A Review on Dependency Analysis of SOA based System

Pawan Kumar and Ratneshwer

Banaras Hindu University, Department of Computer Science, Varanasi, India
Email: pawan.bhuphd@gmail.com
Email: ratneshwer@gmail.com

Abstract—This paper presents a survey of Dependency Analysis of Service Oriented Architecture (SOA) based systems. SOA presents newer aspects of dependency analysis due to its different architectural style and programming paradigm. This paper surveys the previous work taken on dependency analysis of service oriented systems. This study shows the strengths and weaknesses of current approaches and tools available for dependency analysis task in context of SOA. The main motivation of this work is to summarize the recent approaches in this field of research, identify major issue and challenges in dependency analysis of SOA based systems and motivate further research on this topic.

Index Terms—Service oriented architecture, Software services, Dependency analysis

I. INTRODUCTION

Dependency analysis has had its importance recognized by software engineering people. However a systematic and disciplined way to review dependency analysis work for Service Oriented Architecture (SOA) based systems has not yet been observed. This paper presents literature review of Dependency Analysis work of SOA based systems. Different research questions related to dependency analysis in SOA are identified. During the study, we observed a few review papers related to SOA and Dependency Analysis. This review paper will help to understand current situation of research in dependency analysis of SOA. In this study, standard journals related to software engineering and service oriented architecture have been selected. This literature review on dependency analysis of SOA based system is conducted according to guideline proposed by Kitchenham [1].

The main characteristic of SOA is loose coupling and service contracts. Since individual service cannot perform a complex task, composite services are required which are composed by individual services. To create composite service it becomes essential to do dependency analysis carefully. 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 dependency may lead to catastrophic situation.

Our purpose of doing this review is to reveal the issues and challenges of dependency analysis of SOA based systems. SOA presents newer aspects of dependency analysis due to its different architectural style and programming paradigm. This paper surveys the previous work taken on dependency analysis of service oriented systems. This study shows the strengths and weaknesses of current approaches and tools available.
for dependency analysis task in context of SOA. There is similarity between service oriented development and component based development but we focus on the dependency analysis work related to software services. 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.

This paper is organised as follows. In section II, existing works related to review/survey of dependency analysis are mentioned briefly. In Section III, the concepts of SOA and dependency analysis are described briefly. In section IV, the research method for conducting the review is described briefly and research questions are identified. In section V and following subsections, the answer of each question is given as far as possible. In section VI, a brief discussion is given. Finally this paper is concluded in section VII.

II. RELATED WORK

In literature, the problem of dependency has been addressed widely but very few works has been observed regarding review work 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]. This paper contains extensive survey of dependency analysis and 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 requirements 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 [5] discusses the dependencies that exist between statements in that program and detailing several different classes of dependence analysis techniques. Bhuyan, Prakash and Mohapatra [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 [6]. Motlagh [7] has done a survey of testing of SOA based system. This paper has described about testing challenges of SOA based systems. 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. Li et al [9] have performed a survey of code based change impact analysis techniques. Bohner [52] has discussed impact of interdependency relationship among software components.

The above contributions demonstrate that although various approaches of survey and literature works are available in the literature but a systematic and disciplined literature review work of dependency analysis especially in context of SOA is yet not found. The proposed work extends the above contributions further.

III. SOA AND DEPENDENCY ANALYSIS

The one major problem that we 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].

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 [12].
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) and software component level root cause analysis (understanding the cause of a component by looking at the other components it relies on) [13,51]. 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 component 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. These interdependent services work as subsets of the system and is referred as the reduction technique.

IV. RESEARCH METHOD

The identification of research questions is the first step of systematic literature review (SLR). 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 solve dependency issue in SOA is essential. Complex Services may be generated using atomic services dynamically. To identify dependencies among services are essential because fault in atomic service would affect adversely the overall system.

A. Research Questions

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 are the proposed definitions of dependency of SOA based system?*

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

*RQ2. What are the available dependency solutions 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.

*RQ3. What are the relative impacts of syntactic and logical dependencies on the failure proneness of SOA based system?*

**Motivation:** Obtain the reasons of failure of SOA based system due to syntactic and logical dependency.

*RQ4. What is 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.

*RQ5. What are the available tools for dependency analysis in SOA based system?*

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

We have performed two fold manner of searching of research papers related to the topic i.e. sequential manner and random manner. In sequential manner, we search the papers from selected journals for certain duration. In this study, we have taken standard journals and proceeding of conferences from 2000 to 2012. SOA is a new paradigm and most of the significant works are observed after 2000. Because of this reason, journals and proceeding have been selected since 2000. The list of the selected journals and conference proceedings are given in Table I. 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.

B. Search Strategy

Aforementioned journals and proceedings have been opened manually. Researchers can be misguided by using title of research paper to select literature relevant to research questions. To go through full research
### Table I. Selected Journals and Conference Proceedings

<table>
<thead>
<tr>
<th>Source</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information and Software Technology</td>
<td>IST</td>
</tr>
<tr>
<td>IEEE Transactions on Software Engineering</td>
<td>TSE</td>
</tr>
<tr>
<td>ACM Computing Surveys</td>
<td>ACM Sur</td>
</tr>
<tr>
<td>ACM Transactions on Software Engineering Methodologies</td>
<td>TOSEM</td>
</tr>
<tr>
<td>Empirical Software Engineering Journal</td>
<td>EMSE</td>
</tr>
<tr>
<td>Proceeding International Conference on Software Engineering</td>
<td>ICSE</td>
</tr>
<tr>
<td>Service Oriented Computing and Application</td>
<td>SOCA</td>
</tr>
<tr>
<td>Proceeding International Conference on Service Oriented Computing</td>
<td>ICSOC</td>
</tr>
<tr>
<td>Proceeding International Conference on Next Generation Web Services Practices</td>
<td>ICNGWSP</td>
</tr>
<tr>
<td>Proceeding International Conference on Web Services</td>
<td>ICWS</td>
</tr>
<tr>
<td>IEEE Transactions on Service Computing</td>
<td>TSC</td>
</tr>
</tbody>
</table>

paper of every selected journals and conference proceeding is tedious and cumbersome task because of excessively irrelevant material. So it is wastage of time and energy. Our strategy is to go through only abstract and introduction of each paper. Using abstract and introduction it can be identified that which paper is relevant to aforementioned research questions. Relevant papers would be selected for detailed study. The abstract in which keywords `dependency` and `service` contains would be considered as potential research paper relevant to proposed research questions.

V. Observation

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

**RQ1. What are the proposed definitions of Dependency of SOA based system?**

In the present literature, the definitions of service dependencies are given in ad-hoc manner and vary widely in concepts. The dependency is assumed as a relationship of services offering functionality to other services. Winkler et al. [14] 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.” Service dependency is beyond traditional poor service description, and directed by various sources such as data, resource, procedure control, utilizing techniques etc [15]. Another definition of service dependency is given in [16] as ‘A component is said to depend on a service if it (may be temporarily) needs the functionality of that service.’ Most of the definitions describe dependencies in software services at the time of service composition. A summary of the types of the dependencies, for SOA based systems, given in the literature are listed below in Table II.

In SOA it is important to observe dependency at architecture level, design level and code level.

A. Architecture level dependency

When following architecture-driven strategies to develop large software intensive systems, the analysis of the dependencies is not a trivial task [3]. Typically, the level of detailed available for software services is limited, and as a result, dependency analysis of software services may be performed 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, choreography and orchestration of services. An efficient dependency management for service-oriented peer-to-peer architectures turns out to be indispensable for guaranteeing a stable environment in which applications are accomplished to operate in a reliable manner [22].
### Table II. Dependency Relationship Observed in Literature

<table>
<thead>
<tr>
<th>Types of Dependencies</th>
<th>Short Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Time Dependencies</td>
<td>Design time dependency refers to the dependency of a service as it exists after deployment but before execution.</td>
<td>Phukan [17]</td>
</tr>
<tr>
<td>Hidden Dependency</td>
<td>Service Dependency which cannot be identified directly by developer. It may create catastrophic situation in a critical system.</td>
<td>Leon et al. [18]</td>
</tr>
<tr>
<td>Sequential Dependency</td>
<td>Classical temporal language (CTL) such as Linear Temporal Logic (LTL), Computational Tree Logic (CTL) or (\pi) calculus have been suggested as appropriate notations for expressing sequential dependencies between services.</td>
<td>Halle et al. [19]</td>
</tr>
<tr>
<td>Technical Dependency</td>
<td>Technical dependencies between software services being developed by different developer create a need for communication and coordination between developers.</td>
<td>de Souza et al. [20]</td>
</tr>
<tr>
<td>Cyclic Dependency</td>
<td>Suppose there are three services (s_1), (s_2) and (s_3). If (s_1) is dependent on (s_2), (s_2) on (s_3) and (s_3) on (s_1), then it is cyclic dependency.</td>
<td>Omer and Schill [21]</td>
</tr>
<tr>
<td>Input/output Dependency</td>
<td>It occurs when a service requires/ provides data from/to another service.</td>
<td>Omer and Schill [21]</td>
</tr>
<tr>
<td>Constraint Dependency</td>
<td>It occurs due to user constraint.</td>
<td>Omer and Schill [21]</td>
</tr>
<tr>
<td>Cause and Effect Dependency</td>
<td>It occurs when a service has preconditions to be satisfied based on the effect of the other services.</td>
<td>Omer and Schill [21]</td>
</tr>
<tr>
<td>Structural Dependencies</td>
<td>It exists on the level of models and application executions.</td>
<td>Arias et al. [3]</td>
</tr>
<tr>
<td>Transitive Dependency</td>
<td>Assume there are three services (s_1), (s_2), and (s_3). If (s_1) is dependent on (s_2), (s_2) on (s_3) then (s_1) is dependent on (s_3).</td>
<td>Alda [22]</td>
</tr>
<tr>
<td>Static Dependency</td>
<td>This dependency occurs at code level in services. It is identified at compile time of services.</td>
<td>Arias et al. [3], Basu et al. [13]</td>
</tr>
<tr>
<td>Dynamic Dependency</td>
<td>It occurs at code level. It is identified at run time of services.</td>
<td>Arias et al. [3], Basu et al. [13], Keller et al. [23]</td>
</tr>
<tr>
<td>Horizontal Dependency</td>
<td>Dependency between atomic services.</td>
<td>Winkler et al. [14]</td>
</tr>
<tr>
<td>Vertical Dependency</td>
<td>Dependency between atomic service and composite service.</td>
<td>Winkler et al. [14]</td>
</tr>
<tr>
<td>Price Dependency</td>
<td>A price dependency exists when the price of a service depends on the price of other services.</td>
<td>Winkler et al. [14]</td>
</tr>
<tr>
<td>Location Dependencies</td>
<td>Location dependencies occur between two services that need to be executed at the same or at a different location.</td>
<td>Winkler et al. [14]</td>
</tr>
<tr>
<td>Resource Dependency</td>
<td>Two services have a resource dependency, when the availability of a resource, which is needed by one service, depends on another service.</td>
<td>Winkler et al. [14]</td>
</tr>
</tbody>
</table>

#### B. Design level dependency

Detection of inter-service dependencies at design time is crucial in order to enable effects analysis upon failure of a service inside a service composition [24]. 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. This analysis can also be used as a measure of quality for software services that software services are an acceptable design choice.

In MoDe4SLA approach [25], authors analyzed, at design time, the dependencies between different services with respect to the agreed response time and costs of the different services. Phukan [17] described design time dependency as the dependency of a process as it exists after deployment but before execution. Leon and Foss [18] have discussed hidden dependency among services. This paper has described that hidden dependency can affect system severely in critical services. Omer and Schill [21] have described input/output dependency, constraint dependency, cause and effect dependency under the category of design level dependency.

This dependency plays a major role to develop quality software. At this level different modules are constructed according to the basis of core SOA principle. Individual service cannot perform a business function so composition of services is essential for business functionality. In this case composite service is fully dependent on individual services. If individual servicedownload functions, then effect of this will reflect to the composite service.
C. Code level dependency

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 [21]. In order to keep system disturbance as low as possible while replacing existing services with their newer versions, engineers require an adequate illustration of how the connections and dependencies of services change in the running system [26]. The dependence discovery must be focused on capturing snapshots of dependence information relevant to each service request of concern, rather than finding statistical averages for long-term dependencies [28].

Q2. What are the available dependency analysis solutions for SOA based system?

There are various approaches available for dependency analysis of SOA based system in literature. Summary of these approaches are categorised as given in Table III.

<table>
<thead>
<tr>
<th>Dependency Analysis Solutions</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Based Approach</strong></td>
<td>Phukan [17], Omer and Schill [21], Alda [22], Espinha, Zaidman and Gross [26], Egyed [27], Wang and Capretz [29], Ensel and Keller [16], Baresi et al [30], Baloch and Crespi [31], Yan, Xu and Gu [32]</td>
</tr>
<tr>
<td><strong>Formal Modelling Approach</strong></td>
<td>Yanchuk et al [33], Winkler et al [14], Hasselmeier [34], Zhou et al [15], Basu, Casati and Daniel [13], Bodenstaff et al [25], Zheng, Zhou and Krause [35], Brogi and Popescu [36], Kuang et al [37], Yan, Chen Liu and Lei Liu [38], Steinle et al [39], Baloch and Crespi [31], Liu, Ma and Zhao [40], Tolkdorf [41]</td>
</tr>
<tr>
<td><strong>Quantitative Modelling Approach</strong></td>
<td>Cataldo et al [42], Binkley et al [43], Wang and Capretz [44], Tuura and Tylor [45], Wang and Capretz [29]</td>
</tr>
<tr>
<td><strong>Artificial Intelligence Approach</strong></td>
<td>Goehrung et al [46], Novonty et al [28], Ai and Tang [47], Ensel [48]</td>
</tr>
</tbody>
</table>

On the basis of Table III, we have described dependency analysis solutions briefly in the following paragraphs:

D. Graph Based Approach

In an SOA based system, with a few involved services; it would be easy to understand the dependencies among services by their textual description. 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. Phukan [17] 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. Omer and Schill [21] have proposed a method of automatic composition plan based on automatic extraction of dependencies among services. These services are represented using a directed graph. Their approach utilizes existing graph traversal based algorithms to extract cyclic dependency and generate the execution plan. The graph that models the dependency has n nodes where n equals available services to form the composite service and edges represent the dependency link. The edge direction indicates the service dependency flow. Alda, in his work, [22] has represented dependencies among services in a graph like structure and shown a graphical visualization of existing dependencies between provider and consumer peer...
services. The annotations on the edges denote dependency values that declare the importance of a dependency between provider and consumer services. Espinha, Zaidman and Gross, in their work, [26] have created a dynamically updateable time based dependency graph to visualize the run time topology of software services. Egyed [27] have discussed a scenario driven approach for dependency analysis of SOA and presented trace dependencies analysis which is based on graph. Wang and Capretz [29] has identified service dependencies at inter- and intra- service levels and develop graph-based service dependency matrices. Based on the calculation of the matrices, the ripple effect that indicates dependencies and the impact of the change on the service can be analyzed and quantitatively measured. Ensel and Keller [16] have given an approach of computing dependency graph in web based architecture for retrieving and handling dependency information from various managed resources. This approach computes of dependencies between services and applications across different systems and domains, i.e., establishing a ‘global’ service dependency model and enabling system administrators to navigate through the resulting directed graph from the top to the bottom and in reverse order. Romano, Pinzger and Bouwers [49] 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. The information contained in these graphs can help software engineers to measure important quality attributes (e.g., analyzability and changeability) for measuring maintainability of the system under analysis. Baresi et al [30] have proposed to model service oriented architecture as graphs, represented as instances of UML class diagram and to describe their configurations by graph transformation rules. Baloch and Crespi [31] have presented a graph based approach to determine dependency relations among various services in ubiquitous environment to help better analyze the pervasive services. Yan, Xu and Gu [32] have proposed an approach to resolve the composition problem over large-scale services. They used an inverted table as index for a quick service discovery, and applied a Service Dependency Graph(SDG) and an AND/OR graph as the algorithm basis for parallel composition.

E. Formal Modelling Approach

The importance of formal modelling of syntactic and semantic characteristics of software systems has long been accredited. It is practicable to use mathematical foundations to capture the essential behaviour of service oriented software systems for the purpose of dependency analysis. Some major contributions of this approach are given below. 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.[14] have proposed an approach for analysing 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 and determining the effects of events such as service failure or SLA renegotiation on other services. Hasselmeyer [34] has described the need for a generic dependency management approach, identifies the key properties of dependencies as well as the requirements for dependency management schemes, describe the model derived from the requirements, and presents an implementation using the Jini connection technology. In [15], the authors describe a dependency-aware service oriented architecture (DSOA) solution to the issue of service composition that allows the service composer to compose and register a service by analyzing and designing the dependency between existing Web services. They further analyzed and designed service dependency ontology of software services. Basu, Casati and Daniel [13] identified the importance of the problem of discovering dynamic dependencies among Web services and proposed a solution for the automatic identification of traces of dependent messages, based on the correlation of messages exchanged among services. It identifies dependencies between web-services based on real time log data under the context of the HP SOA Manager system. Bodenstaff et al [25] demonstrate how to analyze Service Level Agreements (SLAs) during development phase and how to monitor these dependencies using event logs during runtime. They called this approach MoDe4SLA (monitoring dependencies for SLAs). Zheng, Zhou and Krause [35] have defined ‘A Web Service Automaton’ and with help of this they analysed internal and external data dependencies in web services. Brogi and Popescu [36] have presented a methodology which, given a set of service contracts, tries to construct an aggregation of such services and perform a control flow and data flow dependency analysis of an aggregated service. Yan, Chen Liu and Lei Liu [38] have given an approach to discover operation dependencies. This approach combines the semantic matching of inputs and outputs interfaces between service operations and the analysis of process cases to identify dependencies. Kuang et al [37] explain the importance of importing
dependencies into service discovery and introduce two types of dependency between interfaces of services. Steinele et al [39] have developed non-intrusive and scalable techniques to discover dependencies between components of a distributed system by mining logs. However, the approaches fail to detect the direction of dependency and distinguish the direct and indirect dependencies [39]. Baloch and Crespi [31] have presented a time efficient approach to determine dependency relations among various services in ubiquitous environment to help better analyze the pervasive services. Liu, Ma and Zhao [40] have proposed an approach to identifying conversation dependency between software services to facilitate the dynamic evolution. 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 among services. Tolksdorf [41] have proposed a Dependency Markup Language to capture dependencies amongst activities and generalization/specialization among web services.

F. Quantitative Modelling Approach

Metrics helps in decision making and problem solving and enables the users to compute the desired property based on a defined set of statements and its mathematical formulation. Significant works have been observed in metrics based dependency analysis solutions of SOA. Cataldo et al. [42] has proposed metrics for measuring syntactic dependencies, logical dependencies and work dependencies among software services. Binkley et al. [43] have discussed quantitative approach using dependence clusters. Wang and Capretz [44] have discussed some metrics for dependency analysis in SOA such as : element entropy, dependency entropy and system entropy. The entropy value of a service represents its degree of dependency with other services [44]. Tuura and Tylor [45] has proposed a tool named ‘Ignominy’ for dependency analysis of large software. Ignominy analyses static dependency. This tool has potential for dependency analysis of SOA based system. Wang and Capretz [29] have defined some metrics measure of dependency analysis: Intra-Relation, Inter-Dependency, Inter-Relation, and Inter-Dependency focus on the quantity analysis of dependency while cohesion concentrates on quality analysis.

G. Artificial Intelligence 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 categorised under artificial intelligence technique. Goehring et al.[46] have described the development of a knowledge-based system approach to enhance program comprehension of Service Oriented Architecture (SOA) software. This approach uses rule-based methods to automate the analysis of the set of artefacts involved in building and deploying a SOA composite application. The rules codify expert knowledge to abstract information from these artefacts to facilitate program comprehension and thus assist Software Engineers as they perform system maintenance activities. Novonty et al. [28] 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. Ai and Tang [47] have presented 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. Ensel [48] used neural networks to automatically derive dependency information by looking at pairs of systems’ behaviour over time.

RQ3. What is the relative impact of syntactic and logical dependencies on the failure proneness of SOA based system?

If dependencies exist between 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 these 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 [13]. The knowledge of dependencies between services may further be useful for the prediction of impacts on other services due to management operations. This is of particular interest in the typical ‘repair – scenario’ where a service implementation has to be shut down temporarily: - it might be essential to know the effects on other services beforehand [48]. The analysis of dependencies on consumer peer services is
important for supporting the adaptation of a public service. The uncoordinated adaptation of public services potentially leads to malfunctions in the environment of depending peer services in SOA based systems [22]. An important aspect is that resource dependencies have a transitive nature. Thus, the failure of one service may not only affect the direct consumer but all other services which handle the same resource [14]. Dependency analysis is very critical for testing, maintaining and composing of services. Syntactic Dependency is a parsing scheme where we create edges between services [50]. Logical dependencies impacts more than syntactic dependencies among services. Logical dependencies play adverse role in software failure [42]. The ripple effect measure has been identified as an important method for software impact analysis models [51, 52]. 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. System may be fail adversely because of syntactic and logical dependencies.

RQ4. What is 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 issues, 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. A careful reading of literature helps in identifying research efforts that have gone into classification of concepts and understandings regarding dependency analysis of SOA based systems.

The management of work dependencies is critical in SOA based systems. The automated service dependency is a challenging problem in the administration of large distributed system. In real life the services in a composition behave differently. These differences arise, for example, because content of services is different, and they are offered by different providers. The difference in behaviour, together with the structure of the composition, makes it difficult to manage the dependencies between services [53]. 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 [13]. 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 flavours [13].

The number of dependencies between many involved systems can be computed, but may become very large. From an engineering viewpoint, it is often undesirable and sometime impossible to store a complete, instantiated dependency model at a single place [16]. The question where the dependency information on the managed resources comes from is another crucial issue [16]. How notions of dependencies can be related by specialization to lead to unified hierarchy [53]. 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 [38].

RQ5. 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 know the strengths and weaknesses of various tools used for congestion control. The idea is to get to know that weather existing tools are sufficient for the purpose or it is require revising the tools or proposing a new one in context of newer requirements.

There are various tools available for dependency analysis like JDepend [54], DepAn [30] etc. But we focus on only those tools especially meant for dependency analysis for SOA based systems. These tools are listed in Table IV.

The dependencies among services can be explored in tools such as HP OpenView 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 [13].

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 [17]. SOA Dependency Analyzer [58] 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 [55, 58]. Serviz [60] 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.

Spassov et al [55] have implemented a Data Dependency Analysis Tool (DDAT) to support the testing, validation and verification of both functional and non-functional behaviour 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 [56] 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. Briand et al. [51] has described impact analysis and change management using UML diagram.

DepAn [57] 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 [59] 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. DISCUSSION

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.

VII. CONCLUSION

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.

ACKNOWLEDGMENT

We wish to thank UGC to give financial support as Fellowship to do this research work. We also thank Department of Computer Science, Banaras Hindu University, Varanasi for providing infrastructure and facility for research work.

REFERENCES


80