UML Model Transformation for a Product Line Design

R. Aroulcanessane[#1], S. Srinivasan[*2]

[#]Research Scholar, Sathyabama University, Chennai, Tamil Nadu, India
[1]aroul_308@yahoo.co.in
[*]Professor & Head, Department of Computer Science and engineering, AnnaUniversity, Madurai, Tamil Nadu, India
[2]sriniss@yahoo.com

***Abstract*— There is a tremendous growth in computer and electronics industry in this era. This growth has made a major impact on hardware and software resources. A gap has occurred in designing the hardware resource because, only optimal amount of hardware resources is being used for an application development. The complexity of the design and the scarcity of time (for marketing) require a standard procedural approach, to address problems such as validation, synthesis, verification and testing. We have proposed a procedural approach for the completeness of the hardware and software systems with less effort. We have created an executable model for the application. This model includes different software and hardware modules.A UML based compiler and a hardware-software co-simulation tool is used in our procedural approach, for reconfiguring the architectures and aids validation, verification, mapping and synthesis of the application with limited amount of effort.**

Keyword- containments, dependencies, platform, proxy.

I. INTRODUCTION

In the Software engineering domain there is a trend of adaptation towards methods, tools and languages which are used for the development of a system. Recent techniques for developing software architectures try to capture core design information into sets of semiformal models that are kept permanently by tools. The development of the system with Unified Modelling Language (UML) [6] is moving towards a promising direction, which solves the basic problem regarding software, but lacks in the synthesis of software-hardware, design space, estimation, verification and validation in which some of them are non functional requirements that determines the performance of the system. The worldwide view of the development and performance expects to create major issues onto the performance of the system.

We have introduced an approach for synthesizing reconfigurable hardware using UML models, which helps to convert the object orientation specification into hardware circuits which has a better performance. The computation of the objects is done by structured messages. This approach is suitable for synthesis of standalone hardware and for software-hardware combined synthesis [7] as well. In this approach, we have emphasized the complete object orientation specification which avoids breaking any architectural rules. The paper is organized as 2.An UML approach, 3.Software implementation, 4.Interfaces of hardware and software, 5.Result and discussion and 6.conclusion.

II. AN UML APPROACH

UML is a model description language which confidently supports developing objected oriented software. The materialistic aspects of software can be described using structural diagrams such as class diagrams, component diagrams, deployment diagrams, activity diagrams, sequence diagrams and state machine diagrams. In this paper, we have used the UML diagrams for describing the structure of the MAP processor (multi-channel acquisition processor) as an example. With this application of MAP processor, we checked for the module completeness. Using the operation we have defined the behaviour of the application, these operations are described in detail using the (PDDL) planning domain definition language. The class diagrams model the structure of the class and represent the relationships in the form of containments, inheritance and association. The class diagram is represented using a rectangle in the figure 1.

A class has three parts – the class name at the top, the middle shows the attributes of the class and finally the lower part which has the operations that belong to the class. An association is a common relationship in a class diagram [15] which is used for expressing collaboration among class instances. For example, in the figure 1, the class *Main* is associated with the class *MAPprocessor*. Generalization is another relation which is common in class diagrams. The inheritance of attributes and functions can be represented using generalization.

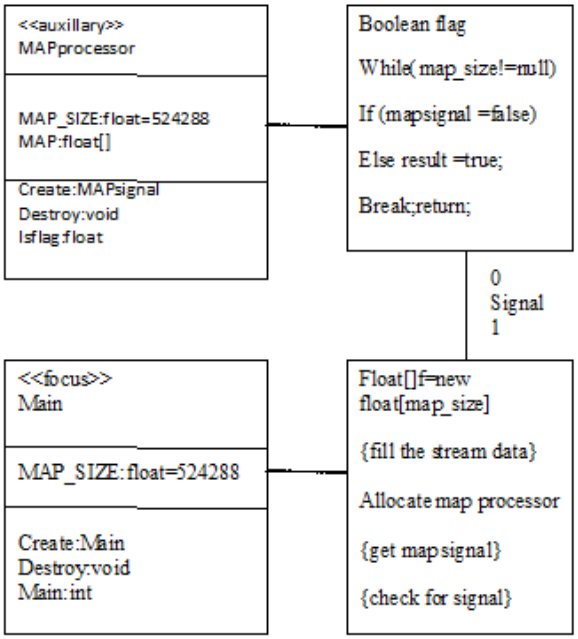

Fig.1. Class diagram for MAP processor design model

The class objects which are present in figure 2 are *bit, remote, time, Boolean, char, string,* for a base class object that has an array. Deployment diagrams are the physical layout of the system, which shows the information about which piece of software run of what piece of hardware [13].

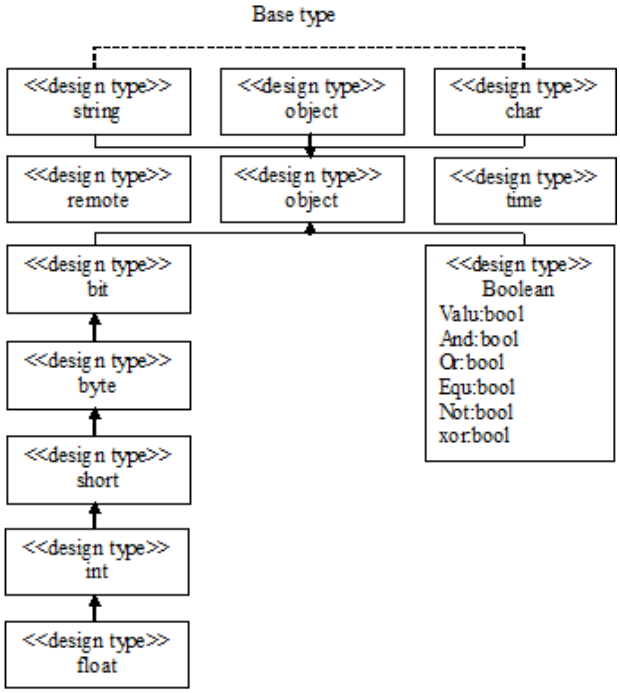

Fig. 2. Class diagram of Platform model Design

The communication paths are shown using the nodes in figure 3. The node represents the hardware piece which hosts the corresponding software. The deployment artefacts are represented using a node which is usually called files. In figure 3, the node *m0* represents the microprocessor which deploys the *MAPprocessor.exe* at runtime.

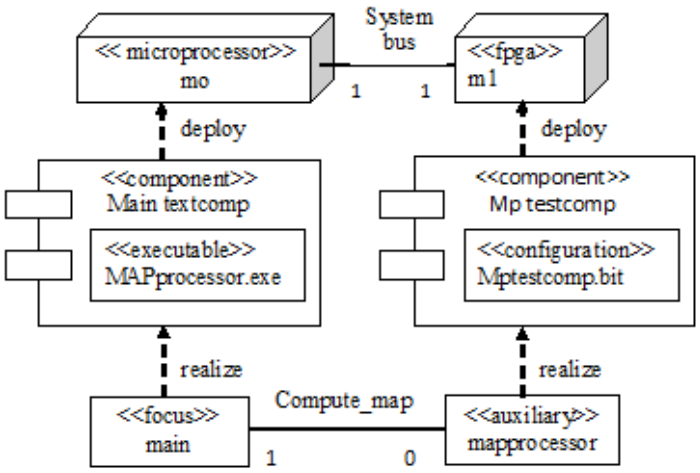

Fig. 3. Implementation and the Development Model for the MAP design model

An artefact *mptestcomp.bit* is used to represent an instance of the component *mptestcomp* which is deployed in the FPGA (field programmable gate array) node. Dependencies are represented using dotted arrows between the elements, a change in any one of the element's definition may lead to a change in another. Using the *<<realize>>* dependency we have show that source is the implementation of the specification which has been defined by the target.

A. *Development Methodology*

In this section we have presented a brief overview of development methodology. The activities and the basic artifacts of our approach is shown in fig 4.

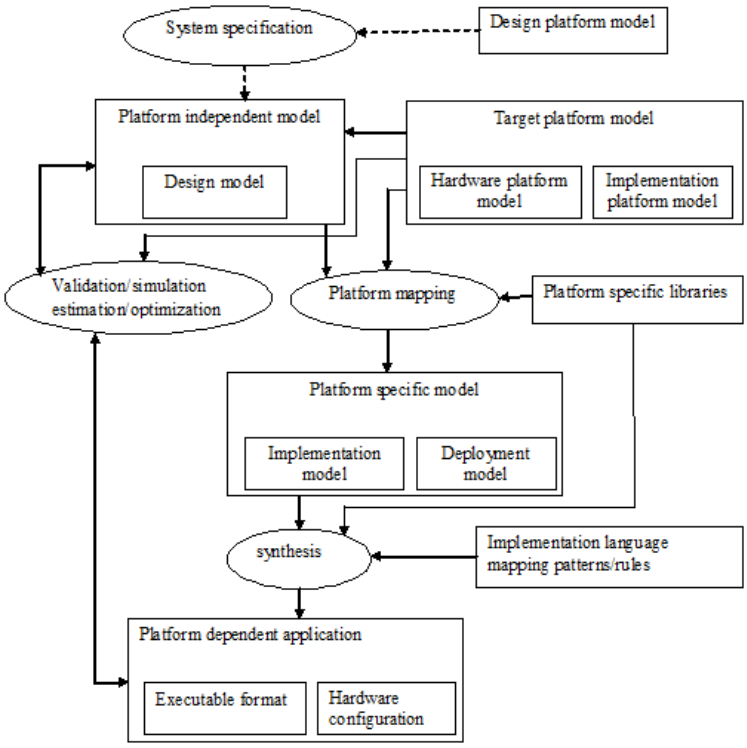

Fig. 4. Artifacts of Development Methodology

In our approach we have incorporated the software hardware codesign into the concepts of model driven architecture (MDA). We have designed the application at the system level using the UML in figure 1, the platform design is described by the set of constraints and types in figure 2.

Using the (PIM) platform independence design model the definitions of the structure, behaviour and functions are identified. Information regarding the mapping of the platform and synthesis is specified by the target platform model (TPM)[4],[8]. It also defines the architecture of target hardware using Unified modelling

language, by specifying the hardware nodes, microprocessors of reconfigurable devices, communication path among them and capabilities of those components which provide an application implementation in figure 3.

A transformation of PIM to the platform specific model is done by mapping application to target hardware structure. In addition the implementation might need the support of operating systems for inter object communication and synchronization with respect to the elements of PIM, for which the resources are provided by the corresponding devices on the target platform.

The last step is the synthesis, if a platform specific model is given to our application; we transform into an implementation model which synthesis into a ready to run system. We generate executable software modules for the function that are deployed onto the microprocessors. Hardware modules are generated with logic of reconfiguration for the corresponding functionalities of the software modules.

B . *Platform and Model*

The application that we have developed is based on the platform, there are many platforms designed, implemented and deployed, which separates the design and implementation in development concern. Our approach identifies portability, reuse, validation and adaptability easily. Platforms play a major role in implementation which makes automatic interpretation to the computers [3], [5].

Assumptions of functionalities are the basic foundation for the development of any platform model. In the approach that we have presented, the assumptions used are made explicit with the platform model, where all the platforms are created using the UML model. The platform models are abstracted to the detailed platform, they try to carry as much of information in order to avoid iteration in the design flow. The relationship between the platform and the application models is shown in the figure 5.

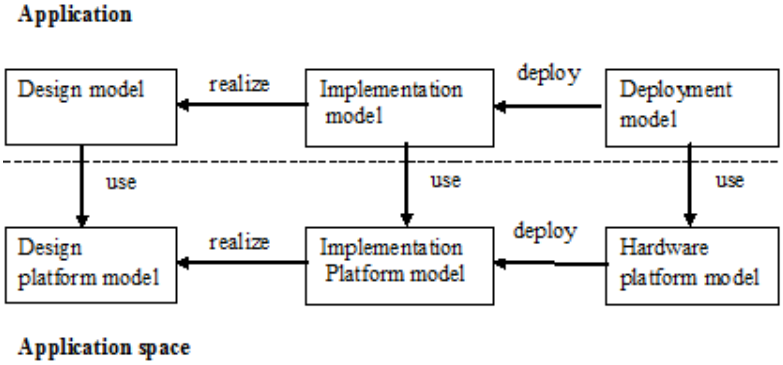

Fig. 5. Relationship among the models

The platform model that is below the dotted horizontal line shows the space of the applications that can be developed with respect to the platform. Defining a complete, concrete application is shown above the line. Platform models can be shared normally by models of different applications.

III. SOFTWARE IMPLEMENTATION

In this section we present the object oriented approach for the synthesis of software from the UML model. The classes that are described are implemented using SystemC. Local proxy objects take care of communication among the local and remote objects. A proxy is automatically instantiated locally for the remote object which is to be accessed by the local object. The proxy precisely handles the communication mechanism. Hence the objects in the application need not share the common address. A proxy is defined as an explicit remote type for the platform models. This property of remote type improves the quality of the model compiler for estimating the quality in the distributed application. Objects of the reconfigurable hardware [10], [11] are monitored and managed by the RTR manager (reliable transaction router) figure 6. The RTR manager encapsulates the specifications of the reconfigurable hardware such as the input, the output, the reconfiguration mode, functions and the communication. An important task of the RTR manager is to manage requests to process the application, which is responsible for the creation and the destruction of the corresponding hardware object. The hardware objects can be created and destroyed on demand. An application which requires a hardware object is instantiated through the RTR manager by type. The RTR manager provides the appropriate object which is instantiated currently by the bit stream hence it serves as the proxy for the application. The RTR manager instantiates the related bit stream dynamically if no bit stream of the type is present.

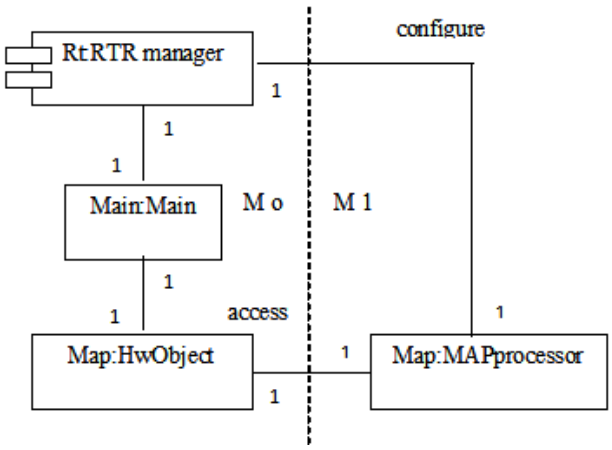

Fig. 6. MAP processor Software Architecture

Fig. 6 shows the principle architecture of the MAP processor example. The instance of the class *Main* and the proxies for the hardware objects are implemented in software. The instance of the class *MAPprocessor* is realized with the help of the reconfigurable resources [9]. The objects for the hardware are accessed through the software's dedicated proxy. The RTR manager is responsible for serving the proxies to the application using the instances. Proxies play a major role and are used in software implementation directly. After servicing, the proxies can be wrapped by the software implementation of the hardware objects. When an instance is created for a software object, it tries to instantiate the corresponding hardware. If it succeeds, the objects of the software switch plainly to the implementation software. This approach allows transparent migration among the hardware and software objects.

IV. INTERFACES OF HARDWARE-SOFTWARE

When the classes, components and features are implemented for software, they can be easily understood and implemented, but when we try to implement these attributes on hardware, some challenging issues must be faced[1],[2].

A. *Polymorphism*

It is the property of an object to have more than one form, it must be supported by the hardware implementation. The current approach avoids this property by overriding behaviour.

B. *Dynamic instantiation/destruction*

Since this process is iterative in nature, the efficiency of the instantiation/destruction slows down and is thus not effective on reconfigurable hardware. Reconfiguration of the reconfigurable devices and is too costly in terms of logic resources and time.

C. *Objects communication*

No common mechanism is present for communication in a mixed software/hardware implementation. This type of mixed implementation has been discussed in the following section. The interfaces to the object oriented implementation along with the reconfigurable hardware are used to define the lifecycle, the access mechanism for the objects and the components to be realized for the reconfigurable hardware. The logical and physical implementation made depends on the target platform and the compiler used for the model.

D. *Life cycle of objects*

The lifecycle of a hardware object is different from the life cycle of a software object. The reconfiguration objects of the hardware may be reused effectively to avoid the cost of implementation. The initial state of the hardware object being UB_OBJECT, the object is not bounded by any of the hardware resources Figure 7. It only acts as a template object that may be realized by hardware configuration. When the template of the component is instantiated, the configuration will be loaded onto the corresponding device [12].

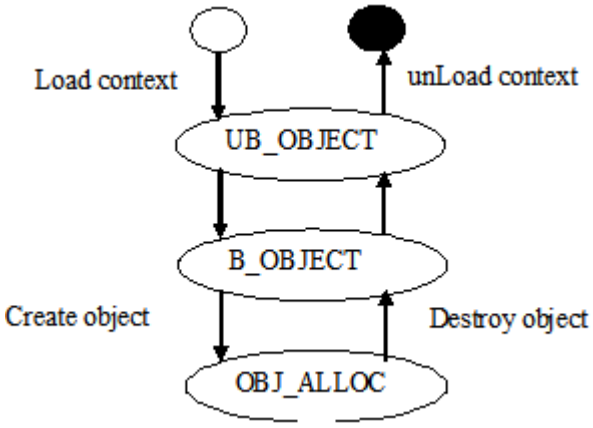

Fig. 7. Lifecycle of Object

The next state is B_OBJECT (bounded object), wherein the object is bounded by the resources of the physical hardware; objects of the B_OBJECT will be allocated using OBJ_ALLOC.

E. *Object Interfaces*

The growth in the microelectronic industry has suddenly increased the number of hardware resources. The design gap is due to the fact that only a small number of applications use these available hardware resources. Access to the objects is allowed by means of the control interface for identification. A unique identification is made for every object space. The unique ID represents the address of the object which is set at the time of initialization. This field is only necessary is the object if the address of the object is to be made explicit. The type field represents the object's dynamic type, which helps in selecting an appropriate implementation, out of the many due to the application's polymorphic nature. The field that is used for the message type identifies the services that the object utilizes [12] when a message is sent to the object. The parameters are passed by means of the data interface.

This interface accesses the objects by means of the respective IP address and the object parameters of the public objects. It also receives the state of object, by providing services to the respective object. Depending on the exception handler that is used during the message passing, the object may try to catch any possible exceptions.

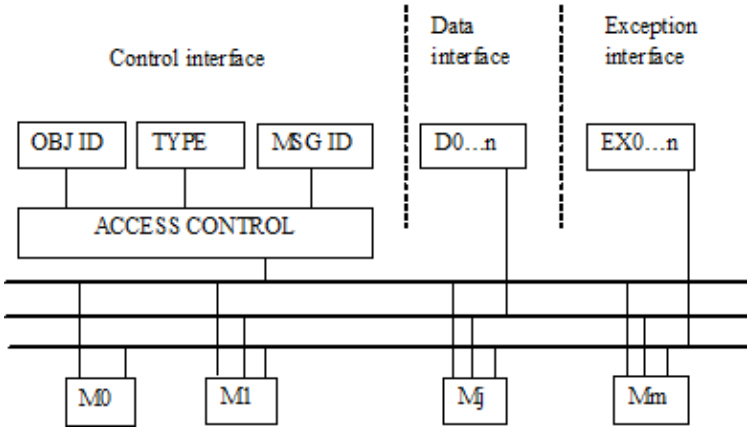

Fig. 8. Interface for Hardware Object

F. *Object Access*

The interfaces of the object which pass the messages (exceptions allow the instantiation) provide services for the hardware and also for object destruction. Each object's interface comprises of a control interface, data interface and an interface handle exceptions. Refer to figure 8[17].

V. RESULTS AND DISCUSSION

In this section, we have demonstrated the effectiveness of our proposed novel procedure in minimizing the consumption of energy of MAP processor which is used for sending multiple signals using the bus.

We have organized our experimental results into three sections.

1. With an embedded software for mapping the hardware and software we have compare the performance results of time and energy
2. Next we have presented the energy savings that is enabled by the runtime for passing the signal.
3. Finally we have presented the performance and the energy cost of reconfiguration.

The execution time i.e., performance of the hardware –software architecture that are shown in the tables are got using a hardware –software simulation framework[16][14]. We hav used the hardware units for the results which are already defined by synosy's RTL estimation tool for power.

A. *Mapping Results of Hardware/Software*

We compared the performance of the Map processor using the hardware –software co- simulation framework, with a standard implementation of embedded software. Table1 shows the result of the comparison in using the time, performance and the energy consumed for passing the signal through communication bus.

The first row which shows the results of a standard algorithm T- Transform, E- Encoding and Q- quantization is done with the help of java software. The second row shows the results of our proposed architecture implementation as quantization for hardware and Encoding for the software. From the table we can identify that the proposed work has a 4X times in performance improvement and an improvement of 5X times in consumption of energy. The improvements are based on the cardinality ratio of hardware implementation which follows the procedure. The results that we have showed are for small amount of signal that has to be used by the *MAPprocessor.*

TABLE I
Comparison of Hardware/ Software co- design architecture with the software implementation

Software	**Hardware**	**Performance (cycles)**	**Energy (mJ)**
T,Q,E	-	1981287	4.520
E	T,Q	455981	0.765

VI. CONCLUSION

In [2], based on the object time environment they have proposed an idea of integrating the performance prediction into a software design environment, with the support of Layered queuing network. In [13], the reconfigurations of the architectures were done based on the resource function management utility, where the functions are presented respect to the hardware and platform models.

In this paper, a fresh approach for reconfigurable architecture of applications, based on UML has been proposed. Architectural design rules play a major role in our research. The inability to formalize the architectural design is one of the main limitations of our research work. This formalization can be taken as a future work, which makes the work as error prone and at the same time it is a time consuming task, taking a lot of effort on part of the architect depends on the software/hardware design.

The artefact chosen as a key concept for our architecture, is the model driven architecture [11], it is a traditional document based development. The conclusions made in this paper can be used in the development of mixed signal applications in which we have provided an application for MAP processor as an example, thereby transforming UML platform independent design models to a hardware- software module has been demonstrated in our work.

REFERENCES

[1] Maykiv. I, et. al, computer Standards and interfaces, volume 34, issue 6, pp509-516,2012.
[2] George Afonso,et. al, Software Implementation versus Hardware Implementation: APLOS, 20012.
[3] Colangelo. D, et. al, Reducing Software Architecture Models Complexity: A Slicing and Abstraction Approach, proceedings Formal Techniques for Networked and Distributed Systems Conf., 2006.
[4] Hofmeister. C, Kruchten. P, Nord. R. L, Obbink. H, Ran. A, America. P, A general model of Software Architecture Design Derived from Five Industrial Approaches, Journal of Systems and Software, Volu.80, no. 1, pp. 106-126, 2007.
[5] Riebisch. M., Problem solution Mapping for Evolution Support of Software Architectural Design, Proceedings of Software Engineering, 2011.
[6] Bass. L, Kazman. R, Ozhaya. I, Developing architectural documentation for the hadoop distributed file system, Open source Systems, springer 2011.
[7] The SAE Architecture and Design Language, http://WWW.aadl.info/,2009.
[8] Pelliccione. P, CHARMY: A Framework for software architecture Specification and Analysis, PhD thesis, Computer Science Dept., Univ. of L'Aquila, May 2005.
[9] Pelliccione. P, Tivoli. M, Bucchiarone. A, Polini. A, An Architectural Approach to the Correct and Automatic Assembly of Evolving Component-Based Systems, http//dx.doi.org/10.1016/j.jss.2008.05.030, 2008.

[10] Erdogmus. H., Architecture meets agility, IEEE Software, projects. Computer.org, 2009.
[11] Aroul canessane. R, Dr. S. Srinivasan, Software Architecture Modeling Framework Using UML, Indian Journal of Computer Science and Engineering, Vol. 4, Issue 2, ISSN : 0976-5166,2013.
[12] Cortellessa. V, A. D. Marco, and P. Inverardi, Model based software performance Analysis , Springer, 2011.
[13] Romina Ermo, Vittorio Cortellessa, Alfonso Pierantonio, Michele Tucci., Performance Driven architectural refactoring through bidirectional model Transformation., ACM SIGSOFT Conference on Quality of Software Architectures, QoSA - 2012.
[14] Brownsword. L., "Current Perspectives on Interoperability", Software Engineering Institute, Carnegie Mellon University, PittsBurgh, 2004.
[15] Morris. G.R, Silas. A.R, Abed. K.H , Analytical and measured sustained bandwidth for an FPGA – based processor, ieeexplore.ieee.org, 2012.
[16] Tang. L, Ambrose. J. A, Parameswaran. S, A Tiny Processor For Reconfigural Baseband Modulation Mapping:MApro, cse. Unsw.edu.au, 2013.

Authors

R. Aroulcanessane received the M.E.in Computer Science and Engineering from Sathyabama University, Chennai , Masters of Computer Applications from St. Joseph's College of Engineering, Chennai, University of Madras. He is presently pursuing the Ph.D degree in the Department of Computer Science and Engineering, Sathyabama University, Chennai, India. He has 13 years of experience in Teaching. He has also held various responsibilities as a part of his research. He has published around 8 research papers in journals and conferences. He has written books for some of the universities. His research area is Software Engineering and also interested in Data Base Management Systems and Data Warehousing and Mining.

S. Srinivasan received the Ph.D degree in Computer Science & Engineering from the Sathyabama University, Chennai, and M.Tech. in Computer Science & Engineering from the Indian Institute of Technology, Chennai, and M.B.A. in Systems & Finance from Sathyabama Engineering College, University of Madras, Chennai, and the M.Sc. in Mathematics from the Gobi Arts College, Gobichettipalayam, Bharathiar University, Coimbatore. He is presently working as Professor and Head, Department of Computer Science and Engineering , Anna University, Regional Centre, Madurai. He has 20 Years of experience in Teaching and Research. He has also held various positions and responsibilities in Technical Institutions. He is acting as expert member at various universities in various capacities.. He has published more than 40 research papers in journals, books, conferences, and workshops. His research interest includes text mining, data mining & data warehouse, and Software Engineering.