Evaluation of Software Components in Embedded Real-Time Systems

Student: 
Muhammad Zaheer  
MSc Computer Science with Software Engineering Profile  
Department of Computer Science and Electronics (IDE)  
Mälardalen University, Sweden  
mzr05001@student.mdh.se

Supervisor: 
Frank Lüders  
Department of Computer Science and Electronics (IDE)  
Mälardalen University, Sweden  
frank.luders@mdh.se

Examiner: 
Ivica Crnkovic  
Department of Computer Science and Electronics (IDE)  
Mälardalen University, Sweden  
ivica.crnkovic@mdh.se

Master Thesis, 20p (D Level)
# Table of Contents

1 INTRODUCTION ................................................................................................................................. 6

2 COMPONENT-BASED SOFTWARE ENGINEERING ............................................................................... 8
   2.1 SOFTWARE COMPONENTS ................................................................................................................ 8
       2.1.1 Interfaces .................................................................................................................................. 9
       2.1.2 Contracts ............................................................................................................................... 10
       2.1.3 Patterns .................................................................................................................................. 10
       2.1.4 Frameworks ............................................................................................................................ 11
   2.2 SOFTWARE COMPONENT MODELS AND TECHNOLOGIES ......................................................... 12

3 EMBEDDED REAL-TIME SYSTEMS (ERTS) ....................................................................................... 16
   3.1 REAL-TIME SYSTEMS ..................................................................................................................... 16
       3.1.1 Soft real-time systems ......................................................................................................... 16
       3.1.2 Hard real-time systems ......................................................................................................... 17
   3.2 DESIGNING OF COMPONENT-BASED REAL-TIME SYSTEMS ...................................................... 17
   3.3 EMBEDDED REAL-TIME SYSTEMS (ERTS) .................................................................................. 19
   3.4 SOFTWARE COMPONENTS IN EMBEDDED REAL-TIME SYSTEMS ................................................. 20

4 COMPONENT-BASED EMBEDDED REAL-TIME SYSTEMS (ERTS) .................................................... 22
   4.1 THE COMPONENT OBJECT MODEL (COM) .................................................................................. 22
   4.2 COMPONENT TECHNOLOGIES FOR ERTS ................................................................................ 23
       4.2.1 Koala Component Model ........................................................................................................ 24
       4.2.2 SaveCCM .............................................................................................................................. 24
       4.2.3 PECOS ................................................................................................................................... 25
   4.3 SOFTWARE COMPONENT SERVICES FOR ERTS ...................................................................... 26
       4.3.1 Logging ................................................................................................................................. 26
       4.3.2 Execution Time Measurement ................................................................................................. 27
       4.3.3 Synchronization ..................................................................................................................... 28
       4.3.4 Execution Timeout .................................................................................................................. 28
       4.3.5 Vertical services ...................................................................................................................... 29

5 CONCLUSION AND FUTURE WORK .................................................................................................. 31

6 REFERENCES ........................................................................................................................................... 33
List of Figures

Figure 1: Common Language Runtime (CLR) ..............................................................................................14
Figure 2: Koala’s Graphical Notation .............................................................................................................15
Figure 3: Design model for real-time components ......................................................................................18
Figure 4: Washing machine an embedded and real-time system .............................................................20
Figure 5: Typical format of COM interface nodes ......................................................................................23
Figure 6: Instance creation and dynamic loading of code in COM .............................................................23
Figure 7: Pneumatic Positioner (TZID) ..........................................................................................................25
Figure 8: A logging service proxy .................................................................................................................27
Figure 9: Generating a proxy object for a component service .....................................................................30
Abstract

Today, computers have become more popular in every field of life especially in industries, robots, automobiles and health care systems of hospitals. This large growth of computers in our daily life is causing so many problems related to softwares. To overcome these problems in softwares, the software engineers are adopting new approaches which is called Component-based software engineering (CBSE). CBSE is widely using in the development of component-based software systems. The main advantage to use the component based software engineering is that, we can increase the efficiency and reduce the time to complete the software’s projects. The development of components is growing technique which is used to build the real-time systems. The trend to use the embedded real-time systems in household products has been increased for last 10 years. In embedded real-time systems the hardwares and softwares are joined in this way that they perform better function. In development of embedded real-time systems the different software component model techniques are used. In this Master thesis we evaluate the software components and their models in embedded real-time systems.
Acknowledgements

First of all, I would like to thank my supervisor Dr. Frank Lüders for his assistance and morally support during my work with this thesis. I like to thank my examiner Dr. Ivica Crnkovic for his support in my entire degree program. Thanks also to my all friends and colleagues respectively in Bjurohovda and Department of computer science and electronics, Mälardalen University Västerås, Sweden. Finally, I wish to thank my family and friends who have been with me (on emails, cards, phones) all the time during last two years of my stay in Sweden.
1 INTRODUCTION

Today, computers have become more popular in every field of life especially in industries, robots, automobiles and health care systems of hospitals. This large growth of computers in our daily life is causing so many problems related to softwares. To overcome these problems in softwares the software engineers are adopting new approaches which is called Component-based software engineering (CBSE). CBSE is widely using in the development of component-based software systems. In component-based software systems, the different smaller components are manufactured at separate places and then do integrate theses components with each other to get final product.

The major advantage to use the component based software engineering is that, we can increase the efficiency and reduce the time to complete the software’s projects. The development of components is growing technique which is used to build the real-time systems. The most common component technique is Microsoft’s Component Object Model (COM) [18] which was developed in 1993. Software component models are using in development of desktop applications e.g. JavaBeans [10] and ActiveX [26]. These component models are also using in development of distributed information systems e.g. EJB (Enterprise Java Beans) [27] and COM+ [12].

Nowadays, embedded real-time systems are using in every product from washing machines to nuclear power plants. Li and Yao describe embedded systems are those systems in which hardware and softwares are joined so strongly to perform the better function. We know that above describe different component models are not using in development of embedded real-time systems. The scientists and different organizations are developing new component models to build the embedded real-time systems. These component models are PECOS [23], SaveCCM [22] and Koala Component Model [16]. PECOS technology is mainly used in development of field devices, e.g. pneumatic positioner (TZID). SaveCCM is used to build the critical embedded real-time systems e.g. vehicular control systems. The koala component model is developed by Philips. This technology is used in household products such as microwave, VCRs and televisions.

We have divided our thesis report in four major Chapters. The first chapter is about introduction. In second chapter we will describe about
component-based software engineering, software components, interfaces, contracts, patterns, frameworks and about software component models and technologies. In third chapter we will describe about embedded real-time systems, designing of component-based real-time systems and about software components in embedded real-time systems. In the fourth chapter we will describe about component based embedded real-time systems, the component object model, component technologies for ERTS and about software component services for embedded real-time systems.
2 COMPONENT-BASED SOFTWARE ENGINEERING

Component-based software engineering (CBSE) is a technique to develop the software systems by connecting the different components. It is very expensive and difficult procedure to develop and maintain the software components. If we look on the other side, it is very inexpensive to distribute and reproduce the software components. The organizations which handle these software systems are facing many new challenges like, to produce the cost effective software systems and then to maintain these systems [1, 2]. Now, we will discuss about the software components, software component models and software component services.

2.1 Software Components

Software components are the major concept in the component-based software engineering (CBSE). These software components or pieces are connected in this way that, we get superior component. Software component is defined by Szyperski [3]:

“A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party”.

There are many different definitions are existing to define the components. These definitions are related to the CBSE in order to the different ways and they only focus on the different parts of software engineering. According to the definition a component is a smaller part of the composition, and these components should be connected to each other and then these combine components put together into system. If the component has the following different elements then we can describe the accurate maintenance, integration and updating of the component [4]:

- A component consists of a set of different interfaces, which are supplied or required from the environment. These components prefer to interact with other components instead of to interact with the predictable software entities.
• An interface of a component plays a role like bridge to assemble the both components. By this interface, we can join an executable code to the code of other component.

The quality of component can be improved by adding the following elements in the requirement of the component.

• The requirement of different features which are not well designed. These features are required and supplied.

• The quality of a component can be improved by the code validation, which is responsible to validate the planned connection between the components.

• The quality of component can be improved by the proper documentation which includes the use cases and design informations.

2.1.1 Interfaces

An interface of a component is defined as "a specification of its access point" [3]. The component provides the services to the clients through interfaces and client can access the services of components by these interfaces. If a component performs different services then it would be must that this component has multiple access points. There are two types of a component interfaces, firstly export interface and secondly import interface [4]. Those services which are provided to the environment by a component is called through these interfaces are called exported interfaces. The imported interface is defined as, those services which are provided to the component by the environment.

2.1.2 Contracts

The exact design of a component’s behavior is carried out by contracts. The contracts and interfaces have diverse ideas; an interface consists of different processes which indicate those services which are presented by the components. In the other side a contract indicates the communication between two or more than two
components. According to Ivica Crnkovic and Magnus Larsson, contracts indicate the following features among the components in terms of [4]:

- The set of participating components;
- The role of each component through its contractual obligations, such as type obligation, which require the component to support certain variables and an interface, and casual obligations, which require the component to perform an ordered sequence of actions, including sending messages to the other component;
- The invariant to be maintained by the components;
- The specification of the methods that instantiate the contract.

2.1.3 Patterns

Ivica and Larsson [4] define patterns as, “a recurring solution to a recurring problem”. Actually patterns are responsible to describe the connection between the structure and method of a system. Patterns do not describe the free and single module. Only design patterns define the structure, method and implementation of the component at bottom level. On the basis of design, patterns can be divided into three main categories, which are describing in following as:

**Architectural patterns** describe the structure and design of that system which deals with the large components.

**Design patterns** describe the structure and design of the associate systems which deals with the components. Design patterns are also used in the development of component based systems. Design patterns make it easy to locate the reusable parts in the system and they also use to know the inner behavior and structure of the component though they are used in the development of components.

**Idiom patterns** are also called low level patterns. They are dependent of the programming language which has been chosen.
2.1.4 Frameworks

A framework is defined as, “a skeleton of an application which can be customized by an application developer” [17]. The main concept of frameworks is this that when computer systems are designed, several results are obtained. These results can be implemented as reused in any other environment. Thus a framework can not be used in specific environment but it can be reused in any other similar environment after getting some changes in the framework.

Frameworks and patterns have very close relationships to each other. Both frameworks and patterns can be reused in similar environment. The main key role of frameworks is to force the different components in performing of their tasks which are organized by the frameworks. There are many examples for component frameworks, but the most common example for component framework is Visual Basic which is developed by Microsoft. In Visual Basic before development, all forms are empty and the users have to add the components to build the forms. Other examples of component frameworks are COM and EJB.

2.2 Software Component Models and Technologies

As mentioned by its name that software component model specifies standards for composition and interaction between software components [9]. Today there are many software component technologies are using in industries, like as: JavaBeans [10], COM [11], .NET [14], CCM [4, 9] and The Koala component model [16].

JavaBeans are very famous software component and it is written in the Java programming language. The JavaBeans is a product of Sun Microsystems, which define the JavaBeans as, “reusable software components that can be manipulated visually in a builder tool”. An example of JavaBean is given in the following [10]:

```java
// PersonBean.java

public class PersonBean implements java.io.Serializable {
    private String name;
    private boolean deceased;
}
// No-arg constructor (takes no arguments).
public PersonBean() {
}

public String getName() {
    return this.name;
}
public void setName(String name) {
    this.name = name;
}

// Different semantics for a boolean field (is vs. get)
public boolean isDeceased() {
    return this.deceased;
}
public void setDeceased(boolean deceased) {
    this.deceased = deceased;
}

// TestPersonBean.java
public class TestPersonBean {
    public static void main(String[] args) {

        PersonBean person = new PersonBean();
        person.setName("Bob");
        person.setDeceased(true);

        // Output: "Bob [deceased]"
        System.out.print(person.getName());
        System.out.println(person.isDeceased() ? " [deceased]" : "");
    }
}

COM stands for component object model. It is developed in 1993 by Microsoft. The basic facility of COM is that it can be written in any
programming language. COM can support two techniques, firstly Containment and secondly aggregation [4]. A COM object can hold the other COM object by using the containment technique. Aggregation is a very difficult technique; by using this technique the outer object can describe the functions of the inner object. Its main disadvantage is this that it asks the source code of inner and outer to change them. The other technologies which provide support to COM are COM+ [12] and DCOM [13].

Microsoft .NET [14, 15] is the most recent component technology from Microsoft. The .NET framework is a primary component which supports a lot of programming languages and used to build and run the different software applications. It makes the easiest to construct, deploy and secure the different applications, which were not so secure before this. The .NET framework consists of CLR (Common Language) and class libraries. CLR gets control on the different services like as, Integration of two or more languages, Memory and Provide the security.

Figure 1: Common Language Runtime (CLR) [15]

Another latest component model is CCM which stands for CORBA Component Model. CCM is developed by OMG (Object Management Group) in October 1999. It is built on the experience of CORBA, EJB (Enterprise Java Bean) and JavaBeans. CCM plays a very important role in development of applications by connecting the different parts. CCM component contains at least one of the following categories, Service Components, Session Components, Entity Components, and Process Components.
The Koala component model [16] is developed by the Philips research laboratories. It is specially developed for the consumer products which are manufactured by the Philips such as telephones, washing machines, televisions and VCRs. Today there are many embedded systems are developing by the Koala component model’s technique. A Koala component consists of different interfaces, and it interacts with the environment by using of these interfaces. The following figure is representing the Koala’s graphical notation. According to this figure, it consists of components like IC chips and the configuration is like electronic circuit. Interfaces represent as pins of the chip and triangles represent the function call. HIP driver is connected to the fast 12C and HOP is connected to the slow 12C service.

**Figure 2: Koala’s Graphical Notation [16]**
3 EMBEDDED REAL-TIME SYSTEMS (ERTS)

3.1 Real-Time Systems

Stankovic [5] defines real-time systems as, “those systems for which the correctness depends not only on the logical results of the computation, but also on the time at which the results are produced”. According to this definition, it is not important that the accurate results are obtained in real-time systems. But in widespread, real-time systems are those systems at which the accurate computation results are obtained at particular time. In other words real-time systems [28] do not depend on what outcome we have but also on what time the outcome comes. Some important terms in real-time systems are tasks, response time, execution time and deadlines. Response time is defined as the time in which system gives output after taking input. Deadline is the time required for system to finish or submit something in a particular time. A deadline can be classified into two groups, hard deadline and soft deadline. Hard deadline is that deadline which must be met. If the hard deadline does fail to meet the system requirements, it means that system is in error state or failed. Soft deadline does not need to meet at all times. The results can be used although, they come late. There are two types of real-time systems, hard real-time systems and soft real-time systems. We are describing both kinds of real-time system in the following:

3.1.1 Soft real-time systems

Soft real-time systems are those systems which give permission to miss the deadline as these systems are not safety-critical systems. There is not given the assurance of the systems temporal behavior at every time in soft real-time systems. Here, we consider an example (Vehicle’s window control system) to understand soft real-time systems. If the window of a vehicle does not close or open in time or not opens correctly, it is not a disaster. But soft real-time system does not allow to be happened or repeated these things every time.
3.1.2 Hard real-time systems

Hard real-time systems are those systems which do not give the permission to miss the even single deadline or incorrect output as these systems are safety-critical systems. Hard real-time systems are describing by the two main examples, nuclear power plant and health care systems in hospitals. In nuclear power plants, if a single deadline is missed or failed it can cause a terrible damage, death and loss of property. If a single dead line is missed during the fitting of heart pacemaker in the human body or during CT scan, it can cause the death of a patient.

3.2 Designing of Component-Based Real-Time Systems

We will describe in this section that how to design a component-based real time system? Before starting the development of a system, we have to need the specification of a system which is gathered from customer. After collecting the required information from customer, we make the design so simply and correct for presentation of our system. According to the following figure, design model for real-time components is divided into many stages.
In the first stage of above figure, design model for real-time components start by input which is in the shape of system specification. In the second step top-level design, which breakdown the system and converts into components. In this phase the designer uses the help from component library and then designs the system after selection from running components. In third step, we get the detailed design of components that which components are right for integration. In the selection of components, it is must to consider the features of real and non real-time systems. In fourth step, the extendibility and protection of design is checked by the implementation of particular architectural analysis methods. In fifth step, the selected component is checked by using scheduling and interfaces that either the component is right for the system. In this phase, if it is needed to change the component or alter the properties of component then we can develop a new component. The scheduling of a component can be repeated many times to choose the best
component. After the selection and attestation of a Component, the new component is sent to the component library. For more details about the designing of real-time component can be seen in the [4].

3.3Embedded Real-Time Systems (ERTS)

Today embedded real-time computer systems are using in many products as main component. These products are microwave oven, washing machines, cell phones, health care systems and robots. The IEEE’s glossary [6] defines an embedded computer system as:

"A computer system is part of a larger system and performs some of the requirements of that system. For example, a computer system is used in an aircraft or rapid transit system".

Li and Yao [7] define embedded system as:

“Embedded systems are computing systems with tightly coupled hardware and software integration that are designed to perform a dedicated function. The word embedded reflects the fact that these systems are usually an integral part of a larger system, known as embedding system”.

According to the Li and Yao’s definition embedded systems are those systems in which hardware and software are joined together so strongly to perform the better function. In the second part of definition the authors describe that embedded systems are very important part of the larger systems.

![Image of a washing machine]

Figure 4: Washing machine an embedded and real-time system [8]
3.4 Software Components in Embedded Real-Time Systems

In these days, software components are using so swiftly in the development of many systems. One thing should be kept in mind that these software components or software component models are not usually using in the development of embedded real-time systems. So, the scientist and engineers are developing new models of software components to develop the embedded real-time systems. The most common examples for these types of models are Koala component model [16], SaveCCM [22] and PECOS [23]. We will describe in detail about these component models in below section 4.2(component technologies for ERTS).

Anders Möller [25] and his colleagues describe in their paper about the requirements of component models which are used in the development of embedded real time systems. These requirements are described in following as:

- **Analysable**
  
  Analysable provides us facility to analyze that every component is tested in respect of timing and memory management.

- **Testable and Debugable**
  
  The most common technique which is used in component models is testing and debugging. It describes all the feature of components and then it is tested before to integrate in the system.

- **Portable**
  
  Portable is very important feature in the development of component. The components should be so platform independent that they could be easily ported with the other hardwares and operating systems easily. Another feature is that components should be independent in this stage that they used the framework during the run-time.

- **Resource Constrained**
  
  If we look on the cost and weight of a product that today those products are more common in market which have less weight and price. So when we develop the components, they should be light weighted and minimized in their infrastructure.

- **Component Modelling**
The Component modelling only bases on the standard modelling language. That modelling language is UML (unified modelling language). This modelling language is technically very strong and its formats can be maintained by developers which belong to third-party.

- **Computational Model**

The passive components should be preferred because they don’t have their own threads for the process of execution.
4 COMPONENT-BASED EMBEDDED REAL-TIME SYSTEMS (ERTS)

4.1 The Component Object Model (COM)

In modern age of computer technology, the development of component is the emergent technique which is used to construct the systems. The mainly component technique is Microsoft’s Component Object Model (COM) [18], which is using to build the desktop applications. The major advantage of Com is that its interfaces are identified as a separate from components, the one which apply and the other which use them. The interfaces are very important in object oriented designs. Interface plays a role like bridge between two components, it connects the both components then they can exchange the information to each others.

COM uses its own language, which is known as IDL (Interface Description Language) [19]. IDL is used to describe the different interfaces of components. IDL has syntax like C language but this language does not describe execution of a component. We can explain the interfaces and COM classes in IDL. We explain the COM interface nodes in the following figure:

![Figure 5: Typical format of COM interface nodes][9]
4.2 Component Technologies for ERTS

As, we have discussed above that with the passage of time computers are growing rapidly in every field of life from manufacturing of toys to nuclear power plants. This large growth of computer systems in our daily life is producing so many problems relating to software systems. Today, developers are fully concentrating on the development of component based software systems. In component based software systems the different components are manufactured and then integrate these components with each other to get the best software. The technique which are using is called Component-based software engineering (CBSE).

Nowadays there are many embedded real-time systems (ERTS) are manufacturing in industries. The component technique is widely using in the manufacturing of these kinds of systems. In the present day the component is using only to develop the desktop applications but the components should be used in the development of critical ERTS. In below, we will describe about some component technologies which are using in the development of critical ERTS.
4.2.1 Koala Component Model

Koala component model [16] is developed by Philips. It is mainly using in production of home appliances which are developed by Philips, such as microwave, VCRs and televisions. Basically koala component model is used to develop the embedded real-time systems. We have already discussed in detail the koala component model in section 2.2 (software component models and technologies).

4.2.2 SaveCCM

SaveCCM [22] is a component technology which is used in the development of critical embedded real-time systems. This model is basically designed for manufacturing of embedded real-time based vehicular control systems. Generally, SaveCCM is composed of four elements which are components, switches, assemblies and run-time framework.

4.2.3 PECOS

PECOS [23] stands for Pervasive component systems. PECOS is also a component technology which is developed for embedded real-time systems. This technology is used mainly in the development of field devices. An example of field device is Pneumatic positioner (TZID). This device is developed by ABB Corporation and it is used to handle the pneumatic actuator which are attached to regulators or taps. Below figure is showing a pneumatic positioner (TZID).
PECOS was a joint venture of following educational and industrial colleagues [24].

- **ABB**: ABB AG - Germany
- **FZI**: Forschungszentrum Informatik an der Universitaet Karlsruhe - Germany
- **OTI**: Object Technology International AG - The Netherlands
- **UNIBE**: Software Composition Group, University of Bern - Switzerland

### 4.3 Software Component Services for ERTS

Software components models are using swiftly in the development phase of different softwares. They are using especially in the development of desktop applications and different distributed systems. If we look on the other side, we will see that software component models are not using so repeatedly in the development of embedded and real-time systems. The main factor is time limit and minimum inputs. So, scientists and engineers are researching to develop component models which are easily useable in embedded and real-time systems.
In this section, we will discuss about the different services of components which deals with the different problems of the embedded and real-time systems. These services consist of logging, execution time measurement, synchronization, execution timeout and vertical services [20]. We will describe these services in the following:

### 4.3.1 Logging

The series of relation between two or more than two components can be traced through logging service. Below figure is showing the implementation of logging service. In this figure, C1 and C2 are two objects. The interface IC2 is placed by the object C2 for execution. Now, the logging service will be applied on this interface IC2.

![Figure 8: A logging service proxy [9]](image)

### 4.3.2 Execution Time Measurement

Execution time measurement is applied to check all the informations about the different continuing operations which are related to the time execution. By using this service, we can collect all types of execution time measurements like as best case, average case and
worst case. The service of execution time measurement is representing in the following configuration case [20]:

```xml
<interface name="IC2">
  <service type="Timing">
    <measurement type="Mean" />
    <measurement type="Worst" />
  </service>
</interface>
```

### 4.3.3 Synchronization

According to [20] "A synchronization service allows components that are not inherently thread-safe to be used in multi-threaded applications". A synchronization rule can be applied for a particular process or group of different processes. The most popular strategy of synchronization is called mutual exclusive. This strategy calls only one process and do block the all other processes. The requirement of read or write strategy can be seen as following:

```xml
<interface name="IC2">
  <service type="Synchronization" policy="RWPolicyX"/>
  <operation name ="DoSomething" type="Write"/>
  <operation name ="WriteData" type="Write"/>
  <operation name ="ReadData" type="Read"/>
</interface>
```
4.3.4 Execution Timeout

The service execution timeout is used to make sure and to call that the process of a component is finished in specified period of time or deadline. It does not only make a call when the operation is finished but it gives us indication, when the process of a component is not finished in particular time or failed. The service of execution timeout is also given in the following configuration file [20]:

```xml
<interface name="IC2">
    <service type="Timeout" deadline="10ms" fail="Terminate"/>
</interface>
```

4.3.5 Vertical services

Vertical services is an extra type of services, these all services are very constructive for embedded and real-time systems. Vertical services [21] are that service at which more services are expected at particular application areas. Below figure is describing the generation of proxy object.
Figure 9: Generating a proxy object for a component service [20]
5 CONCLUSION AND FUTURE WORK

This Master’s thesis confers about the evaluation of software components in real-time embedded systems. The use of components in the development of softwares is emergent technique. Actually, in this thesis we have found a literature study about the software component in real-time embedded systems. Today, many companies are developing distributed softwares. The software component technique is very useful in development of distributed softwares. This technique is also used to develop the real-time systems. The most common components which are using in the development of real-time systems are Microsoft’s component object model (COM), COM+, ActiveX, JavaBeans and Enterprise Java Beans (EJB). One thing is very important in the respect of these above software components that these components are not used in the development of embedded real-time systems.

Today, embedded systems are really increased in household products such as microwave, refrigerators, televisions and VCRs. In the development of these embedded systems, Koala software component model is used. Koala component model is newly technique which is using very effectively in the development of household products. This component model is developed by the Philips. PECOS technology is also used in development of different field devices. SaveCCM is another software component model which is used to develop the critical embedded real-time systems e.g. vehicular control system. In chapter 1 of this thesis, we have described the introduction of the whole thesis. In this chapter, we have tried to explain the detailed summary of our thesis.

In this report, we have discussed about the component-based software engineering. When softwares are developed, when we run these softwares on the system there are some problem are created related to these softwares. Basically component-based software engineering (CBSE) was developed to overcome these problems in softwares. CBSE is widely using in the development of component-based software engineering. In this technique different components are manufactured on different places and then assemble them to get a final product. We have discusses in this report about the software component models and technologies which are used to
develop the embedded and real-time systems. The different software component services in embedded real-time systems are described in detail.

In this thesis, the literature study about the software components present the use of software component models in embedded real-time systems. The main aim of this study about software component is that we can implement some tasks by use of run time services in embedded real-time systems. In future work, we can investigate the use software component models in industrial control systems by controlled experiments.
6 REFERENCES


