Event Semantics for Service Composition in Pervasive Computing

Sourish Dasgupta, Satish Bhat, Yugyung Lee

School of Computing and Engineering
University of Missouri – Kansas City
5100 Rockhill Rd. Kansas City, Missouri 64110, USA
{sdwb7, ssbmvd, leeyu}@umkc.edu

Abstract

In a pervasive environment, prior knowledge of tasks is not always possible owing to the characteristic uncertainty of the tasks. Moreover, we may not be able to define any task-template at all that can be modeled as a goal for a service composition process. In this paper, we have modeled a service composition process as an event-handling process in the domain of pervasive computing. We have also shown how event semantics (i.e. the initial and the final states) can define the way events should be handled by a particular pervasive system. The objective is to find the best event-target. This can only be guaranteed if the contexts of the end services producing such event-targets are compatible with the desired event-target contexts. This requires a source service whose context is compatible with that of the event to be handled. Thus, we define Context-Aware Ontology Framework for Events and Services, called CAOFES, which lies in the semantic formalization of the contextual effects of environmental dynamics that can be brought about by services and events. In addition, we present formal definitions of the three different compatibilities: service-to-service compatibility, service-to-event compatibility, and service-to-event target compatibility. The notion of these compatibilities form the essential basis for logically integrating contexts of services participating in a composition process into event-specific contextual boundaries called a *Situation Boundary* (SB). We have evaluated the performance of compatibility computation of the proposed SB model based on randomly generated service network and contextual information.

Introduction

Pervasive computing is an effort towards achieving the vision of an invisible computing fabric around us (Weiser 1991) (Satyanarayanan 2001). Managing and interacting with resources distributed across devices presents the problem of how to seamlessly integrate resources into our daily activities. For effective management of such systems, what we need is a way to capture the dynamic environment around us and provide adaptive systems that proactively tailor themselves depending upon specific contexts such as location, time, and other parameters. However, there are no design guidelines or methods for formalizing contexts to support the adaptive pervasive computing systems.

Current efforts in the research community demonstrate the ability of managing pervasive systems through diverse adaptive mechanisms, such as location awareness (Want et al. 1995) (Griswold et al. 2004) (LaMarca et al. 2005) ambient awareness (Matthews et al. 2004), and activity awareness (Ward et al. 2006) (Logan 2007). Service-Oriented Architecture (SOA) is particularly appropriate for loosely coupling resources distributed over networks by exposing and providing access to such resources available as services. In general, SOA has been used in Peer-to-Peer frameworks for service composition (Gu, Nahrstedt, and Hu 2004), in the web for semantic web service composition (Bianchini, Antonellis, and Melchiori 2005), for designing specific toolkits (Ponnekanti and Fox 2002), and composition of multimedia services (Nahrstedt and Balke 2004). SOA has been successfully employed within pervasive computing systems too (Kalasapur, Kumar, and Shirazi 2007). However, there are many critical issues pertinent to service discovery and composition for pervasive computing (Satyanarayanan 2001) that have not been satisfactorily addressed. There is a need for context-aware and adaptive models that can handle non task based composition of services in very non-deterministic pervasive computing systems. In order to realize this we need to enhance the current SOA framework with an intelligent layer that can perceive and interpret different contextual situations within such systems.

In this paper we present a new event-driven model for pervasive systems in which event semantics play an important role in the discovery and composition of services. We have proposed a contextual framework called CAOFES (Context Aware Ontology framework for Events and Services) that provides the intelligent layer needed for recognizing event semantics and triggering necessary

reactions. It also provides semantic reasoning for discovering potential services that can handle typical events that may occur in a particular pervasive system. We have shown how contexts of different services can be logically integrated into a *situation* that is specific to a particular event. We have proposed a novel technique for modeling such situations into contextual boundaries called a *Situation Boundary* (SB) and have shown how the generation of such boundaries under a simulated environment can improve event-handling in pervasive systems significantly. We have proposed an algorithm called the *SBGenerator* to create SBs for given test events. We have evaluated the compatibility issues that are involved in the logical integration of contexts into SBs.

Event Semantics in Pervasive Computing

A typical characteristic of event-based models is that there is no strict notion of a task or goal that can be explicitly framed as a graph of abstract services by the user (Masuoka, Parsia, and Labrou 2003) (Perttunen, Jurmu, and Riekki 2007) (Sousa et al. 2006). This is especially applicable for a non-deterministic pervasive system where it is not always possible to define such an 'abstract task-template'. This is because a task-template requires prior understanding of the specific input and output requirements of each of the abstract services that constitutes the task as a whole. In a dynamic pervasive environment such prior understanding is not always possible owing to the characteristic uncertainty of the tasks themselves. Moreover, we may not even be able to define any goal within pervasive systems. Hence, in the domain of pervasive computing, task-based modeling can fail.

Instead we have events whose occurrences are often non-deterministic and that happen to be characterized by changes in the states of the environmental non-system entities. Thus, an event can also be defined in terms of the current state of the environment and the final state after the change is over. In a pervasive system, an event becomes meaningful if this change can be perceived by the system and also handled using a set of system services. This handling has to be done through a composition process.

The objective of the composition process for any event in an event-based model is not to satisfy a well defined task-goal. Instead, the task is to find *reachability* between a source service and an end service such that the end service will be able to produce a result that best handles the initial event. In other words, the end service result is the *target event* of the initial event while the source service is the initial point of execution of the entire composition process. It should be noted that there may be many intermediate services that are required in order to trace out such reachability. These intermediate services may not have any direct relation with the event. Neither may they have any connection with the target of the event. This is yet another difference from a task-based model where all the intermediate services are also part of the explicit task and hence can be mapped with well defined sub tasks. The relation between an event and its target event may not be one to one. A particular event may have more than one target event. It may also happen that such target events may be the results of different end services instead of just one. Hence, the composition process should be such that it completely handles the event for which it is triggered.

As events are associated with event states (initial and current) that constitute their contexts, we can formally represent an event as:

$e_i = (\psi^{Ei}, \psi^{Ef}, \tau_{jf})$

where, ψ^{Ei}: Initial state of the environment E

ψ^{Ef}: Final state of the environment E

τ_{jf}: Change from the initial state to the final state

Thus, events are not only identified by the activity but also by the initial and the final states of the environments to which the activity can be associated. These states form the context of the events. For an example, we might consider a smart home where a person's favorite music is played in the bedroom when she wakes up from sleep in the morning. If the person leaves the room then she is shown all her pending tasks in a display installed in the room where she enters. In this system '*person X waking up in the morning*' is an event. '*Person X leaving the room*' is another event. The system interprets the semantics of the initial and final states of these activities and reacts to it in a certain manner. If person X wakes up and leaves the room then the final state of the activity of *waking up* afterward becomes the initial state of the second activity of *leaving room*. However, if other events occur between the occurrences of these two events such that the initial state of the second activity is different (for an example person X wakes up, goes for a walk, comes back to her bedroom after the walk and then leaves the room) then we actually have a second event version of the activity. In other words, the occurrence of the first event influences the way the second event will be interpreted by the system. In our event model, we implicitly consider the initial state of the event as an effect of all previously occurred events. In this manner we rule out the need for considering all events that have influence over a particular event. Thus, we treat a mutually dependent (and possibly complex) event network into an independent discrete event-context model.

The intelligent service composer registers event context semantics within the scope of a system. This context information for any particular event is used for a suitable composition process to be initiated. As the activity states can change how an event is interpreted by a system, the event-target required for that event also may change from what may apparently seem to be the best handling one. In this paper we do not concentrate on how the intelligent service composer captures the state conditions. We assume such technological availability in a modern pervasive system. We also do not discuss whether such intelligence is distributed or centralized. Instead, we focus on a new

composition approach that can be used for high-performance event handling in a situation-aware event-driven system.

We now define *event* w.r.t pervasive systems as follows:

Definition 1: An *event* (denoted by e) is an atomic process that when executed until completion over a continuous time period brings a *notable* change τ_{jf} from an initial state ψ^{Ei} to a final state ψ^{Ef} of an object or a group of objects outside the *system boundary*. ■

In the scope of our model, events are only those activities that are executed by agents affecting entities outside the closed set D of pervasive service hosting devices (termed as *system boundary* in the definition). The implication of the term *notable* is very significant in the sense that we are only to be concerned about changes in states of non-system entities that have a direct relationship with the execution of the services (either as a cause or an effect). It should also be noted that the definition is not framed in the perspective of a process, but rather in the perspective of a change in the activity state.

The exclusion of the nature of agents from the definition is meant to include both the D set as well the entities $\notin$ D. Thus, any web device executing a service can possibly trigger an event that generates notable change in the states of some entities outside the boundary D. In such cases, we can say that the service result becomes the event. In the next section, we propose an ontology framework that formalizes the semantic relations between events and their contexts along with the services that are associated to the events.

Context-Aware Ontology Framework for Events and Services

Several languages have been proposed to formalize the idea of semantic pervasive services, the most prominent being OWL-S (Bastida, Nieto, and Tola 2008). However, these languages cannot support a pervasive system that is highly dynamic and the services within are context sensitive. It was in a similar direction of thought that a new language called AMIGO-S for ubiquitous services was proposed in (Canal, Poizat, and Sala¨un 2006). AMIGO-S is an extension language to OWL-S (Röcker et al. 2005). However, AMIGO-S is defined from a '*capability*' (i.e. functionality) perspective. A match between two services is done based on the semantic distance of the concepts that describe such capabilities. For causality deduction, a system must have a priori knowledge of the capabilities of a service that is needed from some other services.

Contrary to this perspective, CAOFES maintains the conventional idea of matching based on pre-condition constraint satisfaction. It assumes no a priori conceptualization of a required process by any of the services within a pervasive service system. But at the same time the uniqueness of CAOFES lies in the semantic formalization of the contextual effects of system dynamics that can be brought about by services and events. Moreover, CAOFES is event-based and is a semantic network of three ontologies: (i) *onto_service*, (ii) *onto_context*, and (iii) *onto_event*. Due to its modular design, it helps us to reason about events and their associations with services as well as contexts.

Contextual Elements: As mentioned, we have three ontologies that are interrelated with each other. The service ontology (*onto_service*) is an extension of the standard OWL-S and hence, all the instances of the services so defined will have three semantic layers: (i) service profile, (ii) service process, and (iii) service grounding. However, we also define the contexts of each service by establishing relations with the six basic contextual elements of the context ontology *Onto_Context*: (i) Spatial Context (*Sp_Con*), (ii) Temporal Context (*Temp_Con*), (iii) Actor Context (*Ac_Con*), (iv) Object Context (*Ob_Con*), (v) Background Context (*Bg_Con*), and (vi) Information Context (*Info_Con*).

Service Context: Contextual information provides the '*working condition*' under which a pervasive service can execute. This may constitute: (i) the *spatial context* (in terms of geography, specific address, relative location with respect to a specific address) where a service can operate, (ii) the *temporal context* (in terms of day, year, month, morning, afternoon, evening and night) when a service can operate, (iii) the *actor context* that specifies the device executing a service (i.e. the device profile and capabilities) and also the entity who receives the service (i.e. the user profile), (iv) the *object context* that specifies the entity getting affected by the execution of a service, (v) the *background context* that specifies state information of entities around the place of execution of a service, and (vi) the *information context* (aural, textual or visual) that specifies the kind of interaction needed during execution of a service. Collectively we term the six dimensions of working condition as *s_Con* (meaning Service Context).

Event Context: Apart from the services, the events that are conceptualized and instantiated within *Onto_Event* are also contextually related with *Onto_Context*. As events are associated with two contextual states (initial and final), we formalize their semantics separately. Thus, we have initial state context semantics and final state context semantics for an event. In the event perspective the *Ac_Con* comprises the set of actors who executes the events (may be human, device, or any other machine outside the system boundary) as well as those who are recipients of such an event (the type of device and service). In the same way, all the other contextual elements retain their meaning but are seen in the perspective of initial and final state information of entities that can be associated with the occurrence of the event. Certain contextual elements will be pertinent with the initial state context semantics (like *Ac_Con*, *Ob_Con*) while others may be pertinent to both the state semantics.

Hence, every event has a semantic wrapper comprising all such contextual information that we collectively term as *e_Con* (meaning Event Context).

In certain ways CAOFES can be seen as a modularized extension of the SOUPA model (Chen et al. 2004). However, a major difference between the two is that the three-tier modularity of CAOFES helps us to understand and reason the effect of events on services. A service may be considered as a follow-up of an event if the *e_Con* is semantically compatible with *s_Con* of the service and if the service can trigger a target process for the event. This service acts as a source node in a composition process that should lead to the execution of another service(s) which triggers an event(s) that is compatible with the context of the desired target event. Of course, no such composition process is required if the service can itself trigger the target event.

Semantic Similarity of Services

Semantic similarity measures have been studied extensively for the last two decades. We can categorize the most of the previous researches into two classes: (i) similarity based on subsumption path lengths between two comparable concepts (Rada et al. 1989) and (ii) similarity based on information content of the parent of two comparable concepts (Resnik 1999). Some works have formulated similarity measures by adopting both the perspectives (Lin 1998) (Hirst and StOnge 1998).

A fairly recent work proposed in (Hau, Lee, and Darlington 2005) proposes a triple based similarity measure where a ratio of the common RDF triples between the two comparable concepts and the overall triples (computed by a union operation over each of the concept's triples) is calculated. A threshold value is assigned based on which the similarity between two concepts is determined to be of required closeness. One of the most important contributions of this work is the formulation of semantic similarity between web services. However, the similarity measurement is limited to the functional commonalities between two web services. Similarity was calculated for each of the four service profile dimensions of the OWL-S specification: (i) Input, (ii) Output, (iii) Pre-conditions, and (iv) Result. There was not any attempt to formulate a similarity based on the contextual information of the services.

As mentioned earlier, contextual information provides the working condition under which a service can execute. This may constitute: (i) the *spatial context* where a service can operate, (ii) the *temporal context* when a service can operate, (iii) the *actor context* that specifies the device executing a service and also the entity who receives the service, (iv) the *object context* that specifies the entity getting affected by the execution of a service, (v) the *background context* that specifies state information of entities around the place of execution of a service, and (vi) the *information context* that specifies the kind of interaction needed during execution of a service.

It is notable that for any service composition mechanism similarity metrics provide the key platform for grouping of functionally similar services and mapping those services to requested tasks. This goes the same when the system is non-deterministically event-driven. We may have a situation where there are several candidate services having similar functionalities but dissimilar contexts. Hence, an event taking place in a pervasive system can be mapped to any of such candidate services. The selection strategies in most service selection models that have been proposed in the past were goal-centric. This means that the candidate services were selected in such a manner so that the overall composition process satisfying the task incurs minimum cost. In many cases, the cost is evaluated based on non-functional QoS parameters like bandwidth, memory, CPU, etc. However, as mentioned earlier, such selection strategies are task-based and hence, cannot be used for event-driven models.

The objective of a composition process in such models, as discussed earlier, is to find reachability between source service triggered by an event and an end service that triggers the event target. In order to do so we must ensure that an event is contextually compatible with the set of services that is used to handle it. If the contextual information of an event does not match with that of a candidate service then the selection of that service will not give the optimum composition. This is because it may result in a target event that does not adequately handle the initial event.

For an overall contextual compatibility with a set of services participating in a composition we need to make sure that the services themselves are functionally compatible. This means that in order that one service should trigger another service as its effect the result of the former should satisfy the pre-condition of the latter and the output of the former should satisfy the input of the latter. Hence, we introduce the metric *Dependency Similarity* (Sim^{Dep}) so as to capture such service to service compatibility.

Sometimes a composition process does not involve any causality (direct or indirect) between two services. Such services are independent of each other and it does not matter which one is executed first as neither the validity nor the result of the composition is affected. However, it is necessary to ensure that such independent services are contextually compatible as well. That is, the contexts of two such services should not contradict each other. This check is also important for mutually dependent services having a causal relation between them. Hence, we introduce the metric of *Contextual Similarity* (Sim^{Con}) for capturing such service to service compatibility. Such a metric is also important for ensuring that an event is compatible with the source service that it triggers and also

for ensuring that the end service is compatible with the required event target for that initial event.

As discussed earlier, for grouping functionally similar candidate services, we must introduce the metric of *Functional Similarity* (Sim^{Fun}). This is important for performance optimization of the proposed SB model as the clustering helps to reduce the search space when an event occurs and is required to be handled.

Situation Boundary

As mentioned previously, a service composition process can be modeled as an event-handling process in the domain of pervasive computing. We have also shown how event semantics can define the way events should be handled by a particular pervasive system. The objective should always be to find the best event-target. This can only be guaranteed if the contexts of the end services producing such event-targets are compatible with the desired event-target contexts. This requires a source service whose context is compatible with that of the event to be handled (we call it *s2e compatibility*). At the same time we also need an end service whose context is compatible with that of the required target event (we call it *s2e^h compatibility*). As we trace along any path over a given service network starting from the source service till the end service we must consistently check for compatibility among the intermediate services that are causally dependent (we call it *s2s compatibility*). In other words, we actually trace out a set of services whose contexts are logically aggregated over a service network to form a global context that is conducive to the event and produces the desired target event. We term this global context as a '*situation*'. We can imagine such a trace as the diameter of an abstract boundary of logically aggregated contexts. We term this boundary as a 'Situation Boundary' (*SB*). As a particular situation is the effect of an event that occurs in a pervasive system a particular SB is dedicated for a particular *type* of event (i.e. an event concept defined within CAOFES). Every event concept having a target in a particular pervasive system has a corresponding SB.

Service Compatibility in Service Composition

We consider three types of compatibility issues that can be observed within a pervasive system: (i) *service-to-service compatibility (s2s compatibility)*, (ii) *service-to-event compatibility (s2e compatibility)*, and (iii) *service-to-event target compatibility (s2e^h compatibility)*. For any service composition process to be initiated in an event-driven pervasive environment we need to ensure that the three different types of compatibility are valid.

Service-to-service compatibility is the most widely studied issue in service composition research. This measure checks whether two services have both dependency similarity and contextual similarity for a causality to be established. However, most of the works have been based only on the semantic similarity of IO between two services without considering whether the contexts of these services are similar enough for a dependency to exist in the real-world. In this paper, we consider the required similarity of service contexts as we formally define *s2s compatibility* as follows:

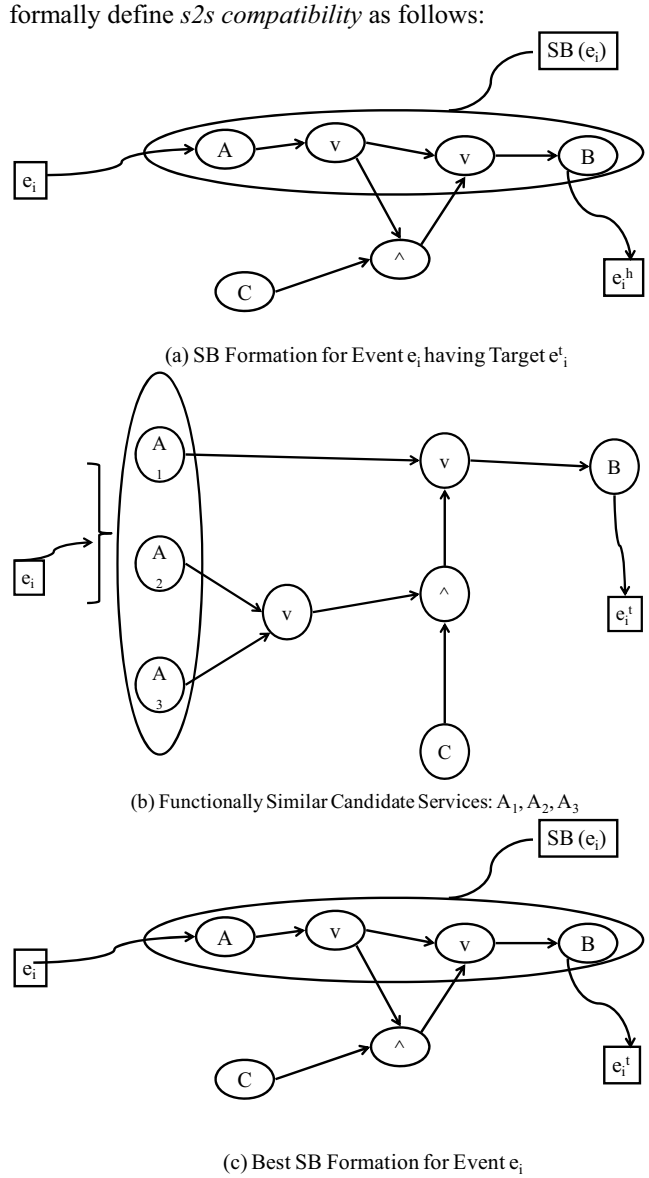


Figure 1: Examples of Situation Boundary

Definition 2: Given an environment E_k and a service network SN_k a service s_i can be said to have *s2s compatibility* with another service s_j if and only if $Sim^{Con}(s_i, s_j) \leq \lambda^{Con}$ and there exists a dependency relation R: $R(s_i, C_{ij}) = s_j$ and $Sim^{Con}(s_i, s_j) \leq \lambda^{Dep}$ and there exists at least one model Δ_{ij} of the set O^{si}: $\Delta_{ij} \models C_{ij}$. ■

Δ_{ij} is a semantic interpretation which is a model of the set of post-condition possibilities of s_i (O^{si}) under the

'working condition' s_i_Con. Two types of pre-defined thresholds have been introduced in the definitions: (i) Contextual Similarity Threshold (λ^{Con}) and (ii) Dependency Similarity Threshold (λ^{Dep}). These thresholds can take any custom value in the range (0,1).

On the other hand, *service-to-event compatibility* has not been studied much in the community. However, we argue that for any effective handling of an event we need to understand whether the event context is in conflict with the context of the service that can initiate a handling process for the event. In other words, the s_Con for a particular service should be similar enough to the e_Con of the event that it is supposed to be handled. We formally define *s2e compatibility* as follows:

Definition 3: Given an environment E_k and a service network SN_k, a service s_i can be said to have *s2e compatibility* with an event e_j if and only if there exists a definition $e_j \equiv triggers \cdot s_i$ within CAOFES and $Sim^{Con}(s_i, e_j) \leq \lambda^{Con}$ provided there exists an end service s_p such that s_p has $s2e^h$ *compatibility* with the target event of e_j and $s_p \leftarrow s_i$. ■

Here we see that a complete comprehension of the definition involves an understanding of $s2e^h$ *compatibility*. Before selection of an end service that might bring about the event target we need to make sure that any such service having a fixed s_Con can actually bring about that change which the target event should bring. In other words, the s_Con of the end service s_p in the above definition should be similar enough with the e_Con of the target event e^h_j. Hence, $s2e^h$ *compatibility* ensures this in the process of handling an event. It also needs to be ascertained that there is reachability from the source service s_i to the end service s_p (denoted by $s_p \leftarrow s_i$). We now formally define $s2e^h$ *compatibility* as follows:

Definition 4: Given an environment E_k and a service network SN_k, a service s_i can be said to have $s2e^h$ *compatibility* with an event target e^h_j if and only if there exists a definition $s_i \equiv triggers \cdot e^h_j$ within CAOFES and $Sim^{Con}(s_i, e^h_j) \leq \lambda^{Con}$. ■

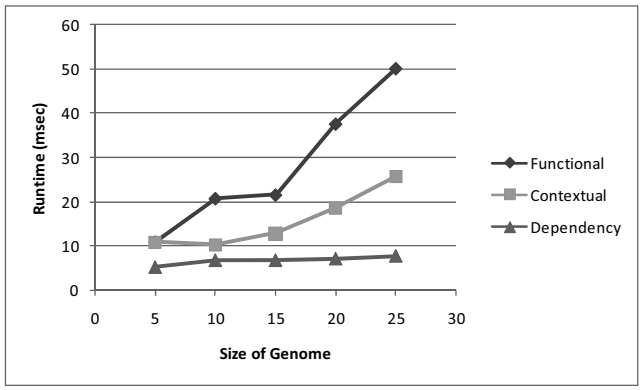
Algorithm SBGenerator

We generate a test set of event types that characterizes a particular type of pervasive system. SBs are created automatically over a given service network instance as follows:

1. Search the service nodes that are end services (must have has a triggers relation with the target event of the test event).
2. If no result is found in step 1 then no solution exist.
3. Else select the end node that has the best average Contextual Similarity with the event target.
4. Search the service nodes that are source services (must have a triggeredBy relation with the test event) and has reachability to the selected end node in step 2
5. If no result is found in step 4 then step 3 is repeated for the next best end node.
6. If no result is found in step 5 then no solution exist.
7. Else select the source node that has the best Contextual Similarity with the e_Con test event.
8. Do a forward tracing till the end node is reached starting from the selected candidate in such a manner that the next hop should be the service node that also has reachability to the end node and has the best s2s compatibility between them.
9. If the end node is not reached then no solution is there for the chosen end service. Repeat step 3 for the next best end service.
10. If there is no result after searching all the end services then there is no solution target to the test event.
11. Else a new SB is formed for the event by including all the services along the traced path.
12. The SB-index is updated with the SBs abstract and real along with the test event that was handled.

Evaluation

The experimental platform was a machine with CPU cycle of 1.4 GHz and RAM of 2 GB. The development platform was NetBeans IDE 6.0.1 with Java as the coding language. The performance is computed in terms of the average execution time (in msec) for each of the three different types of similarity measurement (functional, contextual, dependency) for a random collection of service node pairs in a chosen network. We randomly generate a collection of such service networks using a *ServiceNet Generator*. The *ServiceNet Generator* also randomly assigns the functional and contextual information to each of the services in the generated networks. We term such information (functional and contextual as *genome*). In order to do that we have developed a random *Functional Genome Factor (GF) Generator* and a random *Contextual Genome Factor Generator*. We repeat the experiment for 10 service networks the sizes (in terms of service nodes) of similarity measurements scale for each of these service networks.



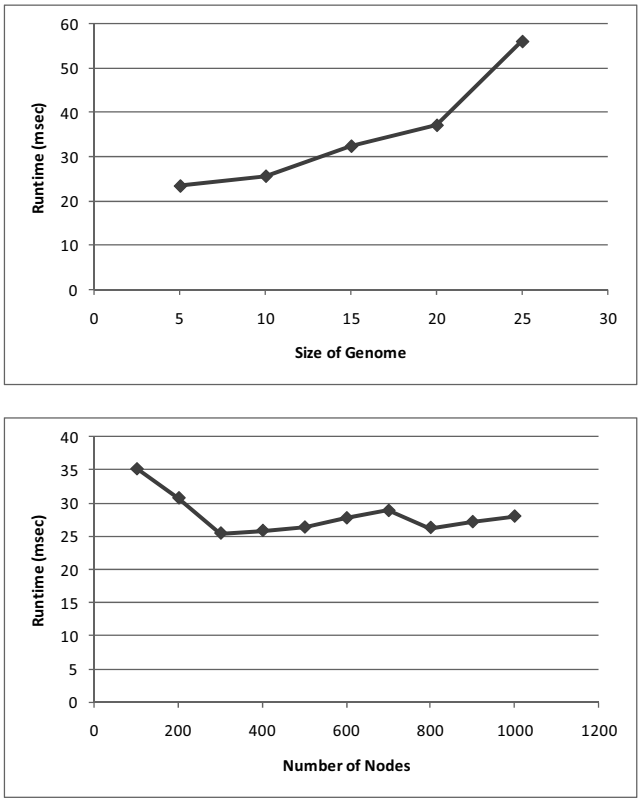


Figure 2: Functional/Contextual/Dependency Similarity (Top) Avg. similarity performance vs. genome size, (Mid) Combined avg. similarity performance vs. genome size, (Bottom) Combined avg. similarity performance vs. service network size

We then record the execution time of these measurements with different collection sizes of the GF sets (i.e. overall cardinality of the mutated GF set, additional GF set and inheritance GF set) of each of the GFs that are contained within a particular gene (functional or contextual). We term this collection size as the *genome size*. In our experiment we took the genome sizes range from 5 – 25 as that can be considered considerably large for a web service domain. We see a very small decrease (execution in msec) in the performance of the individual similarity measures with increase in the genome size in Fig. 2 (Top). This is because of increased number of intersection computations. The Dependency Similarity has the best performance because we only consider a match between the input and the output GFs of two dependent services. Hence, number of intersection computations is least. The best performance case is that of Contextual Similarity as this has the highest number of intersection computation and hence the highest summation overload for the contextual gene. Occasional dips are attributed to the fact that sometimes there are empty GF sets when no intersection computation is required. The combined performance reflects the significant contribution of the Functional and the Contextual measurements which is quite obvious (Fig. 2(Mid)). It can be seen (as expected) that the performance of the combined similarity measure is fairly independent of the service network size (Fig. 2(Bottom)). This is because computation is done for a pair of services by using a fixed number of repetitions for each network size. Hence, we observe high scalability of all the three measurements.

Conclusion

In this paper we have demonstrated how the SB model can help us in achieving high-performance event-handling in a non-deterministic pervasive environment. We have justified why a task-based modeling cannot be suitable for such environments. We have also proposed the CAOFES framework for semantically determining relations between events, services and contexts. We have evaluated the performance time for service-to-event compatibility computation in comparison to service-to-service compatibility (functional and dependency similarity) and have not much deviation between them. We plan to present a comprehensive evaluation of the SBGenerator performance as a future work.

Acknowledgements

This work has been partially supported by National Science Foundation (IIS #0742666).

References

Weiser, M., 1991. "*The Computer for the 21st Century*". Sci. Amer.

Satyanarayanan, M., 2001. "*Pervasive computing: vision and challenges*", IEEE Personal Communications, vol 8(4), p. 10-17.

Want, R., Schilit, B., Adams, N., Gold, R., Petersen, K., Goldberg, D., et al., 1995. "*An overview of the PARCTAB ubiquitous computing experiment*", IEEE Personal Communications, vol 2(6), p. 28-43.

Griswold, W., Shanahan, P., Brown, P., Boyer, S., Ratto, R., Shapiro, M., et al., 2004. "*ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing*", IEEE Computer, vol 37(10), 73-81.

LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., et al, 2005. "*Place Lab: Device Positioning Using Radio Beacons in the Wild*", The International Conference on Pervasive Computing *(*Pervasive 2005*)*, Munich, Germany.

Matthews, T., Dey, A., Mankoff, J., Carter, S., Rattenbury, T., 2004 "*A toolkit for managing user attention in peripheral displays*", The ACM Symposium on User Interface Software and Technology (UIST 2004). Santa Fe, NM, USA.

Ward, J., Lukowicz, P., Trester, G., Starner, T., 2006. "*Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28(10), p. 1553-1567.

Logan, B., Healy, J., Philipose, M., Tapia, E., Intelle, S., 2007. "*A Long-Term Evaluation of Sensing Modalities for Activity Recognition*". LECTURE NOTES IN COMPUTER SCIENCE, SPRINGER-VERLAG. p. 483-500.

Gu, X., Nahrstedt, K., Yu, B., 2004. "*Spidernet: An Integrated Peer-to-Peer Service Composition Framework*". Proc. 13th IEEE Int'l Symp. High Performance Distributed Computing (HPDC '04).

Bianchini, D., Antonellis, V., Melchiori, M., 2005. "*An Ontology-Based Architecture for Service Discovery and Advice System*", Proc. 16th Int'l Workshop Database and Expert Systems Applications, vol (22 – 26), p. 551 – 556.

Ponnekanti, S., Fox., A., 2002. "*Sword': A Developer Toolkit for Building Composite Web Services*", Proc. 11th WWW Conference.

Nahrstedt, K., Balke, W., 2004. "*A Taxonomy for Multimedia Service Composition*", Proc. 12th Ann. ACM Int'l Conf. Multimedia (Multimedia '04).

Kalasapur, S., Kumar, M., Shirazi, B., 2007. "*Dynamic Service Composition in Pervasive Computing*", IEEE Transactions on Parallel and Distributed Systems, vol. 18(7), p. 907-918.

Masuoka, R., Parsia, B., Labrou, Y., 2003. "Task Computing – The Semantic Web Meets Pervasive Computing", Proc. of 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, USA, p. 866-881.

Perttunen, M., Jurmu, M., Riekki, J., 2007. "A QoS Model for Task-Based Service Composition", 4th International Workshop on Managing Ubiquitous Communications and Services (MUCS 2007), Munich, Germany.

Sousa, J., Poladian, V., Garlan, D., Schmerl, B., Shaw, M., 2006. "Task-based Adaptation for Ubiquitous Computing", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Special Issue on Engineering Autonomic Systems, Vol. 36(3), p. 328-340.

Bastida, L., Nieto, F., Tola, R., 2008. "Context-Aware Service Composition: A Methodology and a Case Study", SDSOA'08, Leipzig, Germany.

Canal, C., Poizat, P., Sala¨un, G., 2006. "Synchronizing Behavioural Mismatch in Software Composition", In Proc. of FMOODS'06, vol. 4037 of LNCS, p. 63–77.

Röcker, C., Janse, M., Portolan, N., Streitz, N., 2005. "User Requirements for Intelligent Home Environments: A Scenario-Driven Approach and Empirical Cross-Cultural Study", Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies, vol. 121, p. 111 – 116.

Chen, H., Perich, F., Finin, T., Joshi, A., 2004. "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications", International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, MA, USA.

Rada, R., Mili, H., Bicknell, E., Blettner, M., 1989. "Development and application of a metric on semantic nets", In IEEE Transactions on Systems, Man, and Cybernetics, vol 19.

Resnik, P., 1999. "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language", In Journal of Artificial Intelligence Research, vol 11, p. 95–130.

Lin, D., 1998. "An information-theoretic definition of similarity", Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann (1998), p. 296–304.

Hirst, G., StOnge, D., 1998. "Lexical chains as representation of context for the detection and correction of malaproprisms", WordNet: An electronic lexical database. MIT Press (1998) p. 305–332.

Hau, J., Lee, W., Darlington, J., 2005. "A Semantic Similarity Measure for Semantic Web Services", Web Service Semantics Workshop at WWW (2005), Chiba, Japan.