Online Interaction Analysis Framework for Ad-Hoc Collaborative Processes in SOA-Based Environments

Hong-Linh Truong and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology
{truong,dustdar}@infosys.tuwien.ac.at

Abstract. Today's collaboration in e-science and business environments is no longer limited to the boundary of a single organization. People belonging to different organizations collaborate together to achieve a common goal by establishing virtual teams. The current trend is to rely on SOA-based tools and services for virtual teams and their collaborative processes because SOA offers many technologies to simplify the integration and interoperability of services belonging to different organizations and to allow the user to easily access existing services and tools. In complex environments comprising distributed software services and people, we need to understand how people and software services interact in order to adapt activities and services to the change of their operating environments as well as to allow them to self-manage their behaviors during the collaboration. We observed that there is a lack of tools supporting the analysis of interactions in such ad-hoc collaborative processes at runtime. In this paper we present our VOIA(Vienna Online Interaction Analysis) framework which aims at analyzing interactions within collaborative processes in SOA-based environments. We present a comprehensive list of interaction metrics and patterns associated with ad-hoc collaborations and techniques used to determine these metrics and patterns. We discuss how metrics and patterns can be used in process adaptation and illustrate VOIA's capabilities with several experiments.

1 Introduction

Today's collaboration in e-science and business environments is no longer limited to the boundary of a single organization. People belonging to different organizations collaboratively work together to achieve a common goal. In this context, they establish a *team*, often a *virtual team* [1], and conduct a *collaborative process* implementing their common goal. The current trend is to rely on SOA (Service-Oriented Architecture) to implement tools and services for virtual teams and their collaborative processes because SOA offers many technologies to simplify the integration and interoperability of services belonging to different organizations and to allow the user to easily access existing services and tools. Examples of such SOA-based collaboration tools and services are the inContext [2], ECOSPACE [3] and COIN [4] systems. Given these systems, one important aspect is to understand how *people and software services interact* in order to adapt activities of people and software services to the change of their operating environments as well as to allow them to self-manage their behaviors during their collaboration. Hence, metrics and patterns associated with interactions are a valuable source of information. An interaction metric is a quantitative measure that can be used to characterize

K. Jensen and W. van der Aalst (Eds.): ToPNoC II, LNCS 5460, pp. 260–277, 2009.

and evaluate how an individual service or human is involved in interactions with another service or human[1]. We consider an interaction pattern as a reoccurring structure of interactions that is analyzed from a set of interactions, for example, the interactions among a set of services might follow a one-to-many model [5].

While existing research has been focused on defining and detecting patterns in workflows [5,6,7,8,9,10,11], most of them concentrate on rigid, well-defined processes and workflows in businesses and aim at supporting offline workflow mining. Support for runtime analysis of patterns in dynamic collaboration environments has got little attention. In complex, dynamic collaboration environments, ad-hoc collaboration processes are not pre-defined; their dynamic, ad-hoc activities are defined on-demand. These activities may be combined with well-defined workflows and composition patterns, but not necessarily. In such environments, metrics and patterns characterizing the collaboration and its activities are relevant because they can provide valuable insights into the collaborative process to support runtime adaptation. However, existing offline mining techniques are not suitable because they are not designed (and cannot be tested) with evolving collaborative processes. Existing mining techniques typically require complete log data and do not deal with runtime aspects, such as runtime processing data from various services and runtime provisioning of interaction metrics and patterns. Furthermore, most existing work focus on either human-to-human interactions (e.g., social networks analysis) or service-to-service interactions (e.g., performance analysis or service interaction pattern analysis), whereas SOA-based collaboration environments include diverse types of interactions among humans and services that should be considered together.

Runtime analysis of interactions in such environments poses many research challenges. First, data is obtained and analyzed while the collaborative process just continues to evolve as new activities emerge. This requires us to deal with different types of events collected at different levels, such as activities, interactions and service-specific events. Secondly, metrics and patterns have to be determined for both humans and software services according to different needs of clients, such as determining metrics and patterns based on collaboration contexts, e.g., in individual, group or the whole collaboration levels, and on user-specific conditions, e.g., in particular time period and with a specific threshold. All these challenges imply that runtime analysis frameworks must be flexible and customized: new metrics and patterns analyses can be easily added and analysis requests are user-customized. Currently, there is a lack of such frameworks supporting tools for detecting metrics and patterns in collaborative work. Providing such metrics and patterns to clients at runtime is what motivates our work. In this paper, we contribute to techniques to support online interaction analysis for collaborative processes in SOA-based environments, namely (1) a classification of interaction metrics and patterns covering human-to-human, human-to-service, and service-to-service interactions at different levels, and (2) the design and implementation of a flexible and customizable software framework supporting runtime interaction analysis.

The rest of this paper is organized as follows: Section 2 presents related work. We describe a motivating scenario in Section 3. Preliminaries on terminologies and models are presented in Section 4. Section 5 describes a holistic view of interaction metrics and patterns. The architecture and implementation of VOIA is presented in Section 6.

[1] For metric definition, see http://en.wikipedia.org/wiki/Metrics, last access: 26 August 2008.

Interaction analysis techniques are detailed in Section 7. Section 8 presents experiments to demonstrate VOIA. Section 9 summarizes the paper, presents remaining issues, and gives an outlook to the future work.

2 Related Work

In our work, we develop techniques to determine metrics and patterns in collaborative processes carried out by teams. A collaborative process includes various activities which are associated with humans, artifacts, and software services. Our approach is to study the interaction at multiple levels of abstraction, from the way how humans use services, to how services interact with each other, and how a human interacts with another human through the utilization of services.

Some existing works analyze and define patterns for workflows and services [8,5] and (performance) metrics for workflows [12]. Understanding workflows and business processes has attracted a lot of research efforts. This includes, for example, research on performance monitoring and analysis of workflows [13,12] and on mining of workflow logs [6]. Tools and techniques for monitoring and analyzing performance of Web services focus on extracting logs comprising of service invocations and analyzing the logs to provide performance metrics. However, they focus on metrics associated with individual services, rather than with interaction patterns. When analyzing performance metrics associated services, we apply well-defined metrics from existing works, such as in [12], and focus on the analysis of patterns.

Recent work on mining workflow logs, especially work done by van der Aalst et al., has introduced several novel techniques to determine the relationship between humans and services. ProM [6] introduces many process mining features. Information can be shown, e.g., using social network and clustering. Several issues such as analyzing the conformance between a process model and its execution logs [14] are addressed. [15] discusses how to apply process mining to less structured processes in CSCW (Computer Supported Cooperative Work) systems. These works, however, do not support online analysis of dynamic collaborations. We note that although activity-based collaboration systems exist [16], they do not support the analysis of the collaboration.

In our previous works [17,11,10], we focus on the analysis of patterns in well-defined workflows and email activities. In [17], we have discussed the importance of mining patterns at multiple levels of abstraction. These papers addressed Web service logging, workflow structure discovery and session reconstruction. Patterns, like proxy and broker, and social network were presented and patterns, like proxy and broker, were detected from social networks. We have reused previous patterns. However, neither the analysis of patterns in collaborative work in SOA-based environments nor the online interaction analysis has been performed. In particular, the analysis of patterns in this paper differs from these works. Patterns are not detected from social networks but from streams of events during runtime. Furthermore, these works focus on the analysis of particular patterns, rather than provide a classification of patterns and metrics and a generic software framework which is flexible and customizable.

Recently, the concept of complex event processing (CPE) [18] has been utilized to analyze complex events in SOA environments. However, CPE frameworks, such as Esper [19], just provide fundamental tools for handling complex events and existing work

focuses on detecting service behaviors through events. We utilize Esper for filtering and matching events and implementing primitive metrics and patterns analysis.

In this paper, we focus on analyzing patterns for collaborative work, rather than on the definition of patterns which are well addressed elsewhere [8]. One of the major differences between our work and existing work is that existing work focuses on offline analysis with the assumption that logs are produced by existing workflow systems. Our work differs as we focus on online analysis. Our assumption is that in modern collaboration, collaborative work is continuously being performed. Thus, interaction analysis tools are required during runtime.

3 Collaboration Scenario and Research Approach

In our work, we consider dynamic collaboration environments in which teams utilize different collaboration services for their collaborative work. Team members define and perform activities which require the involvement of many other services and humans. The upper part of Figure 1 illustrates the dynamic collaboration environment, for example, in the inContext project [20]. In such an environment, both humans and software services exist. Humans use services to perform their activities, while services can interact with each other to fulfil requests from humans. Services will follow the SOA model, thus they can be easily integrated together, providing seamless access virtually from anywhere. The SOA-based service model also simplifies the monitoring and maintenance of services, making the acquisition of multiple sources of log information at different levels in the widely distributed system possible and easier.

Using services, people perform their collaboration, of which processes are typically not modeled beforehand. Furthermore, the execution of collaborative processes is continuous and evolving, thus in many cases we will not know in advance when a process finishes. Interactions between humans and services are highly concurrent and distributed as multiple activities are executed in parallel. These activities involve services and people spanning different geographical locations and organizations. Therefore, offline analysis, which either requires complete, centralized information or the completion of the process in order to analyze the process, is not suitable. In addition, the need for adapting activities of and resources for collaboration is required at runtime, due to highly dynamics of modern collaborations (e.g., team member is often on the move). Thus, online analysis techniques are more suitable. The middle part of Figure 1 shows that the *Online Interaction Analysis* analyzes log events obtained from services and provides metrics and patterns back to the services in the dynamic collaboration environment.

However, online interaction analysis for such an environment is a very challenging task due to highly concurrent and distributed actions, such as concurrent executions of distributed services, concurrent interactions among services and humans, and concurrent requests from various clients. Which metrics and patterns associated with interactions are useful for optimizing collaborative work and resources used for the work at runtime? How can we correlate metrics and patterns from different levels of collaboration context, such as individual, group and the whole collaboration? How do we support the customization of metrics and patterns analysis so that different clients can

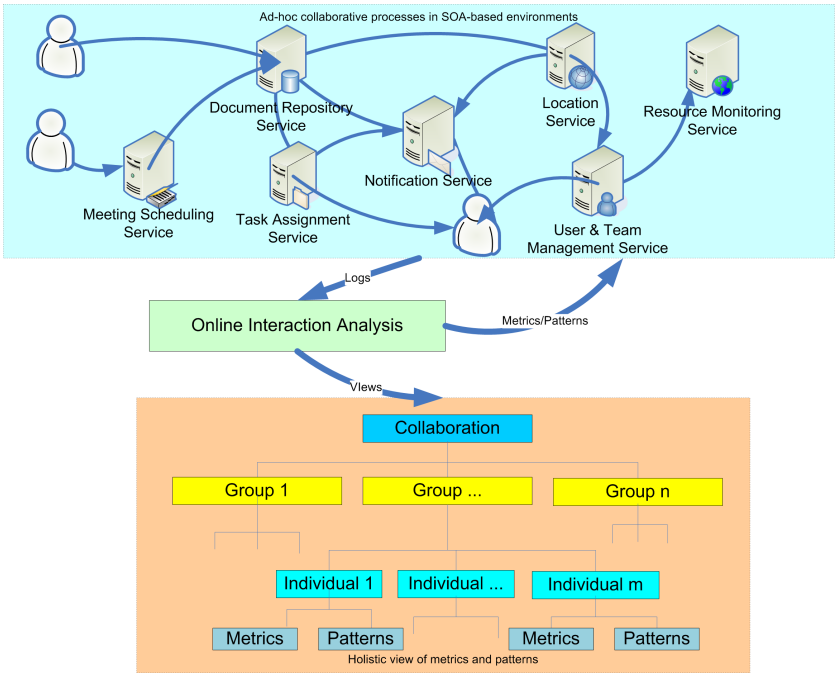

Fig. 1. Online interaction analysis environment and metrics/patterns view

utilize the metrics and patterns differently? How to handle diverse events from various services? How to manage and provide these metrics and patterns to concurrent clients at runtime? We propose to provide a holistic view of patterns and metrics associated with services and interactions at multiple levels, ranging from the *individual* (services or human), to the *group* (of humans or services or the mix of them), and to the *whole collaboration*. The bottom part of Figure 1 shows the holistic view of interaction metrics and patterns that are associated with interactions. Then, we determine metrics and patterns based on different levels, client requests and periods of time, and provide and manage XML-based metrics and patterns at runtime.

4 Models

Before describing metrics and patterns and the analysis framework, in this section, we present our models of activities and services and humans in collaborations and data required for the interaction analysis.

4.1 Activity, Service, and People in Collaboration

In our model, *collaboration* is defined as a joint work between different people. Collaboration can be simple, e.g., including only a single task like `review of a document`, or complex, e.g., a real project. An *activity* describes a task of a collaboration, for example, `review a document`. An activity may consist of sub activities,

but we do not distinguish between atomic activity and composite activity. An activity specifies various related information required for the execution of the activity. An *activity instance* represents all information associated with a particular execution of an activity; information includes, for example, start and end times, initiator, and associated service instances. An activity might be associated with a set of activity instances. Given an activity, we capture all changes in activities and activity instances in *activity events*.

A *collaboration service* is a service that is used in the collaboration; collaboration services are involved in the execution of activities. Collaboration services can be used to communicate between two team members, such as `Notification Service` and `Instant Messaging`, to host files such as `Document Repository Service`, and to store and manage activities such as `Activity Store Service`. Furthermore, there are middleware services which provide facilities for the operation of the collaboration such as `Service Registry` and `Logging Service`. In our model, we assume that services are well-defined based on the SOA principle. Most services are SOAP-based or RESTful (however, in our implementation, we tested only with SOAP-based services). *Service instance* is a particular deployment and running of a service in a particular hosting environment. A service invocation is a particular invocation of a service operation of a service instance. Information about service invocations is captured in *interaction events*. An interaction event consists of information related to the invocation, such as request and response message, service endpoint reference, consumer endpoint reference, etc., which are captured at the level of the hosting environment without the knowledge of service instances (e.g., by using SOAP intercepting mechanism). Furthermore, a service instance can also provide application-specific events about its operation. We call such events *service events*.

During a collaboration, people can initiate an activity, perform an activity or receive a message and handle the message sent by other people. In our model, we assume that a person defines a flow of activities that he/she has to perform. In our work, activities within a collaboration might be modeled in advance and executed in a pre-defined order or defined on-demand. Therefore, we do not assume that the structure of the collaboration and its execution order are known beforehand. Rather, we consider a collaboration to be a set of activities. The goal of this paper is to use online analysis techniques to detect metrics and patterns associated with interactions among humans and services in collaborations, but not on the detection of the process structure of the collaboration (like the work on discovering workflow structure from logs [21]). How to execute, manage and change the activities and the flow of activities are beyond of the scope of this paper.

4.2 Data for Interaction Analysis

To analyze collaboration processes, we rely on three sources of events: activity events, interaction events, and service events. In the following, we discuss the structure of data used in our online interaction analysis.

Each activity in our model is identified by a unique `activityURI`. When an actor performs an action to change the status of an activity, such as executing or delegating the activity, an event will be fired. In principle, we can use any activity model in collaboration, such as IBM UAM (Unified Activity Model) [16], Caramba [22] or the

inContext Activity Model[2] to define activities. In our implementation, we use the inContext Activity Model which can be used to describe artifacts, involved people, and resources in detail. Given an activity event describing actions related to activities and activity instances, we assume that the activity event includes $activityURI$.

As mentioned before, an *activity event* describes changes in an activity and activity instance. This kind of event is captured by the `Activity Store Service` which maintains existing activities and by `Activity Execution Service` which executes the activity. Listing 1.1 presents a sample of an activity event which indicates that an activity has been created.

```
<activityEvent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
actionURI="http://www.in-context.eu/ns/action/tCoordinationAction#6575"
activityURI="http://www.in-context.eu/Activity/Activity#226"
timestamp="1207840402595">
    <ExecutedByFoafAgent>
     http://www.vitalab.tuwien.ac.at/projects/incontext/TEST_LINH1#Martin
    </ExecutedByFoafAgent>
    <CoordinationType>
        <ActivityChangeType>Create</ActivityChangeType>
    </CoordinationType>
</activityEvent>
```

Listing 1.1. Example of an activity event

The second source of data used in the analysis is the *interaction event* which is captured in the Web services hosting environment. By using Web services handlers and SOAP interceptors, we can collect low level data of Web services invocations, such as SOAP messages. From this data, interaction events are generated and provided as an input for the analysis process. Activity events can be correlated to interaction events by using `activityURI` (for example, in the inContext project, the client modified the SOAP message header to include activity-related information). Listing 1.2 presents an example of an interaction event.

```
<InteractionEvent>
 <clientEndpoint>85.18.48.34</clientEndpoint>
 <messageCorrelationID>000a1460-25ba-4fa8-b766-9c3b50aa8c2b</messageCorrelationID>
 <messageType>Response</messageType>
 <serviceEndpoint>http://srvweb02.softeco.it/cgi-bin/SOAP.cgi/Eadt/Tasks/DocService
 </serviceEndpoint>
 <eventSourceID>AL-invoke@128.131.172.208</eventSourceID>
 <timeStamp>1207212091812</timeStamp>
</InteractionEvent>
```

Listing 1.2. Example of interaction event in which not all entries are available

The third source of events is *service event* which is provided by specific services. This type of events is optional and dependent on specific services. Since our goal is to provide a client-customized mining system, VOIA is designed to accept specific service events. However, the analysis of the patterns and metrics related to service-specific events are left to the client.

[2] http://www.in-context.eu/uploads/files/20070530D4.2v1.0Design20and20implementation20of20Context20Tunnelling.pdf

5 A Holistic View of Interaction Metrics and Patterns

Understanding interactions among humans and services in collaborative work highlights characteristics of not only individual humans and services but also of groups of them as well as the whole collaboration. For providing useful information in understanding interactions and in adapting the collaboration based on interactions, we focus on defining and providing quantitative information associated with interactions.

We classify three kinds of interactions associated with humans and services within the dynamic collaboration environments:

- *Service-to-service interaction*: is the interaction between two services, e.g., a service s_i calls another service s_j.
- *Human-to-service interaction*: is the interaction between a human and a service, e.g., how services are selected and used by a human. By saying human-to-service interaction, we mean a person needs and uses a service for his/her activities.

Table 1. Examples of interaction metrics and patterns for service-to-service

Level	Metric/Pattern Name	Description
Individual	ExecutionTime	The average execution time of a service.
	NumServiceCalls	The number of invocations of a service.
	NumUnavailableCalls	The number of times unavailability.
	NumFailureCalls	The number of failures.
	NumConsumers	The number of consumers that call a service.
Group	ServiceInteraction	The interaction between two services, $ServiceInteraction(s_i, s_j)$, represents the number of times that service s_i calls service s_j.
	UsageDistribution	The percent of usages distributed among services.
	UsageIsolatedPattern	Reflect whether a service is typically used in an isolated manner. Let S_j be a set of services invoked by a service s. Service s is in an isolated manner when $count(S_j) < \tau$ where $ServiceInteraction(s, s_j) > 0, \forall s_j \in S_j$. τ is a user-defined threshold and $count(S_j)$ is the number of services in the set S_j .
	UsageCompositePattern	Reflect whether a service is typically used together with other services in activities. Let S_j be a set of services invoked by a service s. Service s is in a composite manner when $count(S_j) > \tau$ where $ServiceInteraction(s, s_j) > 0$, $\forall s_j \in S_j$, τ is a user-defined threshold.
	OneToManyPattern	Related to one-to-many invocation, also called as one-to-many send, multicast, or scatter [5]. Let S_j be a set of services invoked by a service s. Service s and S_j are in this pattern when $ServiceInteraction(s, s_j) > \tau$, $\forall s_j \in S_j$, within a period of time t. t and τ are user-defined values.
Collaboration	as in the Group level	Similar to those in the group level but they are determined based on all information available in the collaboration.

Table 2. Examples of human-to-service interaction metrics and patterns

Level	Metric/Pattern Name	Description
Individual	UsageTime	Describe how much time that a human uses a service.
	NumHumanServiceCalls	Number of service invocations initiated by a human.
	TypicalServiceUsageTime	Identify the typical usage time that a human uses a service by eliminating outliers and calculating the mean value of a data set of UsageTime for the service.
	UsageCompositePattern	Reflect whether a set of services is used together by a human. Services $\{s_1, \cdots, s_n\}$ are considered in this pattern when they are called by a human h in a predefined period of time.
Group	HumanServiceInteraction	The interaction between a human and a service, $HumanServiceInteraction(h, s)$, represents the number of times that human h calls service s.
	UsageDistribution	Given a human, it reflects the usage distribution among services he/she calls.
	DurationUsage	Determine the typical usage time that a human uses a service.
Collaboration	as in the Group level	Similar to those in the group level but they are determined based on all information available in the collaboration.

– *Human-to-human interaction*: is the interaction between human and human, e.g., how a team member interacts with another one in order to perform activities. We determine interactions between two persons only by means of analyzing services they use in their communication and collaboration, e.g., `Notification Service`.

This classification differs from other works which focus on either human-to-human interactions (social networks) or service-to-service interactions by considering all types of interactions among humans and services. This consideration is necessary as these types of interactions are inherent in collaboration processes. Metrics and patterns are associated with these types of interactions. However, unlike other works which typically do not consider global versus local views on metrics and patterns, in our work, for each type of interaction, metrics and patterns are determined for specific time period at three levels: *individual* (for individual human or service), *group* (a team or a set of services), and *collaboration* (all available services and humans within a collaboration). This way takes into account the context of the collaboration in determining the metrics and patterns. The main reason is that interactions of a particular human or service are dependent on the context of the collaboration, such as a human might typically act as a proxy in a group but not in another group or within a set of services, a service might be well utilized, but not in the global view. Therefore, metrics and patterns characterizing the interactions should be determined differently. The metrics and patterns associated with three types of interactions at three levels provide a holistic view of interaction metrics and patterns in VOIA. This view allows us to utilize metrics and patterns differently, dependent on specific purposes and contexts.

Table 3. Examples of interaction metrics and patterns for human-to-human interaction

Level	Metric/Pattern Name	Description
Individual	NumCallers	Number of callers.
	NumCallees	Number of callees.
	NumInteractions	Number of interactions.
	NumAssignedActivities	Number of assigned activities.
	NumDelegatedActivities	Number of delegated activities.
	BrokerIndicator	Determines the broker role of an individual.
	ProxyIndicator	Determines the proxy role of an individual.
	MasterIndicator	Determines the master/slave role of an individual.
Group	TotalInteractions	Total number of interactions.
	AvgNumCallers	Average number of callers in a group.
	AgvNumCallees	Average number of callees in a group.
	AvgNumHumanInActivity	Average number of human involved in an activity.
	BrokerPattern	Related to broker pattern [10], including number of broker patterns, the structure of the broker pattern.
	ProxyPattern	Related to proxy patterns [10], including number of proxy patterns,the structure of the proxy pattern.
	MasterSlavePattern	Related to master/slave pattern [10], including number of master/slave patterns, the structure of the master/slave pattern.
	CoauthoringPattern	Related to co-authoring pattern, including the structure of co-authoring pattern. Two members h_i and h_j are in a coauthoring pattern when they both work on the same n activities, n is a user-defined value.
Collaboration	as in the Group level	Similar to those in the group level but they are determined based on all information available in the collaboration.

Note that the list of patterns and metrics is non exhaustive and many of them are common, well-understood and presented in literature [17,11,10,6,8,5,12]. Depending on specific needs, these patterns and metrics are utilized differently. However, VOIA aims at providing a rich catalog of metrics and patterns suitable for different clients. We would like to stress that the main objective of VOIA is not to define patterns and metrics but provide a framework which the determination of new patterns and metrics can be easily plugged in.

Table 1 presents metrics and patterns associated with service-to-service interactions. Many performance metrics at individual level are well-defined, e.g. in [12]. At the group level, we determine many novel patterns.

Table 2 describes metrics and patterns related to human-to-service interactions. Metrics and patterns are determined by analyzing humans who initiated or invoked services in their activities.

Table 3 describes metrics and patterns related to human-to-human interactions. In particular, we focus on metrics and patterns associated with broker, proxy, master/slave interactions.

6 VOIA Architecture and Implementation

Figure 2 depicts the architecture of VOIA which includes many components and provides Web services interfaces. In our architecture, the *Event Preprocessing* of VOIA accepts different types of events, such as `Activity Event`, `Service Event`, and `Interaction Event`, and pre-processes these events suitable for VOIA's *Metric/-Pattern Detection*. These events are collected at different services, such as `Activity Service`, `Logging Service`, `Communication Service`, and `Access Layer`. As mentioned before, events consist of data collected at different levels, ranging from Web middleware/container level (by using Web Handlers and SOAP interceptors, such as `Interaction Event`) to application level (by capturing application logs, such as `Service Event`).

Events processed by *Event Preprocessing* are passed to *Metric/Pattern Detection* which determines primitive metrics and patterns. The *Metric/Pattern Detection* analyzes events based on a set of pattern specifications and templates stored in *Pattern Specification and Template* to identify which metrics or patterns occurred. Such determined metrics and patterns are passed to the *Interaction Analysis* which provides high level analysis of patterns and metrics for interactions. During runtime, clients of VOIA can specify pattern specifications and submit the specifications to the VOIA service which informs the clients when patterns/metrics met the specifications exist. Thus, the clients can utilize the resulting metrics and patterns for online adaptation. The end-user can use the VOIA portal to manage VOIA and explore results produced by VOIA.

We have implemented the core of VOIA and provide it as a Web service based on JAX-WS[23]. The Web service provides fundamental interfaces for other clients to send events and pattern specifications and templates. VOIA can subscribe other services, such as `Logging Service`, to be informed with events or any services can send events to VOIA via a pre-defined interface. The *Metric/Pattern Detection* employs the Esper engine [19] to process events based on pattern expressions.

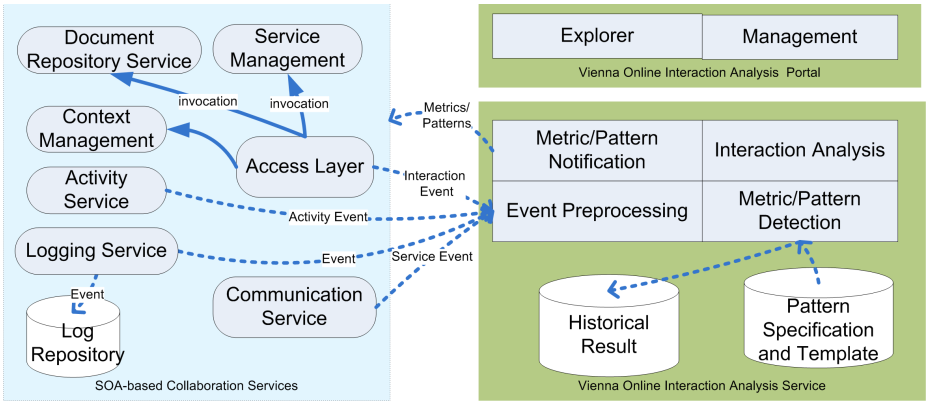

Fig. 2. Vienna Online Interaction Analysis (VOIA)

7 Analysis Techniques

The fundamental difference between our approach and existing ones is that our interaction analysis is conducted online. The metrics and patterns discussed in Section 5 are provided by either *Metric/Pattern Detection* or *Interaction Analysis* components, depending on the complexity of the analysis. The first component is used to determine primitive metrics and patterns which can be identified by applying pattern specification and template directly. The latter implements complex analyses which require different algorithms besides the primitive metrics and patterns.

As the data used for the analysis concurrently arrives in a stream of events, primitive patterns and metrics will be determined on the fly. For any analysis, a time window - specifying a period of time - or space window - specifying a number of events - or a combination of time and space windows will be specified. Together with the level (individual, group, or the collaboration), they determine the context of the analysis.

7.1 Primitive Metric and Pattern Detection

The *Metric/Pattern Detection* utilizes a set of predefined pattern specification and pattern templates in order to determine relevant metrics and patterns. We have defined several pattern specifications and templates based on that well-known metrics and patterns can be determined. Each pattern specification or template is described by

- *Pattern name*: is used to identify a pattern specification or template
- *Result handler*: is used to handle the result of a pattern specification or template.
- *Pattern expression*: is used to filter events and determine primitive metrics and patterns. It is described in Event Processing Language (EPL) [24].

All pre-defined pattern specifications and templates are stored in *Pattern Specification and Template* in XML form. A new pattern template can be defined by specifying the above-mentioned information. Unless we need specific treatment for pattern result handler, generic handler can be used. A result handler actually implements the algorithm to determine the corresponding pattern/metric. Result handlers adhere a pre-defined interface defined by VOIA so that a new handler can be plugged into VOIA. Note that VOIA provides mechanism for writing handlers.

Given a pattern specification/template, VOIA will create an EPL statement and a result handler object using a reflection mechanism. When events arrive to VOIA, in the *Event Preprocessing* component, a CEP engine which is developed atop Esper will use pattern expressions to process as well as to filter relevant events. Then, the engine passes processed events to corresponding result handlers. Depending on pattern expressions, some primitive patterns can be detected in the *Event Preprocessing* whereas the result handler will process patterns and metrics and provide results. The result is described in XML. For primitive metrics and patterns, we provide a generic result handler that provides the result of a specification or template. Note that by *"primitive metrics and patterns"*, we mean metrics and patterns which can be directly determined by issuing EPL-based pattern specification and template to the *Event Preprocessing*. There are metrics and patterns as well as other high level information associated with interactions which need particular analysis besides EPL-based pattern specification and template.

To explain how primitive patterns/metrics can be detected, let us consider that a client would like to determine a `NumHumanServiceCalls` (how many time a human calls a service). Listings 1.3 and 1.4 describe one example of the corresponding pattern expression and the resulting metric. In Listing 1.3, an EPL-based request is used to select all humans (determined by `userID`) and services they use (determined by `serviceEndpoint`) from *interaction event* in the last 30 minutes and then to return only humans who call a service more than 10 times. The corresponding result is given in Listing 1.4. Although it is a simple example, various parameters can be customized to detect the metric for different purposes.

```
<pattern name="NumHumanServiceCalls" resulthandler="voim.NumberServiceCallHandler">
 select userID, count(serviceEndpoint) as NumHumanServiceCall, serviceEndpoint from
       InteractionEvent.win:time(30 min) group by serviceEndpoint, userID
       having count(serviceEndpoint) > 10
</pattern>
```

Listing 1.3. Example of query for determining `NumHumanServiceCall` when a human uses a service more than 10 times in last 30 minutes

```
    <userID>
        http://www.vitalab.tuwien.ac.at/projects/incontext/TEST_LINH1#Linh
    </userID>
    <serviceEndpoint>http://madrid.vitalab.tuwien.ac.at:8080/axis2/
            services/activityservice</serviceEndpoint>
    <NumHumanServiceCall>13</NumHumanServiceCall>
</NumHumanServiceCalls>
```

Listing 1.4. Example of a $NumHumanServiceCalls$ result

7.2 Complex Interaction Analysis

The use of *Metric/Pattern Detection* helps to identify several patterns and metrics at runtime. However, many high-level information cannot be obtained from this component, for example, social network of people or a network of service interactions. The *Interaction Analysis* is used to analyze interaction information which cannot be determined by using pattern specifications or templates directly. The *Interaction Analysis* utilizes information provided by *Metric/Pattern Detection* and provides high-level information. A result handler in *Interaction Analysis* component will receive events which are the output of *Metric/Pattern Detection* handlers and will analyze these events to provide interaction metrics and patterns. Table 4 presents some high level analyses provided.

7.3 Extensibility and Customization of Interaction Analysis

Extensibility and customization are two major requirements for VOIA as different analyses are required for different purposes and because in dynamic collaboration environments various types of events can be used for detecting metrics and patterns. There are two ways to provide new interaction mining analysis: by providing new pattern specification and template and by developing a new plugin. In the first case, the client of VOIA can utilize existing result handlers and focus on writing the specification and template that detect their interesting metrics and patterns. This way is particular useful when dealing with service-specific events. In this case, a new pattern specification and

Table 4. Example of high level interaction analyses

Name	Description
ServiceInteractionNetwork	Describe interaction network between services. We use $ServiceInteraction(s_i, s_j)$ to build the network of service interactions in which the node is a service, the edge is the interaction between two services.
HumanInteractionNetwork	Describe the social network between human. We use $NumCallees$, $NumCallers$ and $NumInteractions$ in human-to-human metrics to build the network. We can also further identify which is a typical service used for the interaction between two persons.
HumanInAllActivity	Describe a network mapping activities to humans. This can be used to detect different types of activities typically performed by a human. Detailed information can be, e.g., the type of activities that a person typically performs.

template can be submitted to VOIA service during runtime. VOIA is acting like a metrics and patterns processing engine. In the latter case, a new plugin can be developed and straightforwardly integrated into VOIA, by (1) providing a pattern specification or template, and (2) a result handler which implements a generic handler interface provided by VOIA. Based on that, a new entry for *Pattern Specification and Template* can be created and VOIA will execute the new plugin.

7.4 Managing and Providing Resulting Metrics and Patterns

To access analysis results during runtime, any client can query or subscribe the results hold in a result handler based on pattern name by invoking Web services operations. Results can be also sent to the client based on notification mechanism. Thus, clients can easily obtain mining results from our framework to perform runtime adaptation.

Given a pattern or metric detected at a specific time, typically such a metric and pattern will be delivered to the corresponding client who initiates the pattern specification/template. However, we also manage such pattern and metric for later use. The information is stored in *Historical Results*. As in online analysis, a result handler will have a new result when events meet pattern specification or template. First, each result handler will keep only n latest results; n is pre-defined. Instead of providing a big XML document including analysis results, we provide a collection of small documents which are also used as events to notify interested clients. Each result is associated with a timestamp. When a result is determined based on space (e.g., number of events in a window) or time (time period associated with a window), we associate the result with the time window to identify the valid period of the result. We do not merge results provided by a handler into a big document. Instead, the list of results will be stored into an XML database. Management features can be used to remove unneeded results. As many results are produced at runtime, aggregating them into a more meaningful information is challenging. Currently, we consider them only historical data and let the client analyze the historical results for its own purpose. For accessing historical data, clients just specify a time period together with an XQuery-based request and a pattern name.

8 Experiments

In this section, we present some experiments to illustrate how VOIA can provide useful metrics and patterns for understanding and adapting collaboration environments. Our experiments are based on the inContext testbed including various collaboration services such as `Document Repository Service`, `Notification Service`, and `Activity Service`. All of them are SOAP-based services. We gathered events from collaborations executed in the inContext testbed and analyzed the collected events. To test VOIA's functionalities and performance, we performed two experiments.

In the first experiment, we analyzed events produced by the inContext testbed. These events were generated from the usage of different services during the integration and testing of the inContext system. Thus, patterns might or might not reflect some use cases[3]. Most events in this experiment are *interaction events*. In the second experiment, we randomly generated a collaboration based on a tree of activities. The tree of activities is generated based on (1) a list of 11 users, a list of 3 roles, 5 levels of a tree, average number of activities per level is 5, a list of more than 20 real services, a list of possible *human activity handling strategies* including *delegate, split, perform, reject, assign*.

Figure 3 presents a snapshot of the service interaction network from the first experiment(for brevity, we removed the hosting URI from the graph). This network evolves during the runtime analysis. The information is provided in XML and exported into a dot format visualized by GraphViz[4]. From that information, we can determine how services were used and then devised service selection strategies. We then examined how humans use services. Listing 1.5 presents the result of human-to-service calls for $UserTest$ which is used to access data from two services. Of more than 20 Web services in the testbed, only `TeamService` and `DocumentService` are interacted with `UserTest`. It is due to the fact that these services were used by the user to acquire the information about other people and to store and search relevant documents. Such activities still involve heavily humans. The information obtained from this type of analysis can be useful, e.g., for determining typical services used by team members to improve runtime service provisioning strategies.

In the second experiment, we examined the proxy pattern. Listing 1.6 presents the result of proxy indicator pattern. Out of 971 activity events, 38 proxy cases were found. For each person involved, we can determine who typically acts as a proxy. This kind of analysis can be useful, for example, for selecting team members and determining trust in collaborative networks of enterprises.

To test the performance of the engine, we wrote a test client that reads events from local file systems and invokes the VOIA service. The test was performed in an Intel Centrino Duo Core 1.83 GHz, 2GB RAM, Windows XP notebook with 10 patterns. Overall, VOIA can handle a high volume of events in a time-responsive manner: it took 9035 ms to process 6775 interaction events (in the first experiment) and 2218 ms for 971 activity events (in the second experiment).

[3] The inContext system includes real services and a research system for the EU project inContext
[4] http://www.graphviz.org/

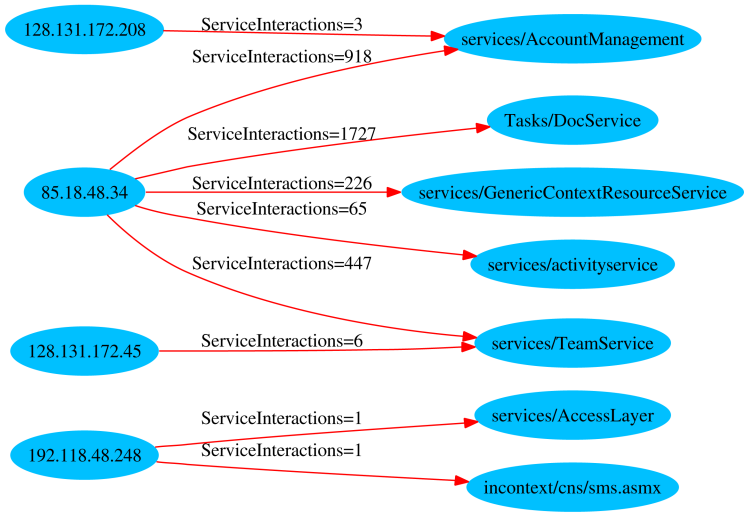

Fig. 3. A snapshot of service interaction network detected

```
<NumHumanServiceCalls>
  <entry>
    <userID>UserTest</userID>
      <serviceEndpoint>
        http://oslo.vitalab.tuwien.ac.at:8080/axis2/services/TeamService
      </serviceEndpoint>
      <NumHumanServiceCall>57</NumHumanServiceCall>
  </entry>
  <entry>
     <userID>UserTest</userID>
     <serviceEndpoint>
         http://srvweb02.softeco.it/cgi-bin/SOAP.cgi/Eadt/Tasks/DocService
      </serviceEndpoint>
     <NumHumanServiceCall>256</NumHumanServiceCall>
   </entry>
</NumHumanServiceCalls>
```

Listing 1.5. Example of human-to-service $NumHumanServiceCalls$ metrics

```
<ProxyIndicator total="38" at="13.04.2008 15:00:21">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Martin" value="7">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Schahram" value="11">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Vasko" value="2">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Florian" value="2">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Atif" value="3">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Shariq" value="3">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Lukasz" value="3">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Christoph" value="3">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Linh" value="2">
 <ProxyIndicator name="http://.../incontext/TEST_LINH1#Kamran" value="2">
</ProxyIndicator>
```

Listing 1.6. Example (simplified) of proxy-related metrics for people

9 Conclusions

Motivated by the lack of online interaction analysis tools in collaborative work and the need of acquiring insightful information for runtime adaptation of collaborative working environments, we presented VOIA (Vienna Online Interaction Analysis) framework which is capable of detecting and providing metrics and patterns associated with human and service in dynamic collaborations. The main contribution of our work is that we provide a Web service-based system that is capable of performing runtime analysis of interaction patterns and metrics. We have provided a rich view of metrics and patterns associated with interactions that spans different levels, including individual, group, and collaboration views, and that characterizes different types of interactions, including service-to-service, human-to-service, and human-to-human. Our system is flexible and customizable, allowing for the inclusion of new analysis and supports client-customized mining. If clients of VOIA understand the structure of their events, they can also define pattern specifications and templates for determining patterns and metrics at runtime. By supporting online analysis, VOIA can provide useful information for runtime adaptation in collaborative working environments.

Various future steps have to be done to fully support online interaction analysis of collaborative processes in SOA-based environments. We are working on determining trust in collaborations in networks of enterprises based on our proposed metrics and patterns. While online analysis allows the client to freely define the analysis they want, the challenge is how to manage the result of different analyses. Currently, we have just stored the result, thus advanced techniques for managing mining results will be studied. Our future work foresees to provide further testing of the systems and support more types of events. We will enhance the pattern specification and template catalog by incorporating new patterns. Another major effort is to perform runtime adaptation based on patterns that motivates this work, but has not been addressed in this paper.

Acknowledgements. We thank Christoph Dorn, Robert Gombotz and Daniel Schall for fruitful discussion on the interaction mining and their assistance in the implementation of VOIA framework. We thank anonymous reviewers for their fruitful and extensive reviews. This research is partially supported by the European Union through the FP6 project inContext and the FP7 project COIN.

References

1. Lipnack, J., Stamps, J.: Virtual teams: reaching across space, time, and organizations with technology. John Wiley & Sons, Inc., New York (1997)
2. Truong, H.L., Dustdar, S., Baggio, D., Corlosquet, S., Dorn, C., Giuliani, G., Gombotz, R., Hong, Y., Kendal, P., Melchiorre, C., Moretzky, S., Peray, S., Polleres, A., Reiff-Marganiec, S., Schall, D., Stringa, S., Tilly, M., Yu, H.: Incontext: A pervasive and collaborative working environment for emerging team forms. In: SAINT, pp. 118–125. IEEE Computer Society, Los Alamitos (2008)
3. ECOSPACE: eProfessionals Collaboration Space (last access April 14, 2008), `http://www.ip-ecospace.org/`

4. COIN: Enterprise Collaboration and Interoperability (last access Novomber 28, 2008), http://www.coin-ip.eu/
5. Barros, A.P., Dumas, M., ter Hofstede, A.H.M.: Service interaction patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
6. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Mans, R.S., de Medeiros, A.K.A., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W.E., Weijters, A.J.M.M.: ProM 4.0: Comprehensive support for *real* process analysis. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)
7. Zdun, U., Hentrich, C., van der Aalst, W.M.P.: A survey of patterns for service-oriented architectures. IJIPT 1(3), 132–143 (2006)
8. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
9. Gombotz, R., Dustdar, S.: On web services workflow mining. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 216–228. Springer, Heidelberg (2006)
10. Dustdar, S., Hoffmann, T.: Interaction pattern detection in process oriented information systems. Data Knowl. Eng. 62(1), 138–155 (2007)
11. Dustdar, S., Hoffmann, T., van der Aalst, W.M.P.: Mining of ad-hoc business processes with teamlog. Data Knowl. Eng. 55(2), 129–158 (2005)
12. Truong, H.L., Dustdar, S., Fahringer, T.: Performance metrics and ontologies for grid workflows. Future Generation Comp. Syst. 23(6), 760–772 (2007)
13. Zhang, P., Serban, N.: Discovery, visualization and performance analysis of enterprise workflow. Comput. Stat. Data Anal. 51(5), 2670–2687 (2007)
14. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
15. van der Aalst, W.M.P.: Exploring the cscw spectrum using process mining. Advanced Engineering Informatics 21(2), 191–199 (2007)
16. Cozzi, A., Farrell, S., Lau, T., Smith, B.A., Drews, C., Lin, J., Stachel, B., Moran, T.P.: Activity management as a web service. IBM Syst. J. 45(4), 695–712 (2006)
17. Dustdar, S., Gombotz, R.: Discovering web service workflows using web services interaction mining. International Journal of Business Process Integration and Management 1(4), 256–266 (2006)
18. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
19. EsperTech - Esper: Event Stream and Complex Event Processing (last access April 14, 2008), http://esper.codehaus.org
20. The inContext project (last access April 12, 2008), http://www.in-context.eu
21. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
22. Dustdar, S.: Caramba - a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. Distributed and Parallel Databases 15(1), 45–66 (2004)
23. Java Specification Request 224: Java API for XML-Based Web Services (JAX-WS) (last access April 14, 2008), http://www.jcp.org/en/jsr/detail?id=224
24. EPL Reference: Clauses (last access April 15, 2008), http://esper.codehaus.org/esper-2.0.0/doc/reference/en/html/epl_clauses.html