*I.J. Information Technology and Computer Science,* **2016, 01, 54-66**
Published Online January 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2016.01.07

# Some Observations on Dependency Analysis of SOA Based Systems

**Pawan Kumar**
Department of Computer Science, MMV Banaras Hindu University Varanasi-221005
E-mail: pawan.bhuphd@gmail.com

**Ratneshwer**
Department of Computer Science, MMV Banaras Hindu University Varanasi-221005
E-mail: ratnesh@bhu.ac.in

***Abstract*—**This paper presents some observations on Dependency Analysis of Service Oriented Architecture (SOA) based systems. In general, dependency analysis is based on the internal properties of artifacts (objects/components/services) and inter-relationship between the artifacts in the system. In order to make a dependency analysis for SOA based systems, one have to consider the special features of SOA that make them different with other approaches. This paper surveys the previous works taken on dependency analysis of service oriented systems. The present work provides insights about definitions related to service dependency, the modeling and analysis techniques of service dependency analysis, failure results to service dependence and some research challenges of the topic.The contribution of this paper is for novice researchers working on this topic as they can get an overview of dependency analysis of SOA based systems for their further research.

***Index Terms*—**Service oriented architecture, Software services, Dependency analysis, Data Dependency, Control Dependency.

## I. Introduction

Dependency analysis is widely addressed in modular design of various software systems, ever since the early days of software engineering. However a systematic and disciplined way to review dependency analysis works for Service Oriented Architecture (SOA) based systems has not yet been observed. This paper presents literature review of Dependency Analysis of SOA based systems. Different research questions related to dependency analysis in SOA based systems are identified. The present work provides insights about definitions related to service dependency, the modeling and analysis techniques of service dependency analysis techniques, failure results to service dependence and some research challenges of the topic. Since this is a comparatively new topic in software engineering, so we have taken an open search strategy. In this study, standard journals related to software engineering and service oriented architecture have been selected. Appropriate key-words have been chosen. A devastating situation may occur if some dependencies among services are not identified carefully. Some SOA based systems may be real time systems. In real time system, it is inevitable to identify every dependency because any deficiency leads to catastrophic situation [1].

Our purpose of doing this review is to get an overview of dependency analysis of SOA based systems. SOA presents newer aspects of dependency analysis due to its different architectural style and programming paradigm. There is similarity between service oriented development and component based development but we strictly focus on the dependency analysis works related to software services only and do not include the dependency analysis works for component based software or object oriented software. In this way this review work will naturally strengthen the coherency between the SOA and dependency analysis and traces a better picture of major issues, challenges and possible solutions of dependency analysis of SOA based systems.

The present work is organized as follows. In sectionII, we briefly mention some existing works related to review/survey of dependency analysis. In Section III, we briefly describe the concepts of SOA and dependency analysis. In section IV, we mentioned the identified research questions. In section V and its following subsections, we made an attempt to answer the identified research questions. Finally we conclude the topic on section VI.

## II. Related Work

In literature, the problem of dependency has been addressed widely but few works has been observed regarding systematic review of dependency analysis. Parnas (1979) [2] pointed out the problems of having uncontrolled dependencies between software modules and introduced the concept of information hiding. One significant work found in the literature is performed by Arias et al [3]. In their workextensive survey of dependency analysis has been performed that gives relevant information about dependency analysis. An industrial survey of requirements interdependencies in software product release planning has been given in [4]. This paper describes the complexity of interdependency analysis in relation to metrics of requirements coupling.

 

A survey of Data Dependency Analysis Techniques for Automated Parallelization is given in [5]. This paper discusses the dependencies that exist between statements in that program and detailing several different classes of dependence analysis techniques. Bhuyan, Prakash and Mohpatra[6] have performed a survey of regression testing of SOA based systems. This paper gives valuable information pertinent to testing in SOA, but it has not discussed about dependency analysis. Motlagh has done a survey of testing of SOA based system. Motlagh has described about testing challenges of SOA based systems [7]. Lewis, Smith and Kontogiannis[8], in their report, outlined the SOA Research Agenda. It also provides detail on specific research challenges related to the maintenance and evolution of service-oriented systems. Trigos[9] has analyzed different approaches for dependency analysis amongst services in a business process.

The above contributions demonstrate that although some approaches of survey and review works are available in the literature but a systematic and disciplined review work of dependency analysis especially in context of SOA is yet not found. The proposed work extends the above contributions further by presenting an extensive review on dependency analysis of SOA based systems.

## III. SOA and Dependency Analysis

The one major problem that we have encountered during this work is the divisive definition of SOA and the confusion associated with the service oriented architecture and service oriented implementation. We made a general understanding of SOA based on following definition.

"SOA is a design philosophy independent of any vendor, product, and technology or industry trend. SOA may be realized via web services but web services are not necessary required to implement SOA. With an SOA the application's functionality is exposed through a collection of services. These services are independent and encapsulate both the business logic and its associated data. The services are interconnected via messages with a schema defining their format; a contract defining their interchanges and a policy defining how they should be exchanged [10]".

"Service-Oriented Architecture (SOA) is a software architecture where functionality is grouped around business processes and packaged as inter operable services. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services [11].

In essence, it is a way of designing a software system to provide services to either end user applications or other services through published and discoverable interfaces [12]. Loose Coupling between services is one of its design principles that help system or architecture to maintain its efficiency because services are less dependent on each other [13]. Service coupling shows how much a service has dependency to other services. Since business process choreography is performed by calling services according to business process control flow, it is possible that input of a service is obtained from output of other services, and these results in coupling of two services. A request to a service is implemented through a message which is sent to service operations [14]. Understanding dependencies in a SOA based system is essential to perform two functions: impact analysis (understanding which other components are affected when a component become unavailable or malfunctions) and software component level root cause analysis (understanding the cause of a component by looking at the other components it relies on) [15]. In an SOA based systems, software services depend on other software services by service providing/ receiving relationships.

A service-oriented system consists of a set of services and includes various types of dependencies among them such as business processes, semantic, messages, and non-functional requirements and the underlying information models shared by these service dependencies. Discovery of dependencies among services of large distributed systems is crucial for management software. Higher dependency leads a complex system vulnerable, which results in poor understanding and a higher maintenance cost in SOA. Dependency is a relationship involving two or more services where a change of state in one or more service(s) leads to a potential for a change of state in one or more other services. Dependency analysis involves the identification of interdependent services of a system.

The knowledge of a service's dependencies is important for a number of management activities and development of a SOA based system. Fault management needs this information to track problems in a distributed service network. Configuration management needs this information to know which services are currently in use and appropriately adapt to changes in the environment. Accounting management needs to know dependencies to appropriately charge for service access. Policy-based management needs to know dependencies and must be able to change them to enforce the policies. All these management activities must have ways to learn the current dependencies, discover their properties, and possibly perform rebinding of services [16].

## IV. Research Questions

Research questions have pivotal role in literature review. Dependency analysis in any system is inevitable for checking the quality of the system. Dependency analysis helps in testing, maintaining, identifying the error of the system. It prevents chaos of the system failure. Since SOA is an active research area and it can solve a number of real problems, to address dependency issues in SOA based development becomes inevitable to avoid undesirable situation.

Some basic questions related to dependency analysis of SOA based systems have been taken. The research questions addressed by this study are as follows:

 

***RQ1. What is the uniqueness of dependencies of SOA based systems?***

Motivation: How the dependence problem among services is different from components and objects?

***RQ2. What are the proposed definitions of dependency of SOA based system?***

Motivation: The first step is to understand about dependency related to SOA.

***RQ3. What are the available 'dependency analysis and modeling' techniques for SOA based System?***

Motivation: Obtain an overview of existing solutions so that one can build newer solution according to existing solutions. Using overviews one can highlight pros and cons of existing solutions.

***RQ4. What are the impacts of fault caused by service dependence?***

Motivation: Obtain the impact of failure of SOA based system due to service dependence.

***RQ5. What are the 'research challenges' observed for dependency analysis of SOA based system?***

Motivation: To identify hindrance to do the work for dependency analysis of SOA. By reviewing the existing research, it becomes easy to find research challenges.

***RQ6. What are the available tools for dependency analysis in SOA based system?***

Motivation: By this question, it has been tried to identify the existing tools and their pros and cons.

Since 'Dependency Analysis in SOA based System' is relatively new topic, so we choose the open search strategy. SOA is a new paradigm and most of the significant works related to dependencies are observed after 2006. Because of this reason, journals and proceeding have been selected since 2006. In random searching, search has been performed at random using keyword related 'dependency in Service Oriented Architecture', 'dependency management in SOA', 'dependency among software services' etc. We have used Google, Live and Babylon search engine for the purpose. Searching has been done manually. We have gone through those papers which are related to dependency analysis of service oriented architecture. We have included some survey papers also. General software engineering papers are excluded. Papers based on object oriented analysis/component based software are also excluded. For understanding purpose some papers related to only SOA and general dependency are included. The selected study acts as a primary studies for literature review. Inclusion and exclusion criteria are done on the basis of studying abstract and introduction of papers from selected journals and proceeding conferences.

## V. Observations

In this section, possible answers of aforementioned questions have been discussed.

***RQ1. What is the uniqueness of dependencies of SOA based systems?***

Although objects, components and services have common concepts of reusability in their origin but they have differences at their architectural, internal description and abstraction levels. An Object is known by its member data and member functions; components are known by their classes and interfaces whereas services are known by their service contracts. Objects have tight coupling but components and services have loose coupling. The major differences are in their connections and the way they provide services to third party. Service oriented computing provides a way to create a new architecture that reflects components trend towards autonomy and heterogeneity [17]. Software components support black box and white box encapsulation both but software services support only black box encapsulation. In CBSE, there is a limited composition support for components of different models but software services are implementing in diverse technologies, on different platforms. In general, dependency analysis is based on the internal properties of artifacts (object/components/services) and inter-relationship between the artifacts within the system. In order to make a dependency analysis for SOA based systems, one have to consider the special features of SOA that make them different with other approaches. The dependency between the service provider and the service client is a run-time dependency, not a compile-time dependency. The service consumer does not know the format of the request message or response message or the location of the service until it needs a particular service [18].

There is a need to make separate dependency analysis approach for SOA based systems due to following reasons.

- A service should be stateless to ensure that it is not dependent on the context or state of other services. Any dependencies between services should be defined in terms of common business process, function and data models [19].
- Services have to be largely independent from implementation specific attributes.
- Services are invoked through defined communication protocols that stress interoperability and location transparency [20].
- The global collaboration pattern between constituent services is located inside a single entity; the composition schema which expresses the overall behavior in terms of work flow and data flow. In component based systems the communications are located inside the connectors which split the global

 

behavior and the work flow is not explicit [21].

- A service defines in terms of data contracts, operation contracts and service contracts. Dependency analysis of SOA based systems should be based on these contracts whereas dependency analysis of component based systems is based on their interfaces.
- A service has taken concept of ownership to the extreme and thus, the providers of the services are responsible for the development, quality of service, maintenance, deployment and execution. On the contrary, components split the responsibilities at the deployment level [21].
- If we work with components, we are predominantly working with the code within the language boundary. If we work with services, we use some remote functionality over network under some contract [20].
- Services operate in distributed environment and focus on document centric communication. In contrast, component based development does not take that much stand on how the components interact with one another –this depends on the technology that the components are based on [22].

The discussions above clearly suggest that there are fundamental operational differences between software services and software components and there is a need for a fresh look at dependency analysis for SOA based systems.

***RQ2. What are the proposed definitions of Dependency of SOA based system?***

In the present literature, the dependency relations among software services are mentioned in circumstantial manner and vary widely in concepts. Although various dependency definitions are available in the literature but here we considered only those that are especially given in context of SOA based systems.

Winkler et al. [19] define service dependency as "A service dependency is a directed relation between services. It is expressed as a 1: n relationship where one service (dependant) depends on one or multiple services (antecedent). A service S1 is dependent on a service S2 if the provisioning of service S1 is conditional to the provisioning of service S2, i.e. if a property of service S1 is affected by a property of S2."

Another definition of service dependency is given in [23] as 'The concept of Web Services consists in the dynamic advertisement, discovery and access of business functionality among multiple cooperating partners. Consequently, failures occurring in one service affect other services being offered to a customer, i.e., services have dependencies on other services.'

The service dependency is assumed as a relationship of services offering functionality to other services. Service dependency is beyond traditional poor service description, and directed by various sources such as data, resource, procedure control, utilizing techniques etc [24].

The above definitions describe dependencies in SOA based system as service providing relation between software services. A relationship among software services can only be treating as dependency relationship if the making change in one service affects the functionality/behavior of other services related to this. The above definitions reflect this concept. But these definitions only concern specific types of dependencies and do not take into account the context of the system or the organization. Also these definitions are concerns to application level dependency only and do not take the network level aspect into consideration. These definitions take a general concept and do not distinguish regarding kind of dependency that is dependency in the source code, during execution time or due to hardware resources. Therefore the current definitions are not across-the-board to accommodate different types of dependencies. If all services come from same service provider then dependency analysis is easy, but if services are obtained by different service providers (and also from different locations) then dependency analysis becomes complex.

***Types of Dependency Relations***

One of the possible classifications of service dependency relations are with respect to their occurrences that is either between the two individual services or among services in a composite service. Winkler et al [19] have named dependencies that occur between the individual services of a service composition as horizontal dependencies because they affect services on the same hierarchical level of composition. Dependencies can also have direct effects on the overall composition. These dependencies, which occur across a hierarchical level of composition, are named as vertical dependencies.

One classification of dependency is proposed to distinguish between inter-service, service-resource, and inter-resource dependencies. These denote dependencies between services, services and resources as well as between resources, respectively [24].

Caswell and Ramanathan [25] describe dependencies for services. They define five kinds of dependencies: Execution dependency (performance of an application server process depends on the status of the host), Link dependency (performance of a service depends on the link status), Component dependency (in case of a web service that is provided on different front-end servers which are selected by a round-robin DNS scheduling the performance depends on the currently selected server), Inter-service dependency (this type of dependency occurs between services, e.g. e-mail service depends on an authentication service and on an NFS service), Organizational dependency (services and/or server may be mapped to different domains of responsibility).

Service dependency can also be categorized on the basis the reason of the cause that creates dependency. Some such dependency relations, observed from literature, are summarized here in the table below (table 1).

It can be observed that there is no unanimous concern regarding the types of dependency. Dependency analysis provides a visual representation of the services in SOA based systems and helps one to monitor and understand

how various services are providing/receiving services to each other. One can get idea about what services are deployed, how dependency relation affects in case of adding new services/deleting services. In SOA based systems, dependency relationships can be observed at architecture level, design level and at execution time. The level of detailed available for software services is limited, and as a result, dependency relationships of software services may be obtained at architecture level. At architecture level, dependencies can be identified based on syntactic and semantic information available in a formal specification of software architecture and the connections among services and the constraints on their interactions. In SOA, the analysis of dependencies at architecture level is important for understanding the coordination and orchestration of services. At design level different modules are constructed according to the basis of core SOA principle. Dependency analysis at design time provides information about which service depend on other services before they execute. If proper dependency analysis, among software services, is performed then the design level weaknesses can be easily identified and then design modifications can be made early in the design cycle and reduce the cost of development. Analysis of service dependencies at execution time is crucial in order to understand the effect of services in case of failure. When a service has a failure or performance degradation, all other services that depend directly or indirectly on this service might be impacted. Dynamic dependencies involve with handling unpredictable changes of runtime environment that cannot be handled using static composition techniques [27].

Table 1. Observed Dependency Relationships among software services

| Types of Dependencies | Short Description | References |
|---|---|---|
| Design Time Dependencies | Design time dependency refers to the dependency of a service as it exists after deployment but before execution. | [26] |
| Cyclic Dependency | Suppose there are three services s1, s2 and s3. If s1 is dependent on s2, s2 on s3 and s3 on s1, then it is cyclic dependency. | [27] |
| Input/output Dependency | It occurs when a service requires/provides data from /to another service. | [27] |
| Cause and Effect Dependency | It occurs when a service has preconditions to be satisfied based on the effect of the other services. | [27] |
| Transitive Dependency | Assume there are three services s1, s2, and s3. If s1 is dependent on s2, s2 on s3 then s1 is dependent on s3. | [27] |
| Price Dependency | A price dependency exists when the price of a service depends on the price of other services. | [19] |
| Location Dependencies | Location dependencies occur between two services that need to be executed at the same or at a different location. | [19] |
| Resource Dependency | Two services have a resource dependency, when the availability of a resource, which is needed by one service, depends on another service. | [19] |

***RQ3. What are the available 'dependency analysis and modeling' techniques for SOA based System?***

If we have the knowledge of internal code and operations of a system then operational dependencies can easily be computed. However, if the system is complex and source code and implementation details of the system are unknown (like web services, components etc), a different approach is required for dependency analysis. Success of a SOA based system is not only depends on the properties of individual services but also on how these services interact/coordinate with each other. The knowledge of this interaction/coordination is essential at the early stage of life cycle to avoid confusion and unpredictable situation. Knowledge about dependencies between services in SOA based system is not explicitly available but is rather implicitly contained in service interfaces, service level agreements and service descriptions. Depending on the specific application and the function of the dependency information, the approaches to analyzing and modeling dependencies vary greatly. A careful reading of literature helps in identifying research efforts that have made into classification of concepts and understandings regarding dependency analysis of SOA based systems. Here we have two possible factors for classification. One classification is possible based on the source of information used by the different approaches and the other one is the type of outcome the different approaches have. One way of classification is that whether we are measuring the service dependency at network level or application program level. For our discussion, we have taken the factor 'type of outcome'. The type of outcome used among existing dependency analysis approaches can be classified in following three groups: graph based approach, formal modeling/algorithmic approach and artificial intelligence and other approaches.

***(A) Graph Based Approach***

In a SOA based system, it would be easy to understand the dependencies among services by their textual description for a simple system having fewer services. But as the number of involved services, in a SOA based system, increases it becomes difficult to analyze and track dependencies among services based on their textual descriptions only. To represent the dependencies among services, in an effective manner, a graphical representation is an efficient and easy to understand approach. Graph Based Approach is very powerful tool for dependency analysis for SOA based system. Significant works have been observed in Graph based

service dependency analysis. The table below (Table 2) summarizes the Graph based approaches for Service dependency analysis.

Table 2. Summary of Graph based Dependency Modeling Approaches

| *Approach* | *Source of Dependency information* | *Strengths/ Weaknesses* | *Type of output* | *Architecture/ design/ execution Time* | *Reference* |
|---|---|---|---|---|---|
| Discover dynamic dependencies among services | Correlations among message exchanges between services, logging mechanism provided by SOA Manager | Poor message log data may affect the measurement. | A probabilistic dependency graph in which edges are labeled with a confidence level | Execution time | [15] |
| Combine approach of semantic matching of inputs and outputs interfaces and process cases for analyzing dependencies | Service interface, service contracts | Measure the strength of dependencies | frequency table, dependency Graph, algorithm to automatically detect the conflicts by using the structure of dependency Graph. | Design time | [17] |
| Active-perturbation approach to infer dynamic dependencies | Initial knowledge of the implementation detail of system | Compute dependency strength, identifying cross-domain dependencies | Active dependency discovery (ADD) Graph | Execution time | [32] |
| Extraction of dependencies among services | local repository with abstract semantic description of web services | extract cyclic dependency, Input/output dependency, Cause and Effect dependency | A dependency Graph | Design/ execution time | [27] |
| Dependency Markup Language to capture dependencies amongst activities | Service interface, service contracts | Resulting specification is more abstract than a concrete control flow | Service Flow/Control Flow Graph, | Design/ execution time | [30] |
| Applying XML, XPath and RDF to the problem of describing, querying and computing the dependencies among services | A web–based architecture for retrieving and handling dependency information | Fault management applications, | Dependency Graph | Design/ execution time | [22] |
| Track the dependencies of services and represent them in graphical models. | Design time dependency of deployed SOA artifacts in an OC4J container. | Layout Flexibility, Single Console for End-to-End Visibility | sCrawler: SOA Dependency Tracker tool | Design/ execution time | [26] |
| Dependency Impact Analysis Model | Change specification of software services | Capture the changing entities | Graph-based service dependency matrix | Design time | [16] |
| Extract Dynamic Dependencies by Vector Clocks | Use the vector clocks to infer dependencies among services. | Implementation into the Apache CXF4 framework | Dynamic dependency graphs of web services | Design Time | [29] |
| Discovering Service Dependencies in Mobile Ad Hoc Networks | Monitoring agents collect dependence data by intercepting the message traffic between services | Evaluate the performance of the method through a series of extensive simulation-based experiments | Dependence Graph | Execution Time | [31] |

Basuet. al. [15] have presented a module that automatically analyzes service execution data to discover dynamic dependencies among services. Construction of a probabilistic dependency graph is concatenation of all identified dependencies between pairs of messages by taking into account the assignment of services to nodes. Edges are labeled with a confidence level, which is the probability of the identified dependency. They experimentally analyzed his approach on HP internal data about the execution of business processes invoking various services within HP. Yan et al [17] have given an approach that combines the semantic matching of inputs and outputs interfaces between service operations and the analysis of process cases to identify service dependencies. The main contributions of this approach are that it can be used to identify the direction of the service dependencies and non-conflict property and non-redundancy property of discovered service dependencies are guaranteed based on a dependency graph. Romano, et al. [29] has given the idea of using explicit system perturbation to elucidate dependencies that they denoted as active dependency discovery (ADD). They have used Active-perturbation approach in which they explicitly inject problems into the system, monitor service behavior, and infer dynamic

dependencies. The ADD procedure builds an operational dependency graph for a particular combination of system and workload while requiring very few details of the internal implementation of the system. Omer and Schill[27] have investigated a method of automatic composition plan creation that relies on automatic extraction of dependencies among services. Extracted I/O dependencies are represented using a directed graph. This approach utilizes existing graph traversal based algorithms to extract cyclic dependency and generate the execution plan. Tolksdorf [30] has proposed a Dependency Markup Language to capture dependencies amongst activities and generalizations/specializations amongst processes. He described composite services at more suited levels of abstraction and has several options to use such descriptions for service coordination, service discovery and service classification. Ensel and Keller [22] have described a novel approach for applying XML, XPath and RDF to the problem of describing, querying and computing the dependencies among services in a distributed computing system. Its output is a consolidated dependency graph that can then be used by fault management applications to perform additional problem determination tasks or event correlation. Phukan [26] has discussed some of the problems inherent in the SOA service life cycle, and shows how graph based automated dependency tracking can help to analyze and alleviate these problems. Wang andCapretz [16] have proposed a service dependency graph model and service dependency and relations matrices to analyze the dependency of Web services. They performed ripple-effect analysis by calculating service dependency, cohesion, and impact effect within and among services. They also proposed the change specification to the web services, so that the service developer can apply the changes directly to the web service definition. Romano, Pinzger and Bouwers [29] have given an approach to build up dynamic dependency graphs of web services. These graphs are commonly weighted, where the weights indicate the number of times a particular service is invoked or a particular execution path is traversed. Novonty et al. [31] proposes a visualization and dependency analysis framework for a Web application. Based on the deep analysis for the text feature of hyperlink, a regular expression-based linkage information extraction method is presented.

The goal of the above approaches is to represent the SOA based system in the form of graph and identify relationships among service

### *(B) Formal Modeling/Algorithmic Approach*

The importance of formal modeling of syntactic and semantic characteristics of software systems has long been accredited. It is practicable to use mathematical foundations to capture the essential behavior of service oriented software systems for the purpose of dependency analysis. Some major contributions of this approach are summarized in the table given below (Table 3).

Yanchuk, Ivanyukovich and Marchese, in their work [33], have proposed mathematical definitions for individual service, service-oriented environment and service-oriented application. Winkler et al.[19]have proposed an approach for analyzing dependencies between services in a composition in a semi-automatic manner at design time and capturing them in a dependency model. Later they applied this model for validating negotiated Service Level Agreements (SLA) and determining the effects of events such as service failure or SLA renegotiation on other services. Zheng, Zhou and Krause [34] have demonstrated how to model BPEL data dependencies with proposed web service automata (WSA). They focus on analyzing BPEL data dependencies. The proposed WSA is implemented in XML. Sell et al [20] have compared two approaches for modeling dependencies as a base for managing adaptations of complex business processes. Based on two use cases from the domain of workflow management and service engineering they illustrated the need for capturing dependencies and derive the requirements for dependency modeling. For dependency modeling we discuss two alternative solutions. One is based on OWL-DL ontology and the other is based on a meta-model approach. Kuang et al. [21] have explained the importance of importing dependency between interfaces into service matchmaking; propose a service specification that describes dependency between interfaces in a concise and easily-extended way and proposed a novel service matchmaking algorithm considering different characteristics. Bodenstaff et al [35], in their paper, have demonstrated how to analyze SLAs during development phase and how to monitor these dependencies using event logs during runtime. They named their approach MoDe4SLA (Monitoring Dependencies for SLAs). With the MoDe4SLA approach they analyze during development phase different types of dependencies between services, and the impact services have on each other.

Keller and Kar [36] have introduced the concept of dependency lifetime that traces the flow of dependency information from the design to installation to runtime stages of a service. A dependency may be directly available from system information repositories such as ODM (AIX) or RPM (Linux) in machine-readable format (high degree of formalization); or a dependency mayexist only in the notebook of a system administrator (very low degree of formalization). This dimension is important because it serves as a metric that helps to evaluate how expensive and/or difficult it is to acquire, identify, represent and track this dependency during the lifetime of the component. Alda [28] have given a novel way for handling service dependencies in peer-to-peer architectures is proposed. His approach is based on presumes two assumptions. The first assumption implies that any consumer peer can subscribe to a list maintained by a provider peer if the peer relies on a public service offered by that provider. If an adaptation is planned, the operator of a provider peer is able to consult all subscribed peers before the adaptation can be carried out. The second assumption states that peers of a peer group have agreed to a common adaptation policy in the run-up to the adaptation of a service. Liu, Ma and Zhao [37]

have proposed an approach to identifying conversation dependency between business processes to facilitate the dynamic evolution of SOA based system. In their approach, a business process is represented as a directed graph, and the matrix method is used to identify the execution order of activities in the business process, which determines the conversation dependency.

### *(C) Artificial Intelligence and other Approach*

In recent years artificial intelligence has gained popularity in every field of modern technology. Seminal works for dependency analysis in SOA based system using artificial intelligence technique are being done. Neural network, fuzzy logic, genetic algorithm based dependency solutions can be categorized under artificial intelligence technique. Some major contributions of this approach are summarized in the table given below.

Ensel has presented a new methodology to automatically generate service dependency models. The approach specially aims for heterogeneous environments. It is based on two key parts. The first are the underlying concepts of dependency determination that are carried out with the help of neural networks. The second part deals with questions of installation efforts and scalability to seamlessly integrate the modeling into real IT-environments [38].Ai and Tang have given a repair genetic algorithm, namely minimal-conflict hill-climbing repair genetic algorithm, to address the Web service composition optimization problem in the presence of domain constraints and inter service dependencies and conflicts [39].

As it is clear from the above observations, that most of the dependency analysis and modeling techniques are centered on Graph based approach or algorithmic approach. The most common source of information is service interface/service contracts.

Table 3. Summary of Algorithm based Dependency Modeling Approaches

| Approach | Source of Dependency information | Strengths/ Weaknesses | Type of output | Architecture/ design/ execution time | Reference |
|---|---|---|---|---|---|
| Dependency analysis between services in a composition | Process structure and SLA information. | Used for SLA validation/renegotiation, Semi-automatic solution | A set of algorithms for finding dependency types | Design time/execution time | [19] |
| Web service automata approach | Business Process Execution Language (BPEL) description of software system | Considered internal and external data dependencies | An Eclipse based tool | Design Time | [34] |
| Dependency Modeling with OWL-DL ontology and a meta-model approach | Knowledge about terms of a domain (e.g., activity or service names) and their interrelations. | Requires specific validation support for validation of dependency model | A meta-model which allows the creation of Dependency Models. | Design time | [22] |
| Dependency between interfaces into service matchmaking | Service interface, service contracts | Consider the internal behavior of service interfaces | A service match-making algorithm | Design /execution time | [21] |
| Monitoring Dependencies for SLAs | SLAs during development phase and using event logs during runtime | Identifying causes for SLA violations, calculate the impact a service has on a depending composite service | monitoring results with analyzed dependencies and impacts of the composite service | Development time | [35] |
| Concept of dependency lifetime that traces the flow of dependency information | System information repositories such as ODM, configuration and installation files, etc. | A classification to identify the various aspects of dependencies | Dependency Architecture for Application Service Management | Design/ execution time | [33] |
| Dependency management in service-Oriented peer-to-Peer architectures | A list of depending peer services that is maintained by each service providing peer. | Rating dependencies to analysis of dependencies, Visualization Tool for Dependency Analysis | Adaption policies to clarify how a service provider peer should handle existing dependencies. | Design/ execution time | [28] |
| Identifying conversation dependency between services | A business process is represented as a directed graph | Identify the conversation dependency to facilitate the dynamic evolution in SOAs | An algorithm that uses matrix method to identify the execution order of activities in the services | Design Time | [37] |

Table 4. Summary of Artificial Intelligence based Dependency Modeling Approaches

| Approach | Source of Dependency information | Strengths/Weaknesses | Type of output | Architectur/design/execution time | Reference |
|---|---|---|---|---|---|
| Propose a service specification that gain management relevant dependency information | Priori description of the environment with its services and dependencies, | Specially aims for heterogeneous environments, based on neural networks | Constructed and trained neural networks with data, an agent based architecture enabling the model creation | Execution Time | [38] |
| Inter-Service Dependencies Using Minimal-Conflict Hill-Climbing Repair Genetic Algorithm | abstract specification of workflow | inter-service dependencies and conflicts caused by domain constraints, technological constraints and context-based constraints | A repair genetic algorithm (MCHC-repair GA) | Design Time | [39] |

***RQ4. What are the impacts of fault models caused by service dependence?***

Detection of dependencies between a failing service and the services that depend on the failing service is important because if a service fails or has changed its service agreement such as its SLA, it can affect other services in the composition [40]. If dependencies exist among services, then developer must be aware about these dependencies so that appropriate actions should be taken in adverse conditions. When a service has a failure or performance degradation, all other services that depend directly or indirectly on this service might be impacted. It is important to understand what the service dependencies are, so that management tools can display and alert users about the business impact of failures and performance degradations. Furthermore, knowledge of dependencies considerably simplifies service-level root-cause analysis that is, trying to understand the origin of a failure [16]. The knowledge of dependencies between services may further be useful for the prediction of impacts on other services due to management operations [41]. Service related fault reports are more ambiguous. They relate to how the quality of the service is perceived by the users and not limited to a complete failure of a service, but also indicate that a service is provided with a low quality [42]. When a customer experiences problems with a service, the provider needs to react quickly in order to honor the Service Level Agreements (SLA) in effect between provider and customer. Conversely, it is desirable to determine the impact onto services and Service Level Agreements when problems with resources or subservices are detected [40].

Dynamic binding of services enables their self-adaptivity and self-management. Nevertheless, all these options do not only bear prospects of improved infrastructure - they can also be sources of serious failures [43]. IT has long been established that many software faults are caused by violated dependencies that are not recognized by developers during designing and implementing a software system [44].

Some of the impacts of faults observed in literature, caused by service dependence, are mentioned in Table 5.

Table 5. Summary of Fault Impacts due to Service Dependence

| Possible Fault (due to service dependence) | Impacts of the faults | Reference |
|---|---|---|
| Transitive nature of service dependency | The failure of one service may not only affect the direct consumer but all other services which handle the same resource. | [19] |
| Inaccuracies in Dependency specification | Wrong dependency identification among services. | [15] |
| Faulty software documentation | Software documentation is the important source for dependency identification. Fault documentation may result in wrong dependency identification. | [36] |
| Lack of coordination in service adaption | The uncoordinated adaptation of public services potentially leads to malfunctions in the environment of depending peer services in SOA based systems. | [28] |
| Change in service operations | A direct impact occurs when the service elements within a service are affected due to their dependency on the changing service element, while an indirect impact may happen when the service elements of dependent services are affected. | [16] |
| Resource failures | Resource failures could endanger the SLAs by affecting the provided services. | [40] |
| SLAviolations | If a service fails or has changed its service guarantees such as its SLA, it can affect other services in the composition. | [9] |

***RQ5. What are the 'research challenges' observed for dependency analysis of SOA based system?***

Researchers and practitioners have been considering various aspects of dependency analysis of SOA based systems and related issues. It is required to consider these efforts for the purpose of identification of research challenges and problems in the field so as to be able to

ascertain some important considerations that need urgent, and possibly immediate attention. This research question attempts to present, in a concise manner, the research challenges related to the topic of discussion.

Some of the research challenges, observed from literature, are mentioned below.

- The automated service dependency is a challenging problem in the administration of large distributed system. The difference in behavior, together with the structure of the composition, makes it difficult to manage the dependencies between services [35].
- Automatically analyzes service execution data to discover dynamic dependency among services is another challenge. The problem is far from trivial as it requires understanding correlations among message exchanges between services [15].
- The discovery of dependencies between Web service executions in distributed environments is a challenging problem in general. Due to the wealth of possible different software and hardware infrastructures and the considerable number of different protocol specifications that can be used in a specific implementation; the dependency discovery problem comes in a variety of different flavors [15].
- It is often undesirable and sometime impossible to store a complete, instantiated dependency model at a single place [45].
- It is practically unfeasible to obtain the dependency information and keep the information up-to-date manually. So it is necessary to detect accurate and up-to-date operation dependencies in an automatic manner [17].
- The dependency analysis problem becomes very challenging in situations where the resources in the system are dynamic in nature. In such cases, resources can appear and disappear during system lifetime because of failures, or deployment of new sub-systems, and the dependency relations can change as a result of change of resource availability or new service level agreements being negotiated [46].
- Measuring indirect dependencies among services.
- Obtaining dependence information in such a system is made difficult by the inherent loose coupling of services, as many dependencies are unknown at design time, and only established at run time through a dynamic service binding mechanism [31].

***RQ6. What are the available tools for dependency analysis in SOA based system?***

In this section a concise summary of dependency analysis tools, in context of SOA based systems, are given. The purpose is to get an overview of various tools used for Dependency analysis and modeling. The idea is to get to know that whether existing tools are sufficient for the purpose or it is require modifying the tools or proposing a new one in context of newer requirements of SOA based systems.

Some tools are mentioned in the section, but there is no implication that these are the only meant for SOA based applications.

These tools are listed in Table 6.

Table 6. Dependency Analysis Tools

| Dependency Analysis Tools | References |
|---|---|
| sCrawler | [26] |
| JDepend | [47] |
| DepAn | [48] |
| ADaM | [49] |
| SOA Dependency Analyser | [50] |
| eDepend | [51] |
| HP OpenView SOA Manager | [15] |
| Serviz | [52] |

The dependencies among services can be explored in tools such as HP Open View SOA Manager, and the performance metrics of all the dependent services are captured. Any change in dependencies of a SOA business application on the underlying services can then be captured in a database and updated in SOA Manager [15].

SCrawler is a dependency tracking utility that tracks dependencies and presents information about them in an application-agnostic manner. It maps the design time dependency of deployed SOA artefacts in an OC4J container. It makes analysis intuitive and less time-consuming, and presents information in a well-structured manner. SCrawler extracts all the process dependencies from the Oracle Application Server and presents the same as graphs in a 2D graphical console. It gets information on all the deployed processes and creates a selection tree. For each leaf in the process tree, it recursively builds the dependencies. The end user need not know anything about SOA or Application Server to find the dependencies [26].

SOA Dependency Analyzer is a tool for graphical visualization of the dependencies between the processes (BPEL-WS) and services (Service Bus). This tool was developed for easy and simple understanding of the dependencies between services and processes, sometimes in very complex environments SOA. It is built on Eclipse RCP (SWT) framework and to visualize the dependency graph used Eclipse GEF/ZEST framework [50].

Serviz is a tool that visualizes how services are activated, and how much they interact over time. It encompasses both a data collection component and the visualization component. Data collection relies on the Turmeric SOA platform, while the visualization is web-based and makes use of open source JavaScript visualization and graphing libraries. Serviz is an open-source tool and its user-interface allows system maintainers to visualize and inspect the runtime data collected from web-service based systems. By using Serviz, maintainers are presented with a topology of a running SOA system and are therefore able to analyze the system's usage over user-defined periods of time [52].

 

Spassov et al. [53] have implemented a Data Dependency Analysis Tool (DDAT) to support the testing, validation and verification of both functional and non-functional behavior of service-based applications at design time as well as at runtime. The Data Dependency Analysis Tool (DDAT) solves two tasks. Firstly, for a given set of BPEL activities, the tool finds a path that goes through all activities in the set starting from one initial activity. Secondly, it finds all control activities on the discovered path and calculates the condition that should be met in order for the process to continue execution along the path.

Wang [54] has developed an impact analysis tool that allows change analysts to identify service dependencies, make changes, and to perform impact analysis. Wang has used this tool to evaluate the actual industry Web Services for service synchronization.

DepAn[48] is a direct manipulation tool for visualization, analysis, and refactoring of dependencies in large applications. DepAn deals with direct manipulation of heterogeneous dependency information in an Eclipse RCP environment, analysis and visualization of very large applications, collapse child dependency into parent entities to reveal class level interactions and import of file systems as source of dependency information.

EDepend [51] integrates a set of tools to effectively manage and control class & package dependencies. Well controlled code dependencies ensure easy maintenance and evolution of code is a graphical, interactive and real-time dependency analysis solution for Java projects in Eclipse. It integrates a rich set of tools to effectively detect, display, navigate and analyze class/package/project dependencies.

## VI. Conclusion

A software service dependence analysis process consists of multiple activities that need to be carried out for achieving the desired goals. These activities may involve human elements and possibility of automation also exists. Software services dependence analysis process must be improved through learning from experiences and the knowledge resulting from the experiences must be analyzed and made use of for the purpose of improvement of the concerned process. One basic idea is to assess the organization's current practice and improve its dependence analysis process on the basis of the competencies and experiences of the practitioners working in the organization. A major challenge is to create strategies and mechanisms for managing relevant and updated knowledge about software services dependence for the purpose of testing and configuration management. It is our conjecture that most of the cost of dependency analysis process can be reduced by using a knowledge base supported by a software tool which gathers and manages the experts' knowledge. To change software developers practices, the organization should improve the practitioners' existing knowledge (both theoretical and practical) of its software practices. In other words, knowledge about the new services should be made available on different organizational levels. This would require to judiciously considering the various activities in a SOA based system for this purpose. Such an effort would finally aim at redefinition of software services dependency analysis processes in context of SOA.

This work explores the literature review of various dependencies in the context of SOA based systems. We understand that the identified issues and challenges regarding the dependencies among services may help in future research in this area. This initial proposition of such a review may be purposefully used by the academician/researchers and the corresponding useful feedback may be analyzed. It calls for further extensive research oriented studies, by all concerned, for identification of newer issues and challenges.

## References

[1] P. Kumar, and Ratneshwer, "A Review on Dependency Analysis of SOA based System", 2014 Fifth International Conference on Recent Trends in Information, Telecommunication and Computing, (IEEE Explore), 21-22 March,2014, Chandigarh, pp. 69-81.

[2] D. L. Parnas, "Designing software for ease of extension and contraction", IEEE Transaction on Software Engineering, 1979, Vol. 5 pp. 128-138.

[3] T. B. C. Arias, P. Spek, and P. Avgeriou, "A practice-driven systematic review of dependency analysis solutions," Empirical Software Engineering, 2011 Vol. 16, pp. 544-586.

[4] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J N Dag, "An industrial survey of requirements interdependencies in software product release planning," In: Proceeding of 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada,2001, pp. 84-91.

[5] J C Libby, and K. B. Kent, "A survey of data dependence analysis techniques for automated parallelization," Technical Report, TR07-188, 2007, pp.1-34.

[6] P. Bhuyan, C P Kashyap, and D P Mohapatra, "A survey of regression testing in SOA. International Journal of Computer Applications," 2012, Vol. 44, pp.22-25.

[7] E S Motlagh, "A survey of service oriented architecture systems testing," International Journal of Software Engineering & Applications (IJSEA), 2012, Vol. 3(6) pp.19-27.

[8] G Lewis, D. Smith, K. Kontogiannis et al., " A research agenda for maintenance & evolution of SOA-based system ," In Proceedings of IEEE International Conference on Software Maintenance,2007, pp. 481-484.

[9] E D Trigos, "Master thesis service dependency analysis based on process models and service level agreements", Dissertation, Dresden University of Technology, 2007.

[10] D Linthicum, "Understanding service oriented architecture. MSDN, Microsoft". http://msdn.microsoft.com/en-us/library/bb833022.aspx. Accessed on 19 May 2013.

[11] R. Cover, "The xml cover pages", 2000, http://www.oasis-open.org/cover/xml.html. Accessed on 19 May 2013.

[12] A El Sharkawi, and A. Shouman, "Service Oriented Architecture for Remote Sensing Satellite Telemetry Data Implemented on Cloud Computing," International Journal of Information Technology and Computer Science

(IJITCS), Vol. 5, No. 7, June 2013, PP.12-26.
[13] J. S. M. Khalid, and M. Asif, "Grid Approach with Metadata of Messages in Service Oriented Architecture," International Journal of Information Technology and Computer Science (IJITCS), Volume 6, Number 2, January 2014, PP.64-71.
[14] A Kazemi, A. Rostampour, et al., "A genetic algorithm based approach to service identification ," In Proceedings of IEEE World Congress on Services, Washington, DC, USA, ., 2011, pp. 339-346.
[15] S. Basu, F. Casati, and F. Daniel, "Toward web service dependency discovery for SOA management," In: Proceedings of IEEE International Conference on Services Computing, doi: 10.1109/SCC.2008.45, 2008, Vol. 2 pp. 422-429.
[16] S. Wang, and M. A. C. Capretz, "A Dependency Impact Analysis Model for Web Services Evolution," In: Proceedings of the IEEE International Conference on In Web Services, 2009, doi: 10.1109/ICWS.2009.62 pp. 359-365.
[17] S. Yan, J. Wang, C Liu, and L Liu, "An approach to discover dependencies between service operations," Journal of Software, 2008, Vol. 3(9), pp.36-43.
[18] M. P. Papazoglou, and W. Heuvel, "Service Oriented Architectures: Approaches, technologies and research issues," The VLDB (The International Journal on Very Large Data Bases) Journal, Volume 16, Issue 3, July 2007, pp. 389-415.
[19] M. Winkler, T. Springer, E D Trigos, and A.Schill, "Analyzing dependencies in service compositions," In: Proceedings of the 2009 International conference on Service-oriented computing, 2009, Springer-Verlag Berlin, Heidelberg. pp. 123-133.
[20] C. Sell, M. Winkler, T. Springer, and A. Schill, "Two Dependency Modelling Approaches for Business Process Adaptation," Knowledge Science, Engineering and Management(KSEM) Lecture Notes in Computer Science Vol. 5914, Springer, 2009, pp. 418-429.
[21] L. Kuang, J. Wu, Y. Li, S. Deng and Z. Wu, "Exploring dependency between interfaces in service matchmaking," In: Proceeding of IEEE International Conference on Services Computing, doi: 10.1109/SCC.2007.60, 2007, pp. 506-513.
[22] C. Ensel, and A. Keller, "Managing application service dependencies with XML and the resource description framework," In: Proceedings of IEEE/IFIP International Symposium on Integrated Network Management, 2001, doi: 10.1109/INM.2001.918072, pp. 661-674.
[23] J. Zhou, et al., "Dependency-aware service oriented architecture and service composition," In: Proceedings of IEEE International Conference on Web Services, 2007, doi: 10.1109/ICWS.2007.71. pp. 1146-1149.
[24] A. Hanemann, M. Sailer, and D. Schmitz, "Towards a Framework for IT Service Fault Management," In Proceedings of the European University Information Systems Conference (EUNIS 2005), Manchester, England, June 2005. EUNIS.
[25] D. Caswell, and S. Ramanathan "Using service models for management of internet services," In HP Technical Report HPL-1999-43, HP Laboratories, Palo Alto, California, USA, March 1999.
[26] S. Phukan, "sCrawler: SOA dependency tracker," Available at http://www.oracle.com/technetwork/articles/scrawler-sandeep-phukan-085368.html, 2009, Accessed on 23rd May 2013.
[27] A. M. Omer, and A Schill, "Dependency Based Automatic Service Composition Using Directed Graph," In: Proceedings of Fifth International Conference on Next Generation Web Services Practices, 2009, Prague, doi: 10.1109/NWeSP.2009.20, pp. 76-81.
[28] S. Alda, "Peer group – based dependency management peer-to-peer architectures," Proceedings of the 2005 International Conference on Databases, information systems, and peer-to-peer computing, 2005, pp.195-202.
[29] D. Romano, M. Pinzger, and E. Bouwers, "Extracting dynamic dependencies between web services using vector clocks," In: Proceeding of IEEE International Conference on Service-Oriented Computing and Applications, 2011, pp. 1-8.
[30] R. Tolksdorf, "A Dependency Markup language for web services," In: Web, Web-Services, and Database Systems, 2003, Springer Berlin Heidelberg. pp. 129-140.
[31] P. Novotny, A. L. Wolf, B. J.Ko, and S Lee, "Discovering service dependencies in mobile ad hoc networks", Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013 pp. 527-533.
[32] A Brown, and D. Patterson, "An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Environment," In Seventh IFIP/IEEE International Symposium on Integrated Network Management, Seattle, WA, May 2001, pp.377-390.
[33] A. Yanchuk, A. Ivanyukovich, and M. Marchese, "Towards a mathematical foundation for service-oriented applications design," Journal of Software vol. 1, 2006, pp. 32-39.
[34] Y. Zheng , J. Zhou, and P. Krause, "Analysis of BPEL data dependencies," In: Proceedings of 33rd IEEE EUROMICRO Conference on Software Engineering and Advanced Applications, Washington, DC, USA, 2007, pp.351-358.
[35] L. Bodenstaff, A. Wombacher, R. Wieringa, M.C. Jaeger, and M. Reichert, "Monitoring service compositions in MoDe4SLA: design of validation," ICEIS 2009 - Proceedings of the 11th International Conference on Enterprise Information Systems, Volume SAIC, Milan, Italy, May 6-10, 2009, pp. 114-121.
[36] A. Keller, and G. Kar, "Dynamic dependencies in application service management," In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV,USA, 2000, pp. 1-7.
[37] M. Liu, D. Ma, and Y. Zhao, "An approach to identifying conversation dependency in service oriented system during dynamic evolution," In: Proceedings of the 2009 ACM symposium on Applied Computing, 2009, pp. 1072-1073.
[38] C. Ensel, "A Scalable Approach to Automated Service Dependency Modeling in Heterogeneous Environments," In Proceeding of 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC '01), Seattle, WA, USA, September 2001, pp.128-139.
[39] L. Ai, and M. Tang, "QoS-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm," In: Proceedings of IEEE Fourth International Conference on eScience, doi: 10.1109/eScience.2008.110, 2008, pp. 119-126.
[40] A. Hanemann, D. Schmitz, and M. Sailer, "A framework for failure impact analysis and recovery with respect to service level agreements," IEEE International Conference on Service Computing, July 11-15,Orlando, Florida, 2005

pp. 49-56.
[41] C.R.B. de Souza, "On the Relationship between Software Dependencies and Coordination: Field Studies and Tool Support," PhD dissertation, Donald Bren School of Information and Computer Sciences, Univ. of California, Irvine, 2005.
[42] A. Hanemann, and P. Marcu, "Algorithm Design and Application of Service Oriented Event Correlation," In Proceedings of the 3rd IFIP/IEEE International Workshop on Business Driven IT Management (BDIM 2008), Salvador Bahia, Brazil, April 200, pp. 61-70.
[43] S. Bruning, S. Weissleder, and M. Malek, "A Fault Taxonomy for Service-Oriented Architecture," Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium, 14-16 Nov. 2007, doi: 10.1109/HASE.2007.46 pp.367-368.
[44] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," IEEE Transactions on Software Engineering, Vol. 35(6), 2009 pp. 864-878.
[45] A. Keller, and G. Kar, "Determining service dependencies in distributed systems," IEEE International Conference on Communications, 2001. ICC 2001, vol.7, pp.2084-2088.
[46] S. Bagchi, G. Kar, and J. Hellerstein, "Dependency Analysis in Distributed Systems using Fault Injection: Application to Problem Determination in an e-commerce Environment," In Proceeding of 12th International Workshop on Distributed Systems: Operations & Management, 15 - 17 October 2001, Nancy, France, pp.151-164.
[47] M. Clark, "JDepend," available at http://clarkware.com/software/JDepend.html, accessed on 24 May 2013.
[48] L. Carver, "DepAn: a dependency analysis tool," available at http://google-opensource.blogspot.in/2008/09/depandependency-analysis-tool.html, accessed on 24th May 2013.
[49] "ADaM - Automated Dependency analysis Manager," availableat http://www.3di-ltd.co.uk/adam.html, Accessed on 24th May 2013.
[50] T. Frastia, "SOA Dependency Analyzer User Guide", Released June 5, 2011 downloaded 30 April 2014, pp.1-22.
[51] "EDepend - Graphical Dependency Analysis Tool 3.5.0," available at http://marketplace.eclipse.org/content/edepend-graphical-dependency-analysis-tool, accessed on 24 May 2013.
[52] T. Espinha, A. Zaidman, and H G Gross," Understanding the runtime topology of service-oriented systems," Proceedings of 19th Working Conference on Reverse Engineering Kingston 2012, Ontario, Canada, pp. 187-196.
[53] I. SpassovI, V. Pavlov, A. D. Petrova, and S. Ilieva, "DDAT: data dependency analysis tool for web service business processes," In: Proceedings of International Conference of Computational Science and Its Applications (ICCSA), Springer Berlin Heidelberg, 2011, pp. 232-243.
[54] S. Wang, "A dependency based impact analysis framework for service-oriented system evolution," PhD Dissertation, University of Western Ontario, Canada, 2010.

## Authors' Profiles

**Pawan Kumar** is working as a Senior Research Fellow at Department of Computer Science (MMV), Banaras Hindu University, Varanasi (India). He is currently working on 'Dependency Analysis of SOA based Systems'. He is pursuing his doctoral work under the supervision of Dr. Ratneshwer.

**Ratneshwer** did his Ph.D. in Component Based Software Engineering from Indian Institute of Technology, Banaras Hindu University, Varanasi (IIT-BHU), India. His research area is CBSE and SOA. He is serving as an Assistant Professor in Department of Computer Science (MMV), Banaras Hindu University, India. He is actively involved in teaching and research for last 8 years. One research monograph is published by LAP Germany and one book chapter has been published by IGI Global Publication. He has 16 research papers in International journals and 16 research papers in international/national conference proceedings in his credit.

**How to cite this paper:** Pawan Kumar, Ratneshwer,"Some Observations on Dependency Analysis of SOA Based Systems", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.1, pp.54-66, 2016. DOI: 10.5815/ijitcs.2016.01.07