Modeling and Analysis of Personal Web Applications: A Vision

Marsha Chechik, Jocelyn Simmonds, Shoham Ben-David, Shiva Nejati, Mehrdad Sabetzadeh, and Rick Salay

Shiva and Mehrdad are with Simula Research Lab, Norway. Other authors are with Department of Computer Science, University of Toronto

1 Introduction and Assumptions

In this paper, we attempt to identify our vision of what personal web is. We then provide a challenge problem for such a vision and discuss our assumptions of how other fields of computer science can contribute to executing this vision. We then discuss where the "traditional" software engineering tasks of specification, modeling, monitoring and verification fit into the vision.

What is Web 3.0? We share the vision of the workshop organizers that it is focused on an individual user (well, consumer) and is trying to elicit and execute this person's goals, through preferred information collection devices and with cooperation with trusted individuals. For example, traditional web applications such as commerce and banking offer a particular interaction with the user and his/her data. Data is stored in the database on a particular application (e.g., shopping list or wish list), and the user is being offered a particular workflow that determines the interaction of the user with the system (e.g., on amazon.com, such things include looking for something, doing a price comparison, determining a particular vendor to go with, choosing the type of shipment and the payment method).

Instead, as users, we find those parts of the different applications which are useful to us, and then combine them in ad-hoc ways. For example, when buying electronics, we (a group of Toronto-based academics) first check amazon.com to look at the models and reviews. Amazon.ca has a much smaller product selection, and very likely will not carry the desired product. Instead, we look for the equivalent models on other Canadian retail sites. After comparing prices and shipping options, we make a purchase. In other words, we have an informal workflow for buying electronics online. On a good day (no paper deadlines, no screaming children, no advisors interrupting with urgent requests), we have time to do each step of this process meticulously and end up completely satisfied with the end result. But when pressed for time, we start skipping steps (since nothing is automated), e.g., we will order directly from amazon.com and pay extra shipping and customs charges, or shop our favorite Canadian retailer instead of comparing prices, and sometimes miss out on a sale.

Our vision of personal web is that it allows users to define and manipulate personal workflows, populated by their favorite vendors and information sites. For example, users would download the "web for a Canadian shopper" workflow, that implements the above workflow for shopping for electronics (clearly, it is right about the same no

matter what is being purchased) and then modify it to suit their goals. Other clear workflows are for organizing a dinner and a movie outing (choosing an interesting movie - and checking appropriate sites to determine what is good - a time that works and at a location which is reasonable to get to and that has a restaurant close by that the person executing the workflow would like to visit - coordination with the person's date and the restaurant review sites).

Thus, our definition of a personal web is *Support for identifying and executing (and monitoring and fixing) "mental" orchestrations, or workflows, for incorporating multiple services in order to accomplish complex personal goals*.

The problem is clearly non-trivial and its solution requires a collaboration from various areas of computer science. Here is a very partial listing of the issues:

Data storage. Personal web needs an ability to store the state of each user within their workflow as well as the associated collected data, since it is no longer done at the vendor's site. We think that cloud computing can readily provide a solution for this challenge.

Turning the web into services. The workflow-based vision means the ability to invoke services rather than browse the web. That is, the web should be turned into a collection of such services. We think that the semantic web research community has a lot to offer on this topic.

Services specification. It is essential to have some notion of specification for services, at least to determine whether a particular service can be invoked at a particular step of the workflow but of course to also discover services (don't you want to know that a local computer shop is having a sale and it might be cheaper to go there rather than continuing the on-line shopping experience?). Personal web is not unique in this challenge - it is essential for creating quality web service applications under existing technologies.

Architectural support. Given that users define these personalized workflows, where are they being executed? How are services being "strung" together? Again, there seems to be a lot of success in existing technologies for creating mashups, yahoo pipes, etc.

Modeling and analysis. This is the purpose of this paper - trying to identify challenges in this category, as well as some approaches towards solving them.

Usability and User Experience. This challenge is truly cross-cutting. The proposal would simply be infeasible if users are unable to specify workflows, provide rankings of various sites, etc. While we touch on this subject a little bit later in the paper, it is mostly orthogonal to our proposal, but, of course, essential.

In the rest of this paper, we will describe a challenge example for providing user-controllable workflows (Section 2) and then use it to describe some modeling and analysis challenges we as a community face before the Personal Web idea becomes a reality (Section 3). We also discuss how to begin solving these challenges (Section 4). We conclude in Section 5.

2 Motivating Example: Online Crib Shopping

We begin by proposing a (real-life!) challenge problem for Personal Web.

Consider the following scenario. Our (Canadian) user is six months pregnant and wants to purchase a baby crib. Quality cribs are durable but expensive, and take a while to get once ordered. So, she wants to try to buy a second-hand crib. The easiest way to get one is through a local online classified ads, such as craigslist.org, since she can go to the vendor in person and inspect it before making a decision. Her parents live in the US and frequently travel back and forth, so they can also look for local deals on used cribs, including their local craigslist.org, garage sales, etc. Our user also knows that quality cribs take 6 weeks to arrive when ordered, so she can only keep looking at used cribs for another 1.5 months. If that (soft) deadline passes, the user will have no choice but to go to a retailer that has cribs in stock and buy whatever they have – clearly not a good choice but might be the only option for meeting the hard deadline – having a crib once the baby arrives.

Here the goal is clear – to have a crib by a due date. But in addition, there is a set of preferences: (a) the user prefers a used crib but if none are available within 1.5 months, she will purchase a new one (although what if a perfect used crib becomes available within days of placing an order for a new crib. Can that order be canceled?); (b) the user wants to avoid shipping from the US in order to avoid customs delays as well as extra taxes. This means that if she buys the crib on a US website, she must remember to ship it to her parents' house (context-based preferences).

To accomplish this scenario, the user needs to interact with various services/sites:

- Research: product databases, review sites, user groups and forums.
- Purchase: auction sites, online classified ads, online retailers (and of course the related payment processing).
- Shipping: shipping estimator, shipping, truck rental.
- Utilities: currency converter, online spreadsheet, email, calendar, task lists.

She also needs to keep her parents up-to-date on the crib search, in order to avoid buying two cribs and coordinate travel dates in case her parents find a crib first.

To make this scenario into a Personal Web application requires, effectively, producing an orchestration for the above services, which allows prioritizing, context aware information, concurrency, and even undoing a finished task (such as attempting to cancel an order for a new crib if a used crib is found soon after). The orchestration should satisfy a number of properties, among them (a) a crib must arrive before the due date; and (2) at most one crib should arrive. For example, the last property is violated in the scenario where both the user and her parents independently find a good local deal on the same day and both decide to buy the crib on the spot to avoid losing the deal.

3 Challenges

As we mentioned earlier, we envision that many workflows can be "prebuilt" or community-shared – much like iPhone applets. However, there should be cases when a user may want to build their own customized and complex workflow, like in our example in Section 2. In this section, we describe specification and analysis challenges associated for providing support for automating complex user-created orchestrations.

1. The first obvious challenge is *specification* – specification of the desired outcome of the orchestration, available services, properties of the orchestrations, preferences, context, etc. The outcome of such specification should be sufficiently precise, so as to enable creation and reasoning about non-trivial service orchestrations, without the user knowing the technical details of service configuration or sequencing. And of course such specifications should be "average" user readable, without resorting to the use of formal logic.

2. Monitoring and statically analyzing correctness of orchestrations. Personal workflows are operationalized through orchestrations of user-level services. These orchestrations might be modified or recreated at runtime as services become unsuitable because of changes in user preferences and constraints (i.e., upon failing to secure the crib in time, the user may want to apply a completely different strategy such as borrowing a bassinet from a friend or deciding to do without a crib for the first few months of the baby's life).

1. Assuming it is possible to use the cloud or the user's own machine to check the workflow against the goals, how do we go about doing it without a major slowdown in workflow execution. What sorts of analyses are appropriate and how to make them scale? (the "compositionality" challenge)
2. Another set of questions for this challenge involves figuring out when a particular service can be substituted for another, whether it can be used in place of a collection of services in an orchestration, and suggesting which combinations are feasible. To enable personalized workflows, it is important to be able to aggregate services, resources, and content from multiple web services centering on the user, her tasks and context. In the crib purchasing workflow, the user may include several online retailers in her bookmarks in the search for a usable and inexpensive crib. Each of these retailers has their own specific shipping policies and methods. To achieve this simple workflow, several activities need to be performed: Different services and resources need to be identified. The compatibility of the interfaces of these services has to be verified. The services might need to be substituted dynamically when the user preferences or context change. For example, if the user realizes that her parents have bought a crib, the services in her workflow related to crib purchasing should be replaced with proper services enabling and coordinating crib shipping or the parents' travel schedule from US. Finally, the services must be composed periodically. (the "compatibility" challenge)

3. Repairing orchestrations. Errors occur in most software applications, but they are unavoidable in web-based applications. They can happen because some partner went down, Internet became unavailable or the logic got violated. In our example, the user wants to buy exactly one crib. However, it could happen that both the user and her parents buy a crib on the same day, violating this user requirement. Repairing user-created faulty orchestrations is clearly needed.

Of course, the users should be able to specify orchestrations and their properties, and once an orchestration is deployed, monitor and repair the orchestration to make sure that their goals are accomplished. We also envision some form of an "orchestration dashboard", where users can create new orchestrations and check state of orchestration instances.

4 Approaches

In this section, we discuss some ideas to approach the challenges identified in Section 3.

1. The Specification challenge. One way to specify workflows is to avoid such specification altogether! Specifically, we may want to synthesize complex workflows automatically, based on the expression of user intent. Effectively, intent is a declarative specification which is then turned, by synthesis, into an operationalized workflow. One way to approach it is to adapt existing work on configuring personal software using goal models [5,6].

In this work, an i* goal model [11] is used to specify a set of possible user goals, how they interact and how they decompose into simpler goals. Each low level service configuration setting contributes, to different degrees, to different higher level goals. For example, in our scenario, the user has a goal "Minimize cost". In this case, each product provider service can specify how much its configuration settings support this goal (e.g., selecting "used" rather than "new" items will contribute more to this goal). Thus, by allowing the users to identify their goals, the optimal service configuration settings can be determined automatically. Furthermore, this can be used to identify when user goals are unsatisfiable (since no settings exist to satisfy them all) and hence require the user to modify their goals.

The goal model also supports AND/OR decomposition and this can be used to automatically define a service sequence. An AND decomposition step defines the set of subgoals that must be satisfied to satisfy a goal. This can be used to constrain the possible service invocations required and the input/output dependencies between these services constrain the possible sequencing of them. An OR decomposition step specifies a set of alternative subgoals and this leads to an interaction with the user to refine their intent by choosing an alternative. For example, the goal "Deliver product quickly" can have alternatives "Deliver within 2 weeks", "Deliver within 6 weeks".

2. The Monitoring and Analysis challenge. To address the compositionality challenge, we begin by looking at approaches for verifying web service orchestrations. Such verification can be done statically or dynamically, e.g., [1–3, 9] and can be relatively easily adapted to user-created orchestrations. However, the distinction between the two approaches is that the user-created ones are much more agile – once anything in the orchestration changes, the whole analysis may have to be redone, and the user is forced to be aware of it as the performance of her system deteriorates. For example, the verification may concern basic functional properties such as "the crib is eventually bought" or the more complex ones such as "the crib is usable and has a reasonable price", "only one crib is bought", or "the crib has been delivered within an acceptable time period". However, periodic application of these techniques in service compositions that evolve over time, where services are frequently added, removed, or revised, is unrealistic. To ensure correctness, we need to design service compositions in a way that verification results can be reused across evolutions, i.e., to allow *regression verification* [4].

To achieve this goal, we propose to exploit composition design patterns [8] and orchestration algorithms that make verification *change-aware*.

To address the substitutability challenge, we aim to create support for compatibility and substitutability of web services. We propose to do this by investigating similarity measures between behavioural models [7] to identify candidate services to replace a

service in use when it becomes unavailable or unsuitable due to evolving needs or a change in the context. Further, we require composition techniques for combining these services. This is a non-trivial activity as it requires integrating both behaviour and data assets of services while preserving their desirable properties and ensuring that interactions between services within a composition do not lead to unforeseen and highly undesirable side-effects.

3. Repairing orchestrations. The goal of this activity is to use semantic information about services involved to try to fix the problem discovered for an orchestration. For example, consider the case when two cribs were purchased at the same time. The problem can be fixed if the user sells one of the cribs, as this returns the orchestration to a state where the user has just one crib. The user may have preferences as to how to accomplish this, e.g., she may choose to sell the most expensive crib, or maybe the one that will arrive last, or the one bought by her parents.

In another scenario, imagine that the parents bought a second-hand crib, but decided to ship it instead of transporting it themselves. Due to a backlog at the customs office, the crib will not arrive by the required date (violating the "the crib has been delivered within an acceptable time period" property). In this case, we can repair the orchestration by suggesting that the user buy a new crib locally (which leads to the satisfaction of the violated property), while selling the other crib when it arrives (to avoid violating the "at most one crib" property).

In the first scenario, we suggested actions that compensate executed actions, leaving the orchestration in a state that does not violate any user requirements. In the second scenario, we also suggested the execution of new activities that lead to the satisfaction of user requirements. We have explored the idea of property-guided recovery in the context of traditional web applications [9, 10], where both the orchestration and its properties are defined by the application developer, but recovery plans are computed for individual execution traces. This framework can be adapted to the Personal Web paradigm. However, as we mentioned earlier, the success of such an approach depends on the ability of end users to specify correctness properties. Also, in the approach of [9, 10], activity compensation and cost are statically defined. In order to move our approach to the Personal Web, compensation and its cost should be user-specified (e.g., to account for cases where some users pay smaller fees for a transaction cancellation, be that for a stop payment or for cancelling a flight). This is theoretically possible, of course, but so far we are not aware of technology that would allow us such dynamic, user-centered compensation definition and configuration.

5 Summary

In summary, the dream of Personal Web seems achievable, provided a number of challenges are met. Many of these are technological, and we have no doubt about their success. Some others need advanced techniques and thus require research. We tried to argue that problems of modeling and analysis of end-user created orchestrations are important, and solutions to them are possible. However, success of the whole endeavor depends on several computing fields and, most crucially, on creating a good user experience, especially in support for specifying desired orchestrations and their properties.

We look forward to collaborating with others on various aspects of problems brought forth by Personal Web.

References

1. H. Foster, S. Uchitel, J. Magee, and J. Kramer. "Model-based Verification of Web Service Compositions". In *ASE '03: Proceedings of 18th IEEE International Conference on Automated Software Engineering*, pages 152–163. IEEE Computer Society, 2003.
2. X. Fu, T. Bultan, and J. Su. "Analysis of Interacting BPEL Web Services". In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 621–630, May 2004.
3. S. Hallé and R. Villemaire. "Runtime Monitoring of Message-Based Workflows with Data". In *ECOC '08: Proceedings of the 12th IEEE Enterprise Distributed Object Computing Conference*, pages 63–72, 2008.
4. T. A. Henzinger, R. Jhala, R. Majumdar, , and M. Sanvido. "Extreme Model Checking". In *Verification: Theory and Practice*, volume 2772 of *Lecture Notes in Computer Science*, pages 332–358, 2004.
5. S. Liaskos, A. Lapouchnian, Y. Wang, Y. Yu, and S. Easterbrook. "Configuring Common Personal Software: a Requirements-Driven Approach". In *RE '05: Proceedings of International Conference on Requirements Engineering*, pages 9–18, 2005.
6. S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, and J. Mylopoulos. "On Goal-based Variability Acquisition and Analysis". In *RE '06: Proceedings of International Conference on Requirements Engineering*, pages 76–85, 2006.
7. S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. "Matching and Merging of Statechart Specifications". In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, pages 54–64, 2007.
8. S. Nejati, M. Sabetzadeh, M. Chechik, S. Uchitel, and P. Zave. "Towards Compositional Synthesis of Evolving Systems". In *FSE '08: Proceedings of SIGSOFT International Conference on Foundations of Software Engineering*, pages 285–296, 2008.
9. J. Simmonds, S. Ben-David, and M. Chechik. "Guided Recovery for Web Service Applications". In *FSE '10: Proceedings of SIGSOFT International Conference on Foundations of Software Engineering*, pages 1–10, 2010. to appear.
10. J. Simmonds, S. Ben-David, and M. Chechik. "Monitoring and Recovery of Web Service Applications". In *Smart Internet*, pages 1–40. Springer, 2010. to appear.
11. Eric S. K. Yu. "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering". In *RE '97: Proceedings of IEEE International Symposium on Requirements Engineering*, pages 226–235, 1997.