ISSN:2278-7690 International Journal Of Research In Educational Methodology
Wwwcirworld.com Council For Innovative Research Vol.2, No.1 December 2012

Modeling Real Time Scheduler in OOAD Using UML

R.Sathiyaraj
Assistant Professor
Dept. of CSE
MITS,Madanapalle

N.Sudhakar Yadav
Assistant Professor
Dept. of CSE
MITS,Madanapalle

M.Prabhakar
Assistant Professor
Dept. of CSE
MITS,Madanapalle

ABSTRACT: Object-oriented analysis and design (OOA&D) tools support object analysis and design technologies and commonly use Unified Modeling Language (UML) notation with a variety of methodologies to assist in the creation of highly modular and reusable software. In this Paper, a brief overview of modeling of an application Real Time Scheduler using UML notations and their strategies with various problems has been discussed.

Keywords;- Object Oriented Analysis and Design(OOAD),Unified Modeling Language(UML),Real Time Scheduler, Rational rose.

1. INTRODUCTION

Object-oriented analysis and design (OOA&D) tools support object analysis and design technologies and commonly use Unified Modeling Language (UML) notation with a variety of methodologies to assist in the creation of highly modular and reusable software. Object-oriented analysis and design (OOA&D) tools support object analysis and design technologies and commonly use Unified Modeling Language (UML) notation with a variety of methodologies to assist in the creation of highly modular and reusable software. Many methodologies have been proposed for object oriented software development. A methodology usually includes:

- ➢ Notation: graphical representations of classes and their relationships and interactions.
- ➢ Process: suggested set of steps to carry out for transforming requirements into a working system.
- ➢ Tools: software for drawings and documentation.

1.1 History of UML

In October 1994, Rumbaugh joined Booch's company (Rational Software Corporation) to unify the Booch and OMT methods. A draft was released in October 1995, and then called the Unified Method. In 1995 Jacobson also joined the unification effort, merging in the OOSE method. Preliminary documents for the Unified Modeling Language (UML) were released during 1996. The UML Version 1.0 definition was finalized in January 1997, including the work of the

UML Partners consortium. UML 1.1 was adopted by the Object Management Group (OMG) in November 1997. OMG is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Revisions to the UML specification through Version 1.5 were released in March 2003. OMG is upgrading UML to Version 2.0. Adopted in late 2003 and posted on OMG's website labeled "UML 2.0 Final Adopted Specification'. Beginning with UML 2.0, the UML Specification was split into specifications: ***Infrastructure*** and ***Superstructure***. The ***UML infrastructure*** specification defines the foundational language constructs required for UML 2.1.1. It is complemented by ***UML Superstructure***, which defines the user level constructs required for UML 2.1.1. The two complementary specifications constitute a complete specification for the UML 2 modeling language.UML 2.1.2 has been released on November 2007. UML 2.2 has been released on February 2009. A complete specification for the UML 2 modeling language, UML 2.3 has been released on May 2010.

UML Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed, and was created by, the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software-intensive systems. UML combines techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. The UML diagrams are:

- ✓ Use Case Diagrams
- ✓ Class Diagrams
- ✓ Sequence Diagrams
- ✓ Collaboration Diagrams
- ✓ State Diagrams
- ✓ Activity Diagrams
- ✓ Component Diagrams
- ✓ Deployment Diagrams

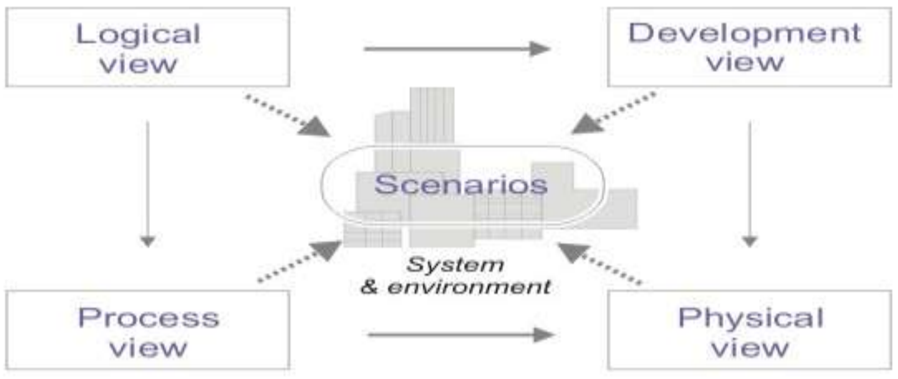


Fig 1: Architectural View Model of UML diagrams

2. MODELING AN APPLICATION WITH UML DIAGRAMS

It is important to distinguish between the UML model and the set of diagrams of a system. A diagram is a partial graphic representation of a system's model. The model also contains documentation that drives the model elements and diagrams (such as written use cases).

UML diagrams represent two different views of a system model:

1. Static (or structural) view: emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams.

2. Dynamic (or behavioral) view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.

Modeling an application Real Time Scheduler for a system using IBM Rational Rose, where the time constraint is very much important when compared to other type of system. The system requests the user to check the deadline for arriving tasks. The system compares the deadlines and allocates the process. It also changes the process states.

2.1 Diagrams overview

UML 2.2 has 14 types of diagrams divided into two categories. Seven diagram types represent structural information, and the other seven represent general types of behavior, including four that represent different aspects of interactions.UML does not restrict UML element types to a certain diagram type. In general, every UML element may appear on almost all types of diagrams; this flexibility has been partially restricted in UML 2.0. UML profiles may define additional diagram types or extend existing diagrams with additional notations.

2.1.1 Behavior diagrams

Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

2.1.1.1 Use Case Diagram

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and the use cases, and generalization among the use cases. The functionality of a system is described in a number of different use cases, each of which represents a specific flow of events in the system.

This use case start the actor to place set of processes with certain tasks. The system requests the resources required. The system requests the user to check the deadline for arriving tasks. The system compares the deadlines and allocates the process. It also changes the process status.

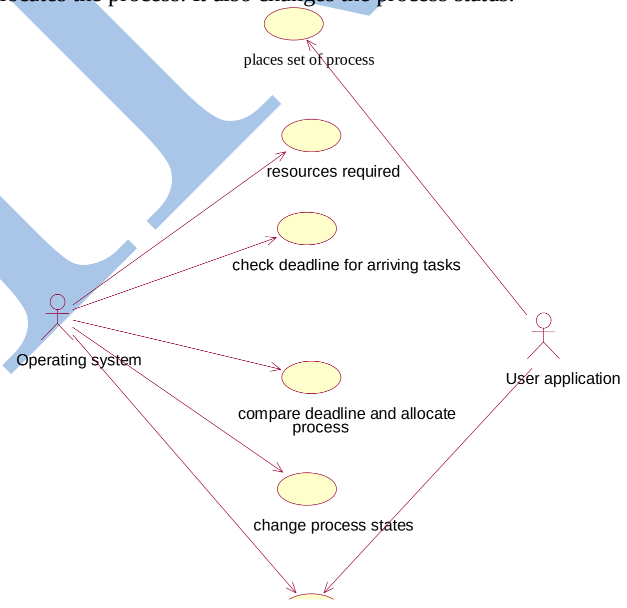
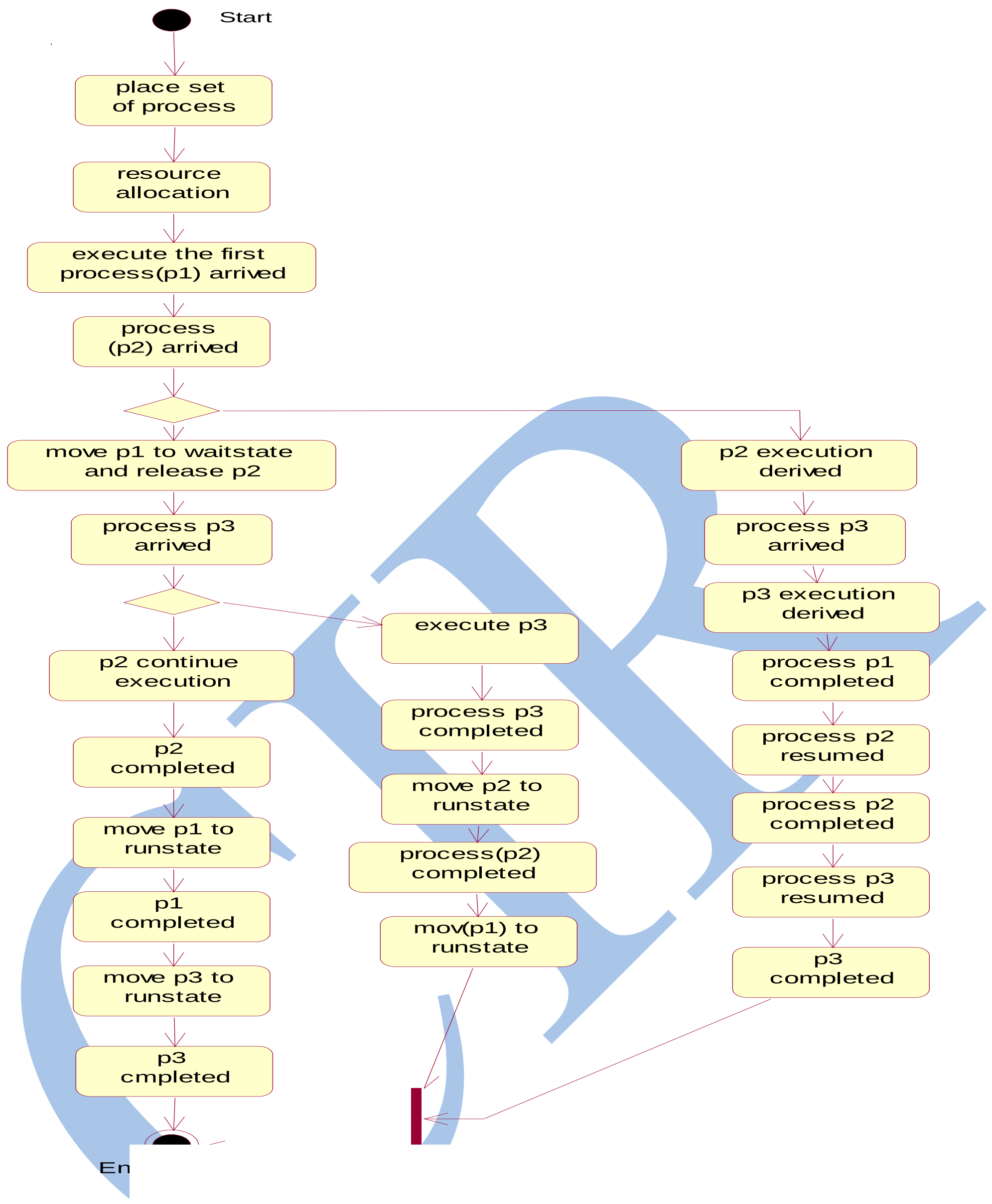


Fig 2: Use case diagram

2.1.1.2 Activity diagram

An activity diagram is a variation or special case of state machine. The purpose of an activity diagram is to provide a view of flows and what is going on inside a use case or among several use cases. It is similar to a state chart diagram, where a token represents an operation. An activity is shown as a round box, containing the name of operation.



3: Activity diagram

2.1.2 *Structure diagrams*

Structure diagrams emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

2.1.2.1 *Class diagram*

A class diagram is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents. A class is drawn as a rectangle with three compartments separated by horizontal lines. The top compartment holds the class name, middle compartment holds the attributes, and the bottom compartment holds the list of operations.

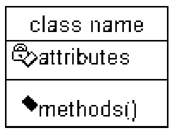


Fig 4: Representation of Class diagram

International Journal Of Research In Educational Methodology
Council For Innovative Research Vol.2, No.1 December 2012

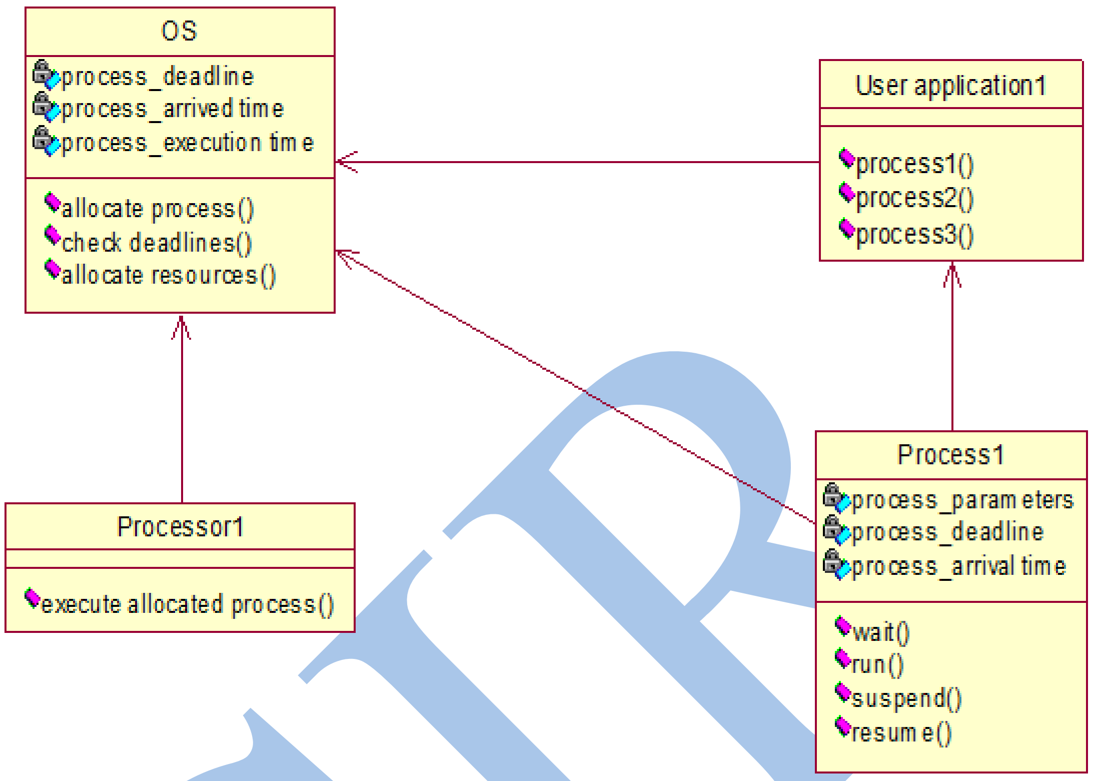


Fig 5: Class Diagram

2.1.3 Interaction diagrams

Interaction diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled:

2.1.3.1 Sequence diagram

A sequence diagram shows an interaction arranged in a time sequence. The horizontal line represents different objects. The vertical line represents the object life time, which is the object's existence during the interaction. An object is shown as a box at the top of a dashed vertical line. It is an alternative flow to understand the overall flow of the control of the program.

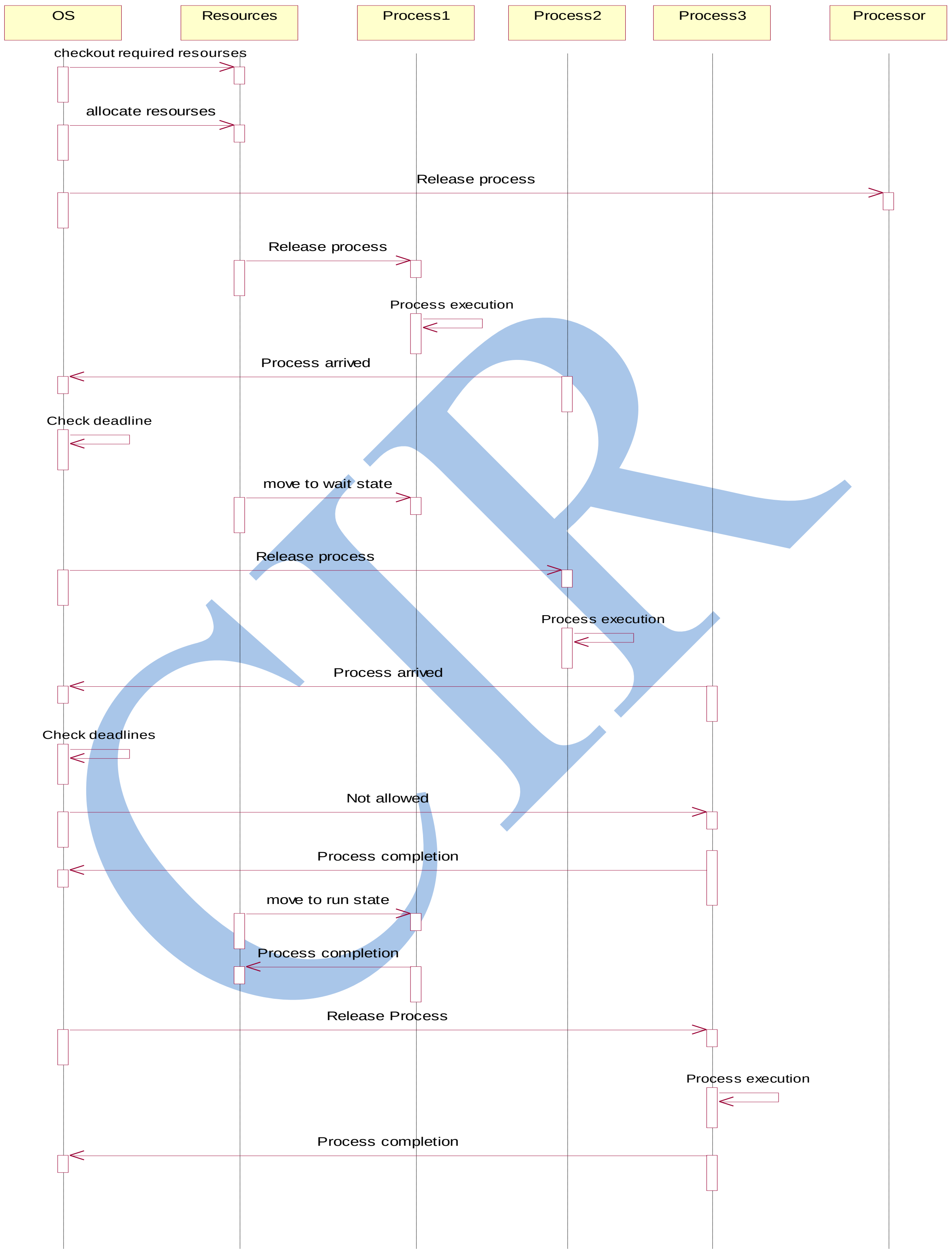


Fig 6: Sequence Diagram

A collaboration diagram represents a collaboration, which is a set of objects, related in a particular context, and interaction, which is a set of messages exchanged among the objects within the collaboration to achieve a desired outcome. The sequence is indicated by numbering the messages.

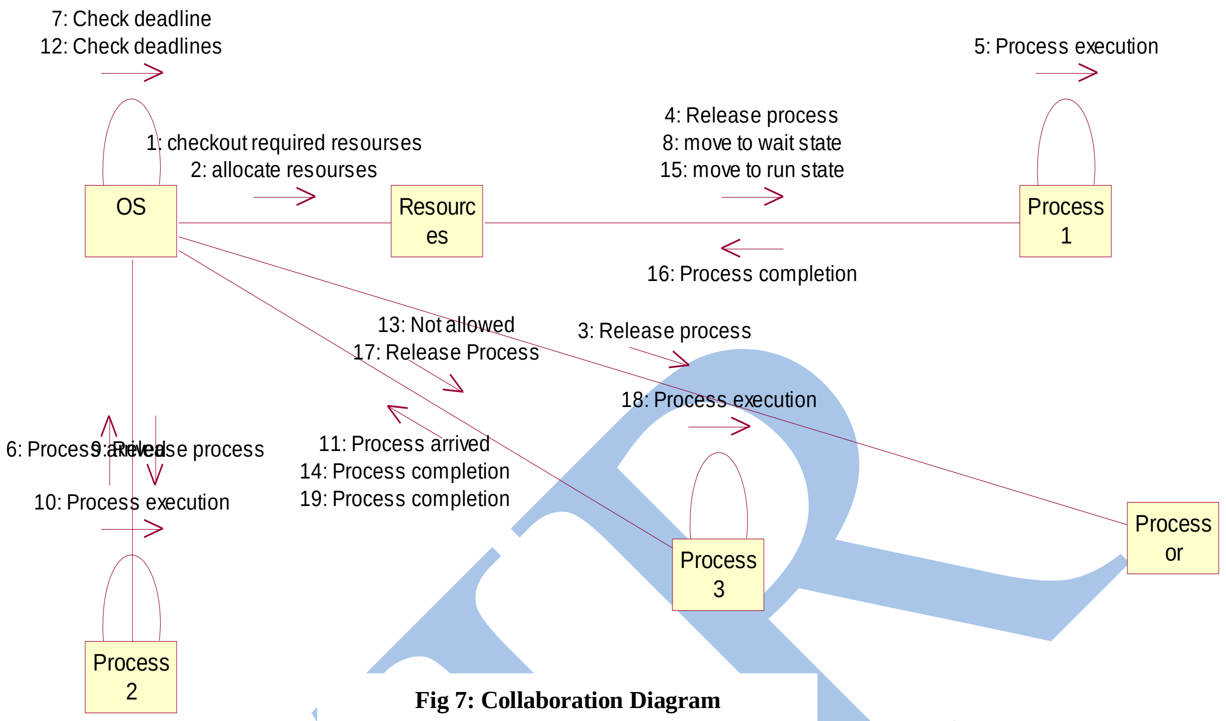


Fig 7: Collaboration Diagram

3. BENEFITS OF UML AND OOAD

- UML breaks the complex system into discrete pieces that can be understood easily.
- Complex system can be understood by the disparate developers who are working on different platforms.
- UML model is not a system or platform specific. It unifies all disparate developers under one roof.
- Promotes better understanding of user requirements
- Leads cleaner design
- Design flexibility
- Decomposition of the system is consistent
- Facilitates data abstraction & information hiding
- Software reuse
- Easy maintenance
- Implementation flexibility

4. CONCLUSION

Object oriented analysis and design elaborates the analysis models to produce implementation specifications.OOAD focuses on what the system does, OOD on how the system does. In this paper, we have modeled an application by implementing the UML diagrams. In the future, we plan to work on more detailed guidance for requirements modeling and develop a demo version of Real Time Scheduler system for demonstrating the power of UML and model-based development approach.

5. REFERENCES

[1] Fowler, M., 2003. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd Edn.Addison-Wesley Professional, Essex, UK., ISBN- 10: 0321193687, pp: 208

[2] Rumbaugh, J., I. Jacobson and G. Booch, 1999. The Unified Modeling Language Reference Manual. 1st Edn., Addison-Wesley Professional, Reading, Mass.,ISBN-10: 020130998X, pp: 550.

[3] UML Specifications, http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML.

[4] What Is Object-Oriented Design?.Object Mentor. http://www.objectmentor.com/omSolutions/oops_what.html.Retrieved 2012-11-10.

[5] Grady Booch,"Object Oriented Analysis and Design with Applications", 3rdedition" http://www.informit.com.

[6] Evans, A.S., 1998. Reasoning with UML class diagrams. Proceedings of the 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques, Oct. 21-23, IEEE Xplore Press, Boca Raton, FL., pp: 102-113. DOI:10.1109/WIFT.1998.766304

[7] Hayes, Frank. "The Reality of Object Reuse," Computer World, May 6, 1996, p. 62.