Using Weak Bisimulation for Enterprise Integration Architecture Formal Verification – I

Dr. Ernest Cachia, Mr. Mark Vella
University of Malta, Msida, Malta
ernest.cachia@um.edu.mt, mvel002@um.edu.mt

Abstract

Weak Bisimulation is a process calculus equivalence relation, applied for the verification of communicating concurrent systems [26]. In this paper we propose the application of Weak Bisimulation for Enterprise Application Integration verification. Formal verification is carried out by taking the system specification and design models of an integrated system and converting them into value passing CCS (Calculus of Communicating Systems) processes. If a Weak Bisimulation relation is found between the two models, then it could be concluded that the EI Architecture is a valid one.

The formal verification of an EI Architecture would give value to an EI project framework, allowing the challenge of cumbersome and complex testing typically faced by EI projects [18], to be alleviated, and thus increasing the possibility of a successful EI project, delivered on time and within the stipulated budgeted costs.

This paper shows the applicability of value passing CCS (or equivalent) formal notation to model the EI systems characteristics, as well as investigates into the computation complexity of available weak bisimulation algorithms, in order to analyze the applicability of this proposition in real life.

1. Background

In the process of searching for an Enterprise Integration (EI) specific framework to guide an integration team in the strategic implementation of an integrated IT landscape within and beyond the scope of a single enterprise, an extensive research in the following areas of Enterprise Integration was made: -

- Integration and middleware technology [7] [20] – for acquiring the understanding of the technological mechanisms that make systems integration possible.
- Standards [3][20] [28] – to be aware of the agreed upon standards in order to work on their lines.
- Scientific foundation [26] [29] [28] – in order to be able to add value to the field of Enterprise Integration based on the concepts of Computer Science and Software Engineering.
- Challenges [9] [10] [18] [20] [21] [22] [33] – in order to locate those Enterprise Integration specific areas that need improvement.
- Methodology and Best Practices (Linthicum 2003) [30] [33], [32] – in order to have the knowledge of existing improvement efforts and possibly build on them.

The industry research, made up mainly of compiled industry reports, articles [4] [16] and literature [5] [7] [10] [20] [30], allowed the broad understanding of the current state of EI projects in industry; from the projects business drivers, technologies and methodologies being used, to the factors affecting the success of these projects. On the scientific level, the research investigated which areas of computer science and software engineering could be applied to EI projects, in order to improve the situation of this area. [26] [29] [28]

1.1. Software engineering principles

A sound software engineering framework is one that delivers high quality software deliverables on budget and on time. [11] [29] In the case of an EI-specific framework, it is being proposed that in addition this would be a framework that is targeted specifically at EI systems, that achieves the EI-specific goals and qualities, allowing the EI project challenges to be overcome, and thus maximizing the probability of success and avoiding project failures as identified in [21].

1.2. EI-specific framework value

The proposed value of an EI-specific Project Framework could be better explained in the following scenario: take a software project manager who has managed traditional software projects for some time, but is now faced with the challenge of setting up an EI project plan. He/she should be aware of the fact that managing an EI project, although still a software project, requires a specific management framework to address the specific EI challenges. An EI-specific management framework would be very beneficial, in this particular case, to start building the EI project plan and carrying out all the necessary tasks leading to an effectively built EI system.

1.3. EI project challenges, goals and qualities

Further to what was presented in [9], according to [21] and [12] the main challenges causing failure in EI Projects include:

- Lack of standard methodologies – so far only industry best practices and EI product specific methodologies were found.
- Lack of proper business process definitions – in fact many business models today exist only in the head of departmental managers and at times these also conflict with the understanding of their colleagues.
- Lack of business units co-operation – In several cases, business units only communicate to put the blame on each other, and compete fiercely for company budgets. On the other hand, EI projects require full business unit co-operation.
- Implementation is more complex than expected – An EI implementation usually consists of several implementation technologies, packages from different vendors, multiple platforms and an unexpected number of interface links.
- Relying too much on integration technology for project success – Middleware technology in fact is only an enabler of integration implementation but far from being a complete software engineering tool.
- Lack of thorough testing - this is so, given the newly introduced integration level and the enterprise wide scope of such systems.
- Lacking the proper integration team roles – EI projects, given their novel nature, are only seen by the IT department as just another IT project, and fail to re-organize the IT roles before the start of these projects.

[30] compiled a list of goals and qualities that should be reached/exhibited by an EI-specific project framework. These are as follows: -

(Goals)
- Ensure that the EI architecture and developed applications satisfy business needs
- Describe how to manage the EI process
- Describe how to work with legacy systems and packaged solutions to integrate them
- Provide guidance on technology selection and standardization
- Ensure that the methodology promotes reuse

(Qualities)
- Align IT with the enterprise business strategy
- Build on a solid enterprise architecture
- Leverage legacy and commercial software
- Focus on security

In addition to these goals and qualities, the main strategic business value of EI today is that of being an enabler of Business Process Management [23] [24] [19]. This point was taken into consideration and a decision was taken to focus on the Business Process Integration [10] type of integration, where the main focal points of integration are the business process and not the applications. Here, Application Integration is only a consequence of joining up the business processes, but not the main driver.

These goals and qualities along with the EI project challenges form the basis for the reasoning underlying an EI-specific framework.

1.4. Framework building blocks

The foundation of this framework is made up of building blocks from the fields of Computer Science, Software Engineering and Business Management. These building blocks are: -

1.4.1. Value passing CCS (Calculus of Communicating Systems) or equivalent Process Calculus – from the field of Computer Science that allows the mathematical modeling of communicating concurrent and mobile systems. [26] CCS allows the formal specification and reasoning of communicating concurrent systems. Its applicability to the EI domain is shown in section 3 of this paper.

1.4.2. Unified Modeling Language (UML) – a software engineering tool allowing the modeling of software specification and design. [28] UML was

chosen due to the wide adoption in the software engineering world and its OMG standard status.

1.4.3. Business Process Management (BPM) – a business management discipline born out from Business Re-engineering that advocates end to end business process modeling, automation and their continuous monitoring and optimization. [5] This building block was made part of the project in order to be able address the goal of EI systems to be an enable to Business Process Management programmes.

This paper concerns only the application of the first building block: value passing CCS. More specifically, Weak Bisimulation, that is a binary relation on CCS processes [26], is being proposed as a possible tool for the verification of EI architectures.

1.5. Scope

The whole process required for completing the formal verification of an EI architecture is illustrated in figure 1, where the business process model defines the system specification and the EI Architecture is the system design that is verified against the business process model.

This paper shows the applicability of process calculus such as value passing CCS to the domain of EI, present a treatment of Weak Bisimulation and the possibilities this offers as a formal verification tool, and place the verification step within the context of an EI-specific framework. The formal specification and design, as well as an in-depth look at the verification process will be treated in subsequent papers.

2. Formal EI verification proposition

Weak Bisimulation (≈) is a process binary equivalence relation based on the equivalence of just the observable reactions between 2 processes. In other words, as long as the second process can match each observable reaction sequence of the first process and vice versa, the 2 processes are regarded as weak bisimilar, irrespective of their internal reactions. Thus Weak Bisimulation is also known as observation equivalence, with the processes in question regarded as black boxes

[26] shows how the weak equivalence relation (≈) could be used to prove that a particular system structure implements correctly a particular system definition. More specifically he showed the application of Weak Bisimulation as follows: ***System*** ≈ ***Specification.***

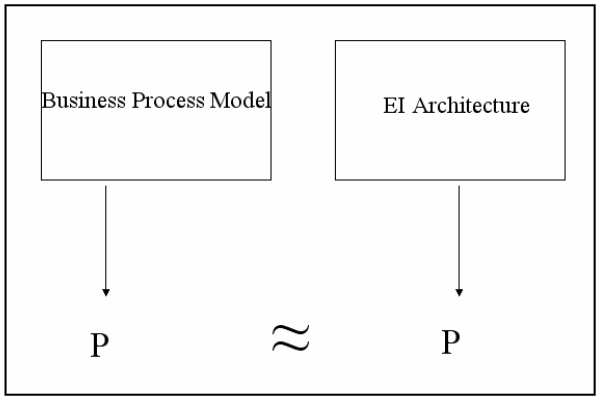

Figure 1 – EI Architecture verification process

Weak bisimulation is the chosen process equivalence relation for the fact that a system specification does not have as yet an internal structure and as a consequence, minimal internal reactions. Thus, this reasoning rules out both Structural Congruence (≡) and Strong Bisimulation (~). Given that Weak Equivalence (≈) is insensitive to both internal reactions and structure [26], it fits well the need of proving the correctness of the implementation of a system against its specification.

Applied to the domain of Enterprise Integration, or more specifically to the chosen Business Process Integration type of integration, this observation equivalence relation is being proposed to be applied as follows: -

Business Process Model ≈ EI Architecture

The Business Process Model defines the required business process flows to be automated by the underlying system, whilst the EI Architecture is the design of the EI system implementing the business flows. By mapping these models into value passing CCS processes, the equivalence between the Business Process Model and the EI architecture, could be formally verified by finding a Weak Bisimulation relation between these two processes.

3. EI systems characteristics

From the initial research underpinning this paper, the following architectural characteristics of EI systems stood out: -

- Concurrent Systems [7] [20]

- Business Process Modelling [GB and Ruh 04] [McGov01]
- Mobility [Smit and Fin 03]
- Security [Ruh 00]

The next four sub-sections introduce these characteristics and show how value passing CSS fairs, in modeling these characteristics.

3.1. Concurrent Systems

In EI architectures, concurrency is exhibited by the several applications, services and middleware executing in parallel, whilst messaging is the communication link between them. In Enterprise Integration, messaging is carried out by several middleware technologies that link applications together. [20] categorizes the middleware technology available today as follows: -

- Remote Procedure Calls – this type of middleware allows a software process to make synchronous calls to remote processes. E.g. Java Remote Method Invocation (RMI) [17]
- Message Oriented Middleware – this type of middleware is a queuing software that allows software processes to write and read messages to and from a queue. Communication between processes is asynchronous with guaranteed delivery. E.g. MQSeries. [14]
- Distributed Object Transactions – this is a middleware infrastructure allowing the exposure of business logic making up applications, supported by a transactional platform. E.g. Component Object Model (COM) [25] and Common Object Request Broker (CORBA) [6]
- Database Oriented Middleware – this kind of middleware provides software processes with access to database servers. E.g. Open Database Connectivity (ODBC) [15]
- Transaction Oriented Middleware – this type of middleware provides co-ordination of information movement and method sharing within the scope of a transaction. These are mainly to link legacy procedural applications to the transactional enterprise level. E.g. Tuxedo [2]

As defined by [26], value passing CCS – a formal way of modeling concurrent communicating systems, where variables are allowed along communication channels - is able to model concurrency and messaging, in the following ways: -

3.1.1. Concurrency. Being an extension of the CCS (Calculus of Communicating Systems) [26] value passing CCS supports the modeling of concurrent processes with the construct $P ::= (P1 \mid P2)$, where processes *P1* and *P2* are parallel composed together.

3.1.2. Messaging. The following value passing CCS constructs support the modeling of messaging as represented in Business Process Models (Figure 2) and EI Architectures (Figure 3). Figure 4 shows how these are represented in value passing CCS.

Input channel - $x(y).P$, means input a name on channel x by substituting with place holder y, and use the input in process P. In the EI scenario, x can represent a listening port such as a web service. The name y represents the place holder for an incoming message. Continuing on the example of a web service implementation, this incoming message can be an input XML (eXtendible Markup Language) document to the web service (a tagged data document), whose schema is referenced in the web service WSDL (Web Service Definition Language), which defined the interface of the particular web service.

Output Channel - $\bar{x}\langle y\rangle.P$ means output the name y on the channel named x, and then do P. In previous the web service analogy, this represents the consumption of the web service, where an XML document y is sent from the client application along channel x. In this case x is the TCP/IP based connection using the SOAP protocol.

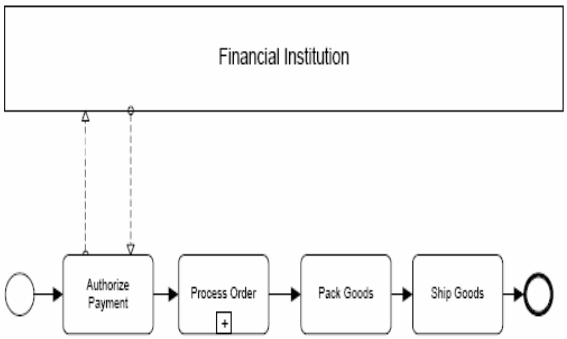

Figure 2 – Business Process Modeling Notation – Sending a message

In EI, messaging can be either **synchronous** for example in the case of a web service call or a Remote Procedure Call (RPC), or **asynchronous**, as in the case of message queues. It is possible to model both these types of messaging using value-passing CCS as follows.

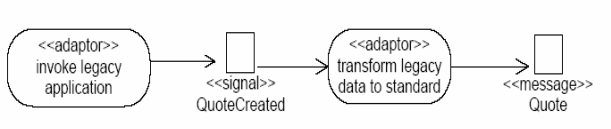

Figure 3 – Messaging in UML Activity Profile for EAI (Enterprise Application Integration) [28]

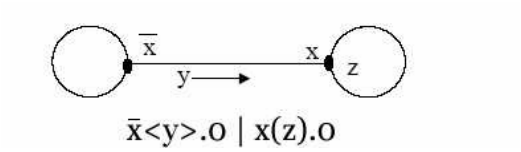

Figure 4 - Messaging in value passing CCS

3.1.3. Synchronous Messaging

$P = new\ x\ (\ P1 \mid P2\ )$

$P1 = \bar{x}\langle y\rangle.x(z).P1'$

$P2 = x(a).\tau.\ \bar{x}\langle b\rangle.P2$

Using the web service analogy in the above simplified process; an application process *P1* consumes web service *P2*. *P1* calls *P2* along channel *x* and passes the XML document *y* as an input. When the call is made, *P1 -> P1''* transition occurs, where *P1'' = (z)x.P1'*. In the *P1''* state, the calling process is kept blocking waiting on the same channel for web service *P2* to return. Once a message is returned back along channel *x* the calling process goes into state *P'* executing the rest of the process.

From the server's perspective *P2* listens indefinitely on channel *x*. When an XML document arrives along channel *x*, the web service executes its internal logic, represented by the tau (τ) symbol, and returns an output document XML back on channel *x* to *P1* on completion. The web service then resumes in state *P2* listening indefinitely for the next call.

3.1.4. Asynchronous Messaging

$P = new\ x\ y\ (\ P1 \mid P2 \mid P3\ )$

$P1 = \bar{x}\langle y\rangle.P1$

$P2 = x(w).\tau.\bar{y}\langle w\rangle.P2$

$P3 = y(z).\tau.P3$

Process *P2* handles continuous asynchronous communication between processes *P1* and *P3*. Even though the communication between *P2* and the other processes is synchronous in terms of readiness of processes to be able to communicate between each other, process *P2* provides the mechanism for asynchronous messaging between *P1* and *P3*. In the above-simplified example, *P2* models a messaging queue-like structure, allowing application *P1* to asynchronously call *P3* without blocking, even in cases when *P3* is not ready to communicate.

3.2. Business Process Modeling

Business Process Modeling involves the modeling and documentation of the business process flows within an enterprise and is a main element of Business Process Management [5] [3] [Smit & Fin 03]. EI systems are expected to serve as an infrastructure to these modeled processes, possibly by means of a Business Process Management System. [23]

CCS abstracts the notion of a process and is not specific to any particular software process living in a computer memory. Thus, it can be argued that CCS could also be suitable for modeling processes in the business sense. In the business context, we have business processes, embodying workflows, running in parallel communicating between each other inter-departmental and business to business (B2B) messages. In [34], Smith and Fingar elaborate in full detail of how Pi Calculus, an extension of CCS incorporating mobility, perfectly suites the modelling requirements of business processes.

As a matter fact, Process Calculi are already being applied to BPM for other reasons. The Business Process Modelling Notation (BPMN) adopted by the Business Process Management Initiative (BPMI) [3] is fully based on Pi Calculus foundations; as is the Business Process Execution Language For Web Services (BPEL4WS) by Microsoft Corporation [25] and IBM [13] and the BPML (Business Process Modelling Language) by the BPMI.

3.3. Mobility

From personal experience in enterprise integration projects, the communication links between the nodes of an EI architecture (applications, services and middleware) are not of fixed nature but rather of a dynamic mobile nature, where communication channels are created, moved and destroyed. For example we have the scenarios where communication channels between two processes are created as in the discovery of a web service by UDDI (Universal

Discovery Description and Integration) protocol [35]. There are also situations of channel proliferation where for example an application server or middleware becomes unavailable. There are also situations of pure mobility where a communication channel is relocated in the process space as in the scenario where a client application is instructed to start communication with an alternate server for example, for performance reasons or during a seamless, no down time, new application roll out.

Mobility in the context of business processes is reflected in the continuous change in business rules as a result of a Business Process Management Programme [Smith and Fingar, 2003]. An example is where in a move to eliminate bureaucracy an electronic components manufacturing company consolidates approvals of component designs in one department. In this case a link between the engineering department and the first auditing department in line, moves to a link with the newly created consolidated department

Pi Calculus supports the modeling of process mobility by definition; in fact Pi Calculus was invented by extending the CCS to support mobility. Mobility is modeled in Pi Calculus by allowing names representing communicating channels to be themselves passed as messages along channels. [26]

Even though Pi Calculus would be required to model mobility, for the time being only value passing CCS is being considered in order to make the proposition of this verification tool clearer. Moreover, the researched Business Process Modeling Notation [3] does not include the concept of mobility within the graphical notation; thus making value-passing CCS sufficient for formally representing business models built using this notation for the time being.

3.4. Security

Now that the applications have been 'opened up' in order to communicate possibly with the whole world, the issues of security breaches increase [Ruh, et al, 2000]. An improper security infrastructure can invalidate an otherwise well built integrated architecture. In fact the role of secure messaging increases immensely in such architectures. The importance of security in EI projects has already been identified in the Secure Application Integration Methodology by [30].

The notion of restriction in CSS denoted by *(new x) P* means that *x* is for the exclusive use of *P*. A more specific example is *P = new x (P1 | P2)* which means that *P1* and *P2* communicate via a private channel x even when placed in the context of other concurrent processes having a channel with the same name [26]. This construct enables us to model secure communication channels in integrated architectures.

Having said that, this does not mean that by using the restriction construct, it means that model is definitely secure, it only specifies that the indicated channel of communication should be secured. A typical case where a secure channel does not imply complete overall security is shown in the following system: -

$System = new\ x\ (P1 \mid P2) \mid P3$

$PI = \bar{x}\langle outval \rangle$

$P2 = x(inval).\bar{y}\langle inval \rangle$

$P3 = y(inval)$

Where even if *P1* and *P2* communicate via a secure channel, over which *P3* can never interfere, it is still up to the implementation of P1 and *P2* to ensure that that any sensitive data is not passed on to *P3*. In the above example, *P2* is sending the data received over the secure channel *x* to *P3*.

Still, having the individual processes formally defined, these can be individually formally verified in terms of system security.

4. Enterprise architecture verification

It is being proposed that CCS based verification occurs within an EI framework as follows.

Once the Business Process Model diagram (the system specification), and the EI architecture (system design) are produced, these are converted to CCS processes. This way the model passes from a semi-formal representation of the system to a formal one. If a Weak Bisimulation relation were found between the two processes, the EI architecture would be considered equivalent in behavior to the Business Process Model, and thus validated.

This verification step is carried out during the framework stage where the EI Architecture is completed and the lower level steps of design and implementation are about to start (Figure 5). Having the architecture validated at such an early stage maximises the success of the EI system by alleviating the challenge of EI testing as discussed in [18] and [21].

5. Practical applicability

In order for the proposed EI Architecture formal verification tool to be practical and usable in real world applications, the verification process must be automated, and the algorithm automating the process should do this in an efficient manner.

[1] explain that Weak Bisimulation can be decided in a time complexity of $O(n^{2.3})$, where n is the number of states, using a technique called the partitioning/splitter technique. Thus, using this algorithm, it is possible to efficiently decide a Weak Bisimulation between two processes given that: -

1. We keep the number of states finite
2. Possibly minimizing the number of states

The first condition can only be adhered to by not using variables from infinite domains (the algorithm does not work on symbolic labeled transition systems). This might sound too restrictive at first, but if it is kept in mind that at this stage the system is simply being described rather than specified in terms computations, it can be viewed from a different aspect. For example, consider a task that receives an integer, and decides on two alternative flow branches based on the value of the integer received. In the system specification and design models, one simply needs to pass a boolean value based on the size of the integer. In this case, instead of using a variable from an infinite domain, one can use a variable from a finite one. The latter can be achieved by assuming synchronous parallelism as described in [1], and thus avoiding a state size explosion when a system is made up of several parallel composed processes.

An alternate approach would be that of using a Weak Bisimulation algorithm that functions on symbolic transition systems, allowing the use of variables from an infinite domain. [8] present an efficient Bisimulation algorithm stating that is possible to port to the symbolic case.

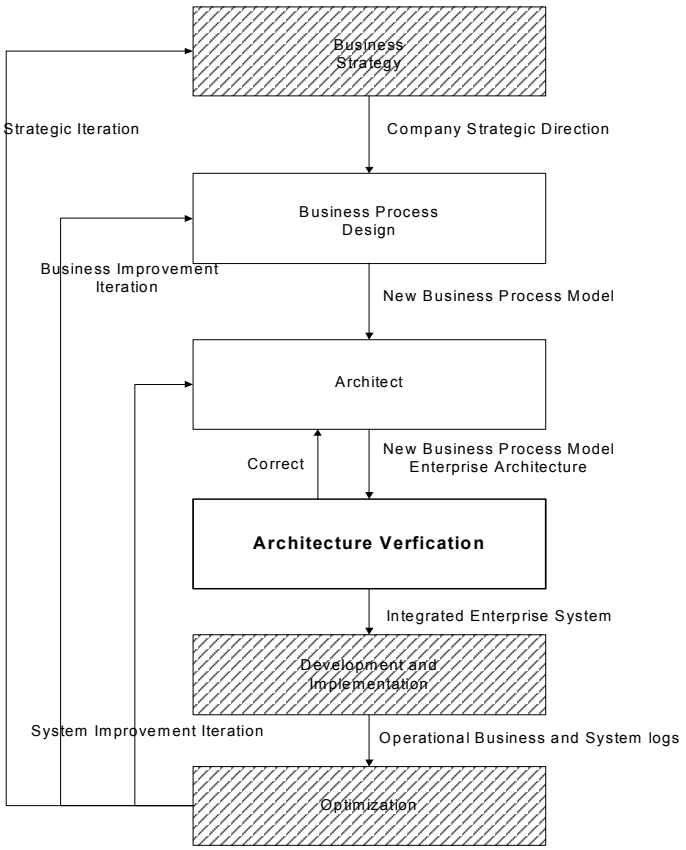

Figure 5 – Architecture Verification as part of an EI framework

6. Conclusions

This paper attempts to show that the application of value passing CCS and Weak Bisimulation as a means of validating an EI Architecture against the Business Process Model is possible by the treatment of the mapping of Business Process Models and EI Architectures to value passing CCS processes. CCS is able to model the main features of EI systems; these being Concurrency, Business Process Modeling, and Security. On the other hand the Weak Bisimulation relation is an equivalence relation that is insensitive to internal structure and reaction. This enables the checking for an equivalence relation between the Business Process Model and the EI Architecture. Whilst Pi Calculus would be required to model the remaining EI characteristic, mobility, this is not considered for the time being since mainly since no Business Process Modeling Notation researched so far addresses mobility, and also for reasons of better focusing on the usage of the proposed formal verification system in real life.

The formal verification of the EI Architecture is proposed to give value to an EI-specific project framework, allowing the challenge of complex testing typically faced by EI projects, to be overcome by allowing the formal verification of the architecture, before development and testing starts.

Finally the paper also proposed a sound foundation for a way forward for a practical application of the proposition, by mechanically automating the verification process.

7. References

[1]Baier, C. and Hermanns, G. (1999) Weak Bisimulation for Fully Probabilistic Processes

[2]BEA Tuxedo. *http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/tux* http://www.bea.com

[3]BPMI.ORG The Business Process Management Initiative Homepage http*://www.bpmi.org*

[4]Business Integration Journal Online. *http://www.bijonline.com*

[5]Burlton, R. (2001). Business Process Management: Profiting From Process. SAMS

[6]CORBA.ORG. The OMG's CORBA Website. *http://www.corba.org*

[7]Cummins, F.A. (2002). Enterprise Integration: An Architecture for Enterprise Application and Systems Integration. Wiley

[8]Dovier A., Piazza, C. Policriti A. An efficient algorithm for computing bisimulation equivalence.
[11][Erk & Pen 98] Eriksson, H.E. and Penker, M. (1998). UML Toolkit, Wiley Computer Publishing.

[9] Garimella, K. Ph.D. (2001). Integration Challenges in Mergers and Acquisitions. In *EAI Journal Aug 01.*

[10]Gold-Bernstein, B. and Ruh, W. (2004). Enterprise Integration: The essential guide to integration solutions, Addison-Wesley.

[11]Ghezzi, C. et al. (2002). Fundamentals of Software Engineering (2md edition). Prentice Hall.

[12]Herrera, J. (2004). Avoiding Common EAI Implementation Missteps, LogicCurve.

[13]IBM.COM IBM Homepage http*://www*.ibm.com

[14]IBM WebSphere MQ. http*://www*.ibm.com

[15]IODBC.ORG. Platform Independent ODBC. *http://www.iodbc.org*

[16]Integration Consortium *http://www.wwintegration.com*

[17] Java Remote Method Invocation (Java RMI) java.sun.com/products/jdk/rmi.

[18]Khanna, R. (2005). Top Challenges in Integration Projects. Wipro Technologies White Paper.

[19]Krishnan, M. (2004). The EAI Paradigm Shift, WIPRO Technologies White Paper.

[20]Linthicum, D. S. (2003). Next Generation Application Integration: From Simple Information to Web Services. Addison Wesley.

[21]Lublinsky, B. and Farrel M. Jr. (2002). Top 10 Reasons Why EAI Fails. In *EAI Journal.Dec' 02*

[22]Maverick, G. (2003). EAI Project Management. In *EAI Journal Nov' 03.*

[23]McGoveran, D. (2001). BPMS Concepts, Enterprise Integrity. In *EAI Journal Jan '01*

[24]McGoveran, F. (2003). Managing Business Process for EAI, In *Business Integration Journal Sep '03.*

[25]Micosoft.com Microsoft Corporation Homepage *http://www*.microsoft.com

[26] Milner, R. (1999) Communicating and mobile systems: the Pi-calculus. Cambridge
University Press.

[28] Object Management Group (2004) *http://www.omg.org*

[29]Pressman, R.S. (1996). Software Engineering: A Practioner's Approach. 4th Edition. McGraw-Hill.

[30]Ruh, A. W., et al; (2000). Enterprise Application Integration: A Wiley Tech Brief. Wiley

[32]Schmidt, J. (2003). EAI Lifestyle Evaluation. The Software Ecologist Column, In *EAI Journal Apr' 03*.

[33]Sifter, C.J. (2001). Integration Project Management 101. In *EAI Journal March' 01.*

[34]Smith, H. and Fingar P. (2003). Business Process Management: The Third Wave. Meghan-Kiffer Press.

[35]UDDI.ORG The Universal Description, Discovery and Integration (UDDI) protocol homepage *http://www.uddi.org*