multithreading where the presentation as the user interface can be shown before the abstraction has fully initialized. This is unlike other software architecture known as Model-View-Controller (MVC) which is restricted to simple GUI's with one or more views on the same model.

- The Control in the PAC is similar to the Controller in the MVC architecture to some extent. The PAC architecture does not have the model as its core component. Like the MVC the Abstraction contains the data in the PAC.

Unit 6

Contents

- **Architectural Patterns – 3**
- Adaptable Systems: Microkernel
- Reflection

Chapter 8: Architectural Patterns-3

Adaptive systems :

Cause and Effect

- For many years scientists saw the universe as a linear place. One where simple rules of cause and effect apply. They viewed the universe as big machine and thought that if they took the machine apart and understood the parts, then they would understand the whole.

- They also thought that the universe's components could be viewed as machines, believing that if we worked on the parts of these machines and made each part work better, then the whole would work better. Scientists believed the universe and everything in it could be predicted and controlled.

Complexity Theory

- Gradually as scientists of all disciplines explored these phenomena a new theory emerged complexity theory, A theory based on relationships, emergence, patterns and iterations. A theory that maintains that the universe is full of systems, weather systems, immune

systems, social systems etc and that these systems are complex and constantly adapting to their environment. Hence complex adaptive systems.

- The agents in the system are all the components of that system. For example the air and water molecules in a weather system, and flora and fauna in an ecosystem. These agents interact and connect with each other in unpredictable and unplanned ways.

- But from this mass of interactions regularities emerge and start to form a pattern which feeds back on the system and informs the interactions of the agents. For example in an ecosystem if a virus starts to deplete one species this results in a greater or lesser food supply for others in the system which affects their behaviour and their numbers. A period of flux occurs in all the populations in the system until a new balance is established.

Properties

Complex adaptive systems have many properties and the most important are,

· Emergence: Rather than being planned or controlled the agents in the system interact in apparently random ways. From all these interactions patterns emerge which informs the behaviour of the agents within the system and the behaviour of the system itself. For example a termite hill is a wondrous piece of architecture with a maze of interconnecting passages, large caverns, ventilation tunnels and much more. Yet there is no grand plan, the hill just emerges as a result of the termites following a few simple local rules.

· Co-evolution: All systems exist within their own environment and they are also part of that environment. Therefore, as their environment changes they need to change to ensure best fit. But because they are part of their environment, when they change, they change their environment, and as it has changed they need to change again, and so it goes on as a constant process. (Perhaps it should have been Darwin's "Theory of Co-evolution".)

Some people draw a distinction between complex adaptive systems and complex evolving systems. Where the former continuously adapt to the changes around them but do not learn from the process. And where the latter learn and evolve from each change enabling them to influence their environment, better predict likely changes in the future..

· Requisite Variety: The greater the variety within the system the stronger it is. In fact ambiguity and paradox abound in complex adaptive systems which use contradictions to create new possibilities to co-evolve with their environment. Democracy is a good example in that its strength is derived from its tolerance and even insistence in a variety..

· Connectivity: The ways in which the agents in a system connect and relate to one another is critical to the survival of the system, because it is from these connections that the patterns are formed and the feedback disseminated. The relationships between the agents are generally more important than the agents themselves.

· Simple Rules: Complex adaptive systems are not complicated. The emerging patterns may have

a rich variety, but like a kaleidoscope the rules governing the function of the system are quite simple. A classic example is that all the water systems in the world, all the streams, rivers, lakes, oceans, waterfalls etc with their infinite beauty, power and variety are governed by the simple principle that water finds its own level.

· Iteration: Small changes in the initial conditions of the system can have significant effects after they have passed through the emergence - feedback loop a few times (often referred to as the butterfly effect). A rolling snowball for example gains on each roll much more snow than it did on the previous roll and very soon a fist sized snowball .

· Self Organizing: There is no hierarchy of command and control in a complex adaptive system. There is no planning or managing, but there is a constant re-organizing to find the best fit with the environment. A classic example is that if one were to take any western town and add up all the food in the shops and divide by the number of people in the town there will be near enough two weeks supply of food, but there is no food plan, food manager or any other formal controlling process. The system is continually self organizing through the process of emergence and feedback.

· Edge of Chaos: Complexity theory is not the same as chaos theory, which is derived from mathematics. But chaos does have a place in complexity theory in that systems exist on a spectrum ranging from equilibrium to chaos. A system in equilibrium does not have the internal dynamics to enable it to respond to its environment and will slowly (or quickly) die. A system in chaos ceases to function as a system. The most productive state to be in is at the edge of chaos where there is maximum variety and creativity, leading to new possibilities.

Complex adaptive: systems are all around us. Most things we take for granted are complex adaptive systems, and the agents in every system exist and behave in total ignorance of the concept but that does not impede their contribution to the system. Complex Adaptive Systems are a model for thinking about the world around us not a model for predicting what will happen.

Microkernel :

The **Microkernel** pattern applies to software systems that must be able to adapt to changing system requirements. It separates a minimal functional core from extended functionality and customer-specific parts. The microkernel also serves as a socket for plugging in these extensions and coordinating their collaboration.

Context and Problem

The pattern may be applied in the context of complex software systems serving as a platform for other software applications. Such complex systems usually should be extensible and adaptable to emerging technologies, capable of coping with a range of standards and technologies. They also need to possess high performance and scalability qualities; as a result, low memory consumption and low processing demands are required. Taken together, the above requirements are difficult to

achieve.

Solution, Consequences and Liabilities

- The most important core services of the system should be encapsulated in a microkernel component. The microkernel maintains the system resources and allows other components to interact with each other as well as to access the functionality of the microkernel. It encapsulates a significant part of system-specific dependencies, such as hardware-dependent aspects. The size of the microkernel should be kept to a minimum, therefore, only part of the core functionality can be included in it; the rest of the core functionality is deferred to separate internal servers.

- Internal servers extend the functionalities of the microkernel. Internal servers can for example handle graphics and storage media. Internal servers can have their own processes or they can be shared Dynamic Link Libraries (DLL) loaded inside the kernel. The external server provides a more complex functionality; they are built on top of the core services provided by the microkernel.

- Different external servers may be needed in the system in order to provide the functionality for specific application domains. These servers run in separate processes and employ the communication facilities provided by microkernel to receive requests from their clients and to return the results. The role of the adapter is to provide a transparent interface for clients to communicate with external servers. Adapter hides the system dependencies such as communication facilities from the client. Adapter thus improves the scalability and changeability of the system. The adapter enables the servers and clients to be distributed over a network.

The benefits of the pattern can be mentioned like:

Good portability, since only microkernel need to be modified when porting the system to a new environment.

High flexibility and extensibility, as modifications or extensions can be done by modifying or extending internal servers.

Separation of low-level mechanisms (provided by microkernel and internal servers) and higher-level policies (provided by external servers) improves maintainability and changeability of the system.

There are some concerns about it as well.

1. The microkernel system requires much more inter-process communication inside one application execution because of the calls to internal and external servers.
2. The design and implementation of the microkernel -based system is far more complex than of a monolithic system

Reflection :

- The **Reflection** pattern provides a mechanism for changing structure and behavior of software systems dynamically. It supports the modification of fundamental aspects, such as type structures and function call mechanisms. In this pattern, an application is split into two parts. A meta level provides information about selected system properties and makes the software self-aware. A base level includes the application logic. Its implementation builds on the meta level. Changes to information kept in the meta level affect subsequent base-level behavior.

- The **Reflection** pattern provides a mechanism for changing structure and behavior of software systems dynamically. It supports the modification of fundamental aspects, such as type structures and function call mechanisms. In this pattern, an application is split into two parts. A meta level provides information about selected system properties and makes the software self-aware. A base level includes the application logic. Its implementation builds on the meta level. Changes to information kept in the meta level affect subsequent base-level behavior.

- The Reflective Blackboard architectural pattern is independent of programming languages and specific implementation frameworks, and its use can minimize the complexity caused by the presence of numerous system-level properties in MASs.

Structure

- The controller subsystem is composed of a meta-object protocol (MOP) component that together with a collection of meta-objects implement the multi-agent system control strategies. Meta-objects are composed of data (metadata) and are responsible for associating specific behavior (reactions) to operations performed over specific pieces of data.

These meta-objects can transparently modify the normal behavior of the blackboard, thus implementing the multi-agent system control strategies. Different agents can act over the blackboard by means of their sensors and effectuators, which can respectively sense and perform changes in the blackboard that can be considered their environment. The agents do not communicate directly; they only write and read data from the blackboard.

- Whenever an agent performs any operation over a specific piece of data stored at blackboard, the MOP component verifies if there is any meta-object associated to it. If positive it executes the reaction associated to the meta-object, i.e. its behavior.The meta-object execution can access the blackboard writing and deleting data.

- In this way, in a reflective blackboard architecture the semantics of a blackboard operation, in fact, is the result of the execution of the meta-objects associated to it. Meta-objects also may exist in the control subsystem without correspondent data in the blackboard. In this way the multi-agent system can associate reactions to data that is part of the multi-agent system vocabulary and probably will be written in the blackboard at runtime.

Reflection is used to intercept and modify the effects of operations of the blackboard.

From the point of view of application agents, computational reflection is transparent: an agent writes a piece of data on the blackboard, and has no knowledge this write operation has been intercepted and redirected to the meta-level. The following scenario illustrates the general behavior of the Reflective Blackboard architecture:

1. A knowledge source (or agent) performs an operation on the blackboard (write for example), supplying a piece of data and expecting some other piece of retrieved data;
2. This operation is intercepted by the meta-level's MOP, which will perform, if specified, control activities over the performed operation;
3. The MOP checks for the existence of meta-objects associated to the blackboard data and related to the performed operation. If the knowledge source has performed a write operation on the blackboard, the searched meta-objects will be those related to the written piece of data. On the other hand, if the knowledge source has performed read or delete operations, the searched meta-object will be related to the piece of data read from the blackboard;

UNIT-7

Contents

- **Some Design Patterns**
- Structural decomposition
- Whole – Part
- Organization of work: Master – Slave
- Access Control
- Proxy