Design and Implementation of Context Program Compiler for developing Context Aware Applications in Pervasive Computing Environment

Thyagaraju G. S., Umakanth P. Kulkarni

Abstract— With the advancement of pervasive computing, sensors technology and the wide deployment of wireless communication, there is an increasing demand for the context aware computing application. Contextual presentation is an emerging technique that has huge commercial possibilities .The theory behind the applications is complex and this make the implementation non trivial. Although some good applications / devices have been built no general solutions are available. There is no programming language or scripting language available to find solutions to context based problems. In this direction we are developing a generic context compiler and generic context programming language using which one can write and execute programs to develop any context aware applications. In this paper we are presenting the design and implementation of context program compiler for developing context aware applications in pervasive computing environment.

Index Terms— ubiquitous environment , Context script, pervasive, compiler, bnf .

I. INTRODUCTION

The context –awareness is one of the most important technologies in pervasive computing, which can facilitate information acquisition and execution by supporting interoperability between user, sensor and devices based on user's context. The proposed Context Program Compiler (CPC) facilitates to write context script to establish context association between user, sensors and devices. One can write any script to describe the three aspects of context like (1) where you are; (2) who you are with; and (3) what resources are nearby by configuring sensors, devices, user and associating set of rules among them.

The context program written in context programming language (CPL) based on specific grammar have features like sensors initialization , device configuration ,user profile configuration , rules association , and conflict resolution which are essential procedures for context aware applications. The CPL is very near to english and is inspired from the context script language(CSL) designed by Yongki Kim and Jaewoo Chang [1,2]. It is object based programming language. When the context program is executed we get a series of actions (interactive as well as non interactive) to provide a necessary context services for user based on his/her context. The Compiler has been tested for simulated input and output.

In section 2 we are describing the related work .Section 3 gives the complete description of the proposed architectural design of the context program compiler. Section 4 describes the different aspects of compiler like BNF rules of a context program language, tokens, syntax and features of the language. Section5 discusses the implementation and validation of the compiler and language with example. Section 6 concludes with feature work possibilities.

II. LITERATURE SURVEY

In this section, we introduce some related work on context-aware computing applications and framework [3].

RCSM- Arizona State Univ. Presented Reconfigurable Context-Sensitive Middleware (RCSM) which made use of the contextual data of a device and its surrounding environment to initiate and manage ad hoc communication with other devices. In addition, they provided a context definition language called CA-IDL (Context-enabled Interface Definition Language). That is, context tuples for defining context can be represented as Context tuple : < a1, ..., an, tm>. Here, n means the number of unique context data, ai means the value of the ith context data, and tm means the time for tuple creation [4].

INRIA- in France proposed a general infrastructure based on contextual objects to design adaptive distributed information systems in order to keep the level of the delivered service despite environmental variations. The contextual objects (COs) are mainly motivated by the inadequacy of current paradigms for context-aware systems. The use of COs does not complicate a lot of development of an application, which may be developed as a collection of COs. The COs are defined as CO (id) : <Variant Vx Attribute Ai: value, Attribute Aj: value, ...>[5].

COP- Context Oriented Programming (COP) enriches programming languages and execution environments with features to explicitly represent context-dependent behavioural variations [6].

CRL - Context Request Language (CRL) provides a solution for an intelligent technique to context sensing. This language allows applications to specify inferences for monitoring contextual information.

Manuscript received May 30, 2011.

Thyagaraju G. S., Dept of CSE, VTU/SDMCET/ Dhavalagiri, Dharwad, Karnataka, India, Mobile Number: +919480123526, (e-mail: thyagaraju_gs@yahoo.co.in).

Umakanth P. Kulkarni, Dept of CSE, VTU/SDMCET/ Dhavalgiri, Dharwad, Karnataka, India, Mobile Number : +919448915301 (e-mail: upkulkarnu@yahoo.com).

The use of predicate calculus in this language definition enables users to utilize context aggregation and precise control over contextual information. CRL forms an integral part of our new Web Service architecture, CRL framework, which supports dynamic reflective reconfiguration, asynchronous communication, and predefinition of context composition through CRL-rule definitions [7].

COPAL - is a runtime context provisioning middleware that, via a loosely-coupled and composable architecture, ensures context information from wireless sensor networks and other sources can be processed for the needs of context-aware applications. COPAL-ML[8] is a macro language that extends Java programming language and is tailored for the application development using COPAL. Its main task is to reduce development efforts, hide the inherent complexity of COPAL API, and separate concerns of context-aware application from underlining wireless sensor network.

CSL – A scripting language where in the user has to use predefined set of script commands to create a context script. The context script is made up of different components like context objects, context definition and destruction followed by context instance activation and deactivation [1, 2].

The proposed system is a framework for design and executing a context programs. Here the user has to write a program to establish an association among sensors, devices and users profile. One can configure sensors and devices, can write a code for resolving conflict among users, devices and sensors.

III. ARCHITECTURAL DESIGN OF THE PROPOSED SYSTEM

Context is [9] any information that can be used to characterize the situation of any entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. The overall architecture of the proposed context programming environment for realizing various context aware applications is divided into three components such as editor, compiler and action.

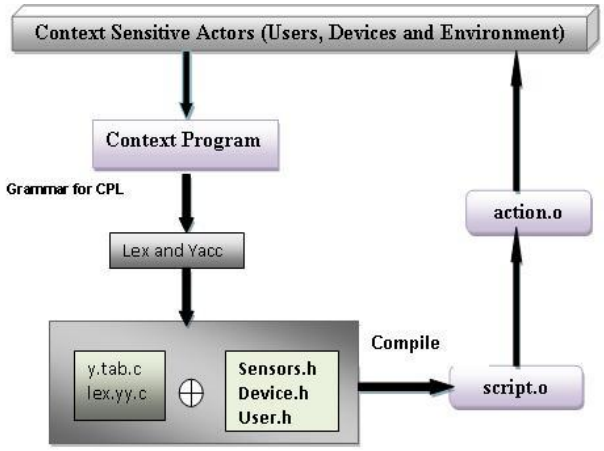

Fig1 Architectural design of proposed system

Fig1 shows the architecture of the proposed system; here actors include person, location, devices, environment, etc. Based on the initial and required state of actors a context program which uses the context program language is written in the editor and is compiled. After compilation based on context of actor as specified in the program appropriate action will be generated as the output.

A. Context Programming language

In order to develop a variety of context-aware applications in an effective manner under our proposed architecture, a context program language is required to describe both various decisions on context-awareness and appropriate procedures according to the decision. Thus, here we introduce a new Context Program Language (CPL) using standard BNF rules. CPL represents a given context as a standard syntax in a clear and precise manner as well as can provide users with functions to describe a variety of contexts with a general purpose.

A.1 General Structure of a Context Program

```
ContextBegin  C1
{       Context Actors/Entities declaration;

     Context  Statement1;
    Context Statement2;
     Context  Statement n;
}
ContextEnd
```

Fig 2 General Structure of a Context Program

Fig2 gives the structure of Context Program. It is divided into different sections like *a.ContextBegin and ContextEnd b.Context Actor declaration and c. Context Rules or Association* .ContextBegin is the statement from where the execution of the context starts which is followed by the context identifier that is the name of the context and it ends with the statement ContextEnd. User can declare context actors like sensor and devices at the beginning and use those entities for establishing context association with user. Context statements are used to describe the situation through establishing context association among sensor, device and user for given context. CPL has five fundamental context tokens.

Context Tokens: Context identifiers, Keywords, Operators, Separators, Constants.

Context Operators: ">"| "<" | ">=" | "<=" | "==" | "!=" |"&"|"|"

Separators: BlankSpace" " (comma), ;(Semicolon).

Context Identifers: It is a sequence of letters .Upper and lower case letters are different .It may have any length

Context Keywords: Table1 gives a list of identifiers which are reserved for use as keywords and may not be used otherwise:

Table 1: Context Keywords.

Category	Keywords
Context Entities	user , device , sensor , tv ,ac , fan , door , window , curtain , bulb , musicsystem , heater , radio , tubelight.
Sensor Instances	location, livingroom , bathroom, meetingroom, conferencehall , busstand, airstation , railstation ,time , am , pm , am , pm , attemp , humidity ,atm pressure , bodytemp , bplevel, sugarlevel, verylow , low , norm , high , veryhigh , summer , winter , spring ,rainy ,season, winter , summer , spring , rainy ,vacation , exam time
User mood	relax, cheerful, depressing, exciting, disturbing, comforting
User Activities	sleeping , walking , reading , bathing , running , eating , sitting , resting , talking , playing , entering , exiting , reading , exercise, sitting , driving
User Age	old , youth , young , child , adult
Sex	male ,female
Role	father ,mother , son , daughter , grand father , grand mother ,aunt ,uncle , chairman , teacher , student , manager , president , prime minister , principal , hod ,celebrity , singer , dancer , sports person , hostile, tourist ,passenger, driver , conductor--
Connecter	is , and ,or , on ,are,-----
Actions	action, switch ,on ,off , set , get, ----
Program	contextbegin , contextend
Conditional	if , while , for, when

B. Language Features

- It is an object based structured language.
- It uses statements and expressions.
- It uses conditional statements to control the flow of execution.
- The CPL is a case sensitive which differentiates between upper and lower case alphabets.
- Uses characters and strings.
- Entity types include device and sensor.
- Specified format – some keywords such as time, temp have predefined format .User should specify the values for those variables in that format only.
- External Files has to be included as a part of header files.
- Comments are not supported.
- Space is a counting factor.

IV. BNF GRAMMAR FOR CPL

Backus-Naur notation (more commonly known as BNF or Backus-Naur Form) is a formal mathematical way to describe a language, which was developed by John Backus (and possibly Peter Naur as well) to describe the syntax of the Algol 60 programming language. It is used to formally define the grammar of a language, so that there is no disagreement or ambiguity as to what is allowed and what is not. In fact, BNF is so unambiguous that there is a lot of mathematical theory around these kinds of grammars, and one can actually mechanically construct a parser for a language given a BNF grammar for it.

A. BNF Rules of a Context Program Language

Context_declaration = "ContextBegin" Context_identifier "{" <Field_declaration> "}" "ContextEnd"

Field_declaration = <variable_declaration> <Context_Statements> ";"
context_entity_declaration = ”sensor” <sen>”;” ”device” <dev> “; “
sen = “lightIntensity” | “status” |”noiselevel” |”temp ”| ”humidity” | “pressure” | “userid” |”location” | “chargelevel” |”time” | “usermood”|”useractivity”|”username” ----------
dev = "TV" | "AC" | "Fan" | "Tubelight" | "Heater" | "MusicSystem" | "Radio" |”Door”|”Window”| ”Curtain” | ”Laptop” | ”Camera” | ”Mobile” | “ LCDProjector” |----------
context_statement = statement_block | if_stmt_block | while_stmt_block| for _stmt_block;
statement block = "action" "=" <state> ” “< state> “ “ <device> | (<state>) * <device> “ [<dattrib> ,<dval> ”]” “;”
if_stmt_block = "if" "(" expr ")" "{" statement_block "}"
while_stmt_block = “while” “(“ expr “)” “{ statement_block “}”
for_stmt_block = “for” “(“ expr “)” “{ “ statement_block “}”
expr = sen <op1> <sval> | dev <op1> <dval> | user <op1> context_identifier |expr<op1>expr
state = “action" | "do" |"occurs" | "set" | "get" | "switch" | "frequency" | "volume" |"channel" | "when" |”on” |”off”|”
op1 = ">"| "<" | ">=" | "<=" | "==" | "!=" |”&”|”|” and” |”or” |”is”|” ----
Context_identifier = “a...z, $, _” < “a...z, $, _, 0...9, unicode character over 00C0 “>

sval = “verylow”| “low” | “normal” | “high” | “veryhigh” | “Open” | ”Close” | ”ON” | ”OFF” | ”LivingRoom” |”BedRoom” | “Kitchen “ | “BathRoom” | “Office” | “Dining Hall “ | “ Hall “ | “MeetingRoom “ | “ClassRoom “ | “ConferenceHAll” |”Library” | “CollegeCampus” | “Bustand” | “Hospital”|”AirStation” | “RailStation” |”Relax”|”Cheerful”|”Happy”|”Sitting”|”Reading” |”EarlyMorning”|”AfterNoon”|”Evening”|”Night”|”MidNight”|
---------- ----------
<N>”C”|<<Date><HMS><tm>|Context_identifier

dattrib = “intensity” | “speed” | “state” | “volume” |”mode” | “brightness” | “Channel” “actemp” | “Frequency “ | “Power”

dval = “verylow” | “ low” | “ normal” | “high” | “veryhigh” | “dim” | “bright” | “Open” | “Close” | “ON” | “OFF” | Context_identifier
N = [0-9][0-9]

HMS = [0-9][0-9]":"[0-9][0-9]
tm = "am" | "pm" | "AM" | "PM"

Table 2: Syntax and examples of a different CPL Keywords.

keyword	Syntax	Example
username	sensor username;	username is rahul
time	sensor time;	While time is earlymorning
temp	Sensor temp;	if temp is high
device	device devicename;	device AC ;
if	if (expression){ }	if(temp >= high)
location	sensor location ;	if location is living room
useractivity	sensor useractivity;	if useractivity is walking
usermood	sensor usermood ;	If usermood is happy ;
action	action = statements	action = switch on AC[temp,norm] ;
Expression	temp op tempkeyword	temp >= high
ContextBegin, ContextEnd	ContextBegin context_id {context_statements }ContextEnd	ContextBegin c1 { temp high; ---------- }Context End

V. IMPLEMENTATION AND VALIDATION

In the preceding sections we motivated and described the design of Compiler. Although its iterative design was guided by our findings from our formative study, we still need to determine how usable it is and whether it supports the conceptual models we discovered. We validated the compiler by answering the following questions:

- Can programmer use the compiler and accurately build context aware rules or associations?

- Can programmer use the compiler and accurately build solutions to context based problems?
- Does the Compiler support to express the conflict resolution among users, devices and sensors.

We address the above questions through programming more than 100 scenarios. Following are some of illustrations.

Procedure : Program writing has two sections 1.Declaration 2. Context Association.

1. Declaration: Here the Context Actors like sensors, devices and users has to be declared.

ex1: sensor location, activity, mood, username-----;

This declarative statement initializes the sensors for capturing .

location – returns tokens like BedRoom, LivingRoom ,etc

username- returns will returns the value of user.name.

ex2: device AC, TV,FAN,DOOR -----;

This declarative statement initializes the devices in active mode which can be further used for context association.

2. Context Association: Here the association or relationship between sensor and devices has to be established in order to provide the context aware services for user based on her context.

ex1: while (location is livingroom and username is rahul) switch on TV[channel,12];

ex2: if (location is bathroom and time is morning and user is anyuser) switch on Heater [temp, high];

Context Program Examples:

Example 1: Consider a scenario "whenever user enters into bath room (where in Heater is situated) at a time 5:00 am, if the temperature is low then the heater has to get switched on ". For this scenario context program is as follows:

```
ContextBegin HeaterOn
 { sensor location,time,temp,username ;
   device heater ,tublelight;
   if ((location is bathroom) and( username  is
      Thyagu) and (time is earlymorning) and
          (temp <= low )
      {       action = switch on heater[temp, high] ;
              action = switch on Lamp[Bright, Dim] ;
        }
 } ContextEnd
```

Example 2: Consider a scenario "user enters into the reading room during early morning say in between 5.00am to 7.00am.Table Lamp has to be switched on.

```
ContextBegin TableLamp
{
  sensor location ,time ,username;
  device TableLamp ;
    if ((location is readingroom) and (time is
      earlymorning) and (username is xyz))
   {  action = switch on TableLamp[Bright,High];
     }
 }
```

Ex3: In the morning secretary comes to the office .When she enters the room, the light is turned on and the curtain is opened .The music player plays the secretary's favorite song. After that professor arrives .Because the priority of professor is higher than that of secretary the music is changed to professors favorite .When a student enters the office and sits down on the chair in front of professor's desk, knowing that professor is having a meeting with his students , the music will be switched off.."

```
ContextBegin Office
{  sensor location ,activity,time,userid;
  device light,musicplayer;
   if ((location is office)and (time is morning))
     if(userrole is secretary)
     {  action=switch on curtain(status,open);
        action=switch on light(bright,norm);
       while(!((userrole is professor) )
         {
           action = switch on MusicPlayer (Song ,
                    favorite(secretary) );
          }  While(!useractivity is meeting)
            action=switch on musciPlayer(song ,
                    favorite (professor)) ;
        action = switch off  musicplayer;
 }
```

Table3 illustrates some more scenario and context programming examples.

Table3: Scenario and Context Programs

Scenario	Context Program

If I'm in the living room after 10pm, dim the living room lights.	Context Begin C1
	{ sensor time, location livingRoom,username;
	device Lights;
	while(username =I is in LivingRoom)
	if(time>=10pm)
	{ action = switch on Lights[Brightness,Dim] in LivingRoom;
	}
	}

When I Walk into the kitchen and turn on the stove ,turn on the television with the volume low.	ContextBegin C2
	{ sensor location , useractivity, username;
	device TV;
	if(username isI) AND(location is Kitchen) AND(Activity is Cooking)
	{ Action = switch on TV[volume,low];
	}
	}ContextEnd

If any one is sleeping at noon turn on his/her alarm clock	ContextBegin C3
	{ sensor time,username,useractivity ;
	device AlarmClock;
	While((time ==noon)AND (username is I)AND(useractivity is Sleeping))
	{ switch on AlarmCock[Alarm,ON];
	}
	}ContextEnd

If I m Sleeping turn the stereo off.	ContextBegin C4
	{sensor username,useractiviy;
	device stereo;
	While((username is I)AND(useractivity is Sleeping))
	{ action = stereo[status, off];
	}
	}Context End

If humidity is higher than 80percent and temperature is higher than28 degrees , turn on the air conditioner with 25 degrees of temperature setting	ContextBegin C5
	{ sensor humidity, attemp ;
	device AC ;
	if((humidity>80%)AND(attempt>28))
	action = switch on Ac[temp,25];
	} ContextEnd

User Interface: The compiler for the proposed system is implemented using Java Software Development Kit, Eclipse and Linux Platform (Fedora version 14). The Compiler has been implemented making use of ***propositional logic and fuzzy logic algorithms in addition to the lex and yacc program [10]***. Fig3 shows the program editor interface wherein the user can write the context program using the CPL. Like any other editor our context program editor contains all the basic functions of creating, opening, adding users and their profiles , closing , saving the context file and Edit menu which contains cut, copy, paste functions. Here the editor is provided with Build function which is used to debug and check for errors in a context script and a Run function to run the application and generate the corresponding actions as a output of context script. In case of incorrect code errors are reported to the user.

Fig4 shows a snapshot of the compiler throwing context error whenever the programmer attempts to declare **abc** as a device. Since **abc** is not a keyword or the name of any device the compiler fails to recognize the string.

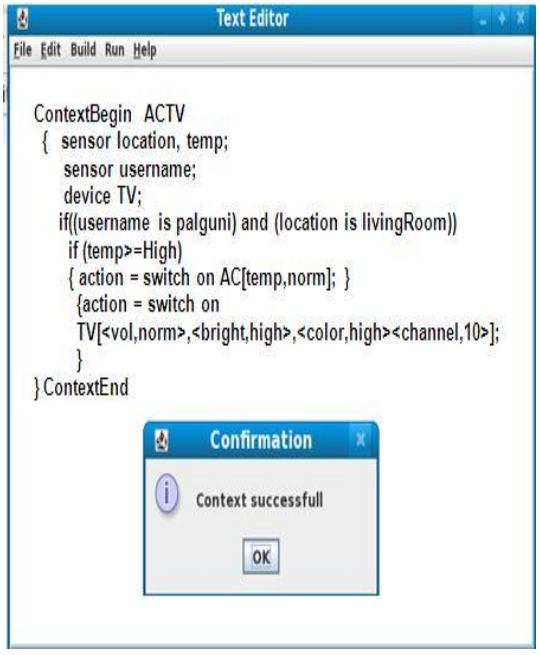

Fig3: Context Program Editor

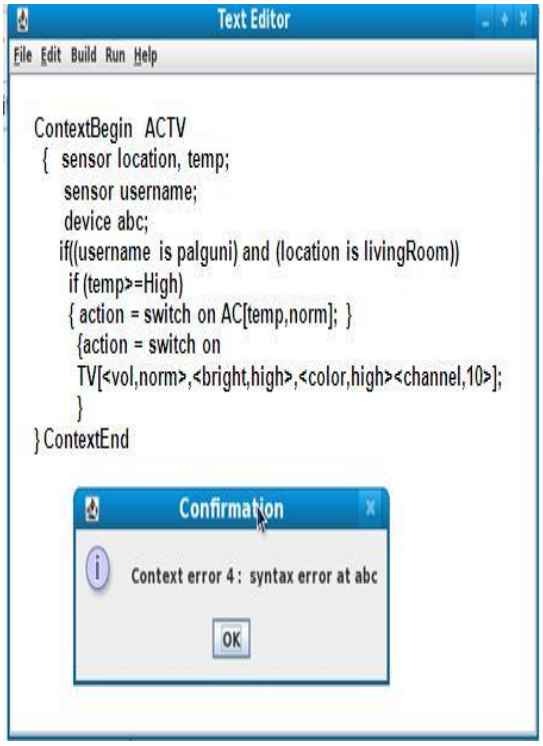

Fig 4: Context error: abc is not a device

VI. CONCLUSION

The Compiler designed successfully implemented many context aware applications which were based on simple if then and while rules. Whenever the complexity of context aware applications increases complexity of context programming also increases. As a future work the compiler and grammar has to be refined further to execute more complex context aware applications which encounters conflict among device, sensor and users , more libraries has to be included to make programming easier.

REFERENCES

1. Jaewoo Chang and Ahreum Kim ," A New Context Script Language and Its Processor for Developing Context- Aware Applications in Ubiquitous Computing" , 2008 11th IEEE International Conference on Computational Science and Engineering, DOI 10.1109/CSE.2008.49
2. Yongki Kim and Jaewoo Chang , "Design of a Context Script Language for Developing Context-Aware Applications in Ubiquitous Intelligent Environment ",2008 4th International IEEE Conference "Intelligent Systems" , 978-1-4244-1739-1/08/$25.00 © 2008 IEEE.
3. Devdatta Kulkarni and Anand Tripathi ," A Framework for Programming Robust Context-Aware Applications." , IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36 ,2010 , Digital Object Identifier no. 10.1109/TSE.2010.11
4. S. S. Yau, and F. Karim, "Context-sensitive Middleware for Realtime Software in Ubiquitous Computing Environments," Proc. of 4th IEEE Symposium on Object-oriented Real-time Distributed Computing, pp.163-170, 2001.
5. P. Couderc, A. M. Kermarrec, "Improving Level of Service for Mobile Users Using Context-Awareness," Proc. of 18th IEEE Symposium on Reliable Distributed Systems, pp. 24-33, 1999.
6. CONTEXT-ORIENTED PROGRAMMING by M.L.Gassanenko SPIIRAN, St.Petersburg, Russia mlg@forth.org, mlg@iias.spb.su.
7. CRL: A Context-aware Request Language for Mobile Computing by Alvin T.S. Chan, Peter Y.H. Wong, Siu-Nam Chuang Department of Computing, The Hong Kong Polytechnic University Hung Hom, Kowloon, Hong Kong.

8. COPAL-ML: A Macro Language for Rapid Development of Context-Aware Applications in Wireless Sensor Networks by Sanjin Sehic, Fei Li, and Schahram Dustdar Distributed Systems Group Information Systems Institute Vienna University of Technology.
9. Dey.A, "Providing Architectural Support for Building Context Aware Applications", Ph.D Thesis Dissertation, College of Computing, Georgia Tech, December 2000.
10. lex & yacc by John Levine,Tony Mason and Doug Brown.

AUTHORS PROFILE

Thyagaraju.GS received the M.Tech Degree in Computer Science And Technology From University Of Mysore ,India in 2002.He has got ten years of experience in academics ,fours years of Research Experience . He is a member of IETE.He has guided many students at UG and PG level.He is pursuing Ph.D in Computer Science Engineering. His Research Interests are Context Aware Computing in Ubiquitous and Intelligent Systems He is now working as a Senior Lecturer in Dept Of CSE, SDM College Of Engineering, Karnataka, Dharwad.He can be reached at thyagaraju_gs@yahoo.co.in.

Dr. Umakant Kulkarni obtained his BE Degree from Karnataka University, Dharwad in the year 1989, ME Degree from PSG College of Technology, Coimbatore in the year 1991 and PhD from Shivaji University, Kolhapur in the year 2007. He has published many papers at International Journal and IEEE conferences in the areas of Pervasive and Ubiquitous Computing, Distributed Data Mining, Agents Technology and Autonomic Computing. He is Member of IETE and ISTE. He served as Head of Department and Chief Nodal Officer- TEQIP a World Bank funded project. He has guided many students at PG level and five research scholars are pursuing their PhDs. Currently he is serving as professor in the Department of Computer Science & Engineering, SDM College of Engineering & Technology, Dharwad, Karnataka State, India. He can be reached at upkulkarni@yahoo.com