a common repository

Conclusions

Implicit Invocation is not efficient when it is come to change data representation, but it is easy to maintain. Reusability is also supported

Change in algorithm:

Provides a loosely coupled design between objects that interact. Loosely coupled objects are flexible with changing requirements.

Change in data representation:

It protects computations against data representation changes. Because the implicitly invoked modules only rely on the occurrence of certain externally triggered events.

Change in functionality:

Additional modules can be added to the system by registering them to be invoked on data-changing events, making it simple to add new functionality to the system. Because data is accessed abstractly.

Mini-Project
Evaluate Architectural Styles (Implicit Invocation)
Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu
November 22, 2021

Performance Comparison:

Performance Evaluations	Main and subroutines	OO	Pipe-and-filters	Implicit Invocation
Performance:				
Size of object code	595 KB	473KB	5KB	467 KB
Execution Time1	50.12ms	157ms	7.52ms	1128ms
Execution Time 2	61.27ms	234ms	7.09ms	1161ms
Execution Time 3	55.14ms	160ms	7.42ms	1108ms
Average Exec Time	55.51ms	184ms	7.34ms	1132ms

Summary

The best architecture for this project should be based on the architectural quality attributes required by the stakeholders. If performance is the top attribute, then implicit invocation will not be a good choice. If modifiability or reusing is the top attribute, then it would be good. It is seldom that just one quality attribute is used to determine an architecture. All top-quality attributes must be taken into consideration and balanced to provide the stakeholders with the best possible solution. Below is a comparison of quality attributes to be taking into consideration:

Description	M&S	OO	P&F	I-I
Change in algorithm				
Process Intuitive	Yes	No	Yes	No
Uses information hiding	No	Yes	No	Yes
Easy to scale	No	Yes	Yes	Yes
Able to process non-sequential/non-batch	Yes	Yes	No	Yes
Description	**M&S**	**OO**	**P&F**	**I-I**
Change in data representations				
Uses shared data across modules	Yes	Yes	No	Yes
Good attention to data structure	No	X	No	X

Mini-Project
Evaluate Architectural Styles (Implicit Invocation)
Reem Ezeddin
IT818: Software Architecture
Mondays, 6:10-9:00 PM
R153e128@home.ku.edu
November 22, 2021

Description	M&S	OO	P&F	I-I
Change in functionality				
Easy to modify	No	Yes	Yes	Yes
Loose coupling	No	Yes	No	Yes
Efficient for high-performance modules	X	No	Yes	No
Description	**M&S**	**OO**	**P&F**	**I-I**
Preformance in time	Yes	X	No	No
Description	**M&S**	**OO**	**P&F**	**I-I**
Performance in space	Yes	X	No	No
Description	**M&S**	**OO**	**P&F**	**I-I**
Reusability	No	Yes	Yes	Yes
Reduced need for significant middleware	Yes	No	X	
Legion for KWIC project architecture:				
Yes = Advantage in architecture				
No = Disadvantage in architecture				
X = Neither advantage or disadvantage in architecture				