Control Models in Software Engineering

Akshay Sharma, Manushi Shah

Abstract— This research paper gives you the brief idea of the different control models used in software engineering and its types. Control models are widely classified into centralized and event-based control models. Centralized models are classified into call-return model and manager model whilst event-based models are classified into broadcast models and interrupt-driven models. Every model has its own application and expertise.

Index Terms— Software Engineering, Control Model, Centralized Model, Event-based Model

1 Introduction

Software engineering is a very important aspect of business world when it comes to development of software projects. Software engineering is application of engineering to the design, development and maintainence of software.Control models are models deployed in software engineering that are concerned with the control flow between the sub-systems. They are distinct from the system decomposition model. They are classified into centralized and event-based model. They are classified into centralized and event-based models.Centralized models are classified into call-return and manager model. Event-based models are classified into broadcast and interrupt-driven models.

2 Types of Control Models

2.1 Centralized Model

Centralized model is a formulation of centralized control in which one subsystem has overall responsibility for control and starts and stops other subsystems. It is a control sub-system that takes responsibility for managing the execution of other subsystems.

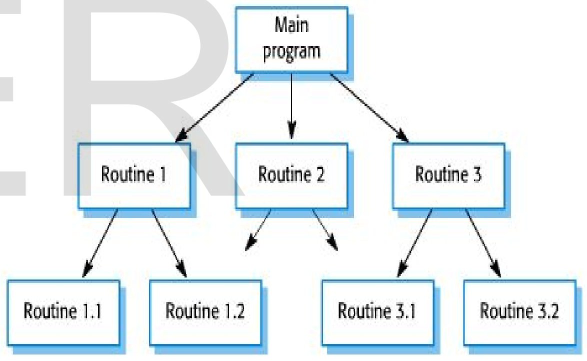
2.2 Event-based Model

Event-based models are those in which each sub-system can respond to externally generated events from other subsystems or the system’s environement. It is a system driven by externally generated events where the timing of the events is out with the control of the subsystems which process the event.

3 Types of Centralized Models

3.1 Call-return Model

In call-return model, it is a model which has top-down sub-routine architecture where control starts at the top of a subroutine hierarchy and moves downwards. It is applicable to sequential systems. This familiar model is embedded in programming languages such as C, Ada and Pascal. Control passes from a higher-level routine in the hierarchy to lower-level routine.This call-return model may be used at the module level to control functions or objects.



Fig(3.1).Call Return Model

The call-return model is illustrated in Figure 3.1.The main program calls routines 1, 2 and 3 whilst Routine 1 can call Routines 1.1 or 1.2. This is a model of program dynamics. It is not a structural model; there is no need for Routine 1.1, for example, to be a part of Routine 1.

3.2 Manager Model

Manager model is applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes. It can be implemented in sequential systems as a case system.

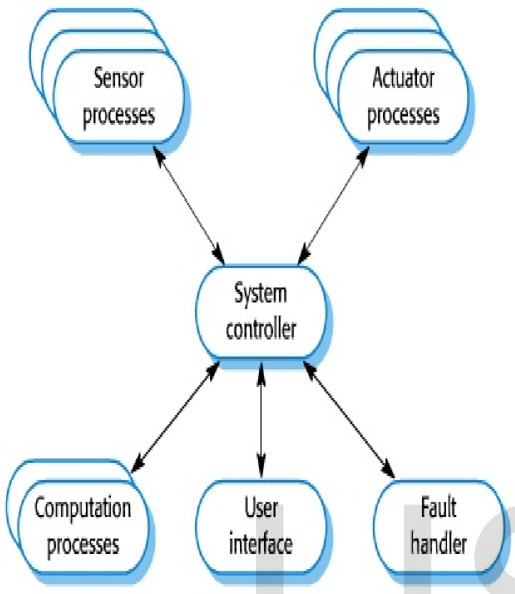


Fig (3.2). Manager Model

Figure 3.2 is an illustration of centralized management model for a concurrent system. It is often used in real time systems which do not have very tight constraints.The central controller manages the execution of a set of processes associated with sensors and actuators.The system controller process decides when processes should be started or stopped depending on system state variables. The controller usually loops continuously, polling sensors and other processes for events or state changes. For this reason, this model is called an event-loop model.

4 Types of Event-based Models

4.1 Broadcast Model

It is a model in which an event is broadcast to all subsystems. Any subsystem which can handle the broadcasting event may behave as a broadcast model. These are effective in integrating components distributed across different computers on a network. The advantage of this model is that evolution is simple. This distribution is transparent to other components. The disadvantage is that the components don't know if the event will be handled.

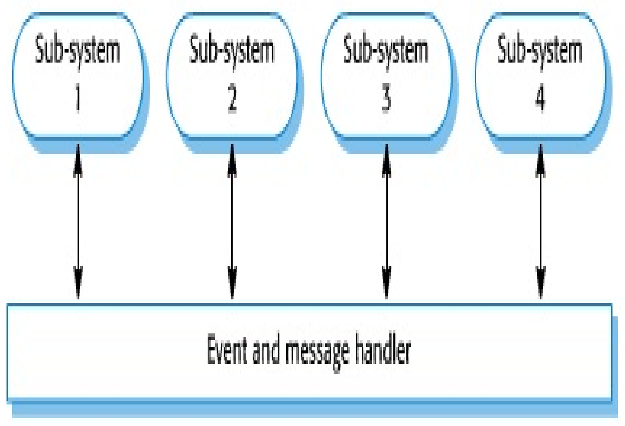


Fig (4.1).Broadcast Model

In Figure 4.1, components register an interest in specific events. When these events occur, control is transferred to the component that can handle the event.

4.2 Interrupt-driven Model

Interrupt-driven model is used in real time systems where interrupts are detected by and interrupt handler and passed to some other component for processing. This model is used in real-time systems where immediate response to some event is necessary.The advantage is that it allows very fast responses to events to be implemented. The disadvantage is that it is complex to program an difficult to validate.

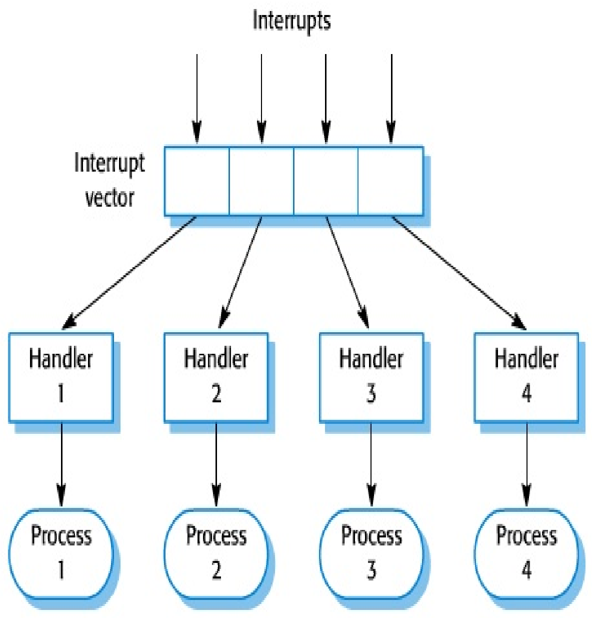


Fig (4.2). Interrupt Driven Model

In Figure 4.2, there are known number of interrupt types with a handler for each type. Each type of interrupt is associated with the memory location where it handler's address is stored.The interrupt handler may start or stop the processes in response to the event signaled by the interrupt.

5 End Sections

5.1 Conclusion

In an era of meeting deadlines and covering an execution of services round the clock, these control models deploy efficient solutions to new requirements of reaching goals and meeting expectations of the clients towards the softwares engineered.

5.2 Acknowledgments

We the authors would like to thank Prof. Sharnil Pandya for giving his valuable time and proficiency in guiding us making our research paper a consistent and important resource of information.

5.3 References

[1] http://www.cs.odu.edu/~price/cs451/Lectures/05design/arch/arch_htsu2.html

[2] http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Architecture/ArchPatterns/CentralControl.html

[3] http://www.slideshare.net/ahirsiddharth/software-engineering-ch11

[4] http://phoenix.goucher.edu/~kelliher/cs319/nov13.html

[5] http://en.wikipedia.org/wiki/Management_control_system

[6] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5719840&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5719840

[7] http://en.wikipedia.org/wiki/Empirical_process_(process_control_model)