Personalized Web-Tasking Applications: An Online Grocery Shopping Prototype

Lorena Castañeda *†‡, Hausi A. Müller *†, Norha M. Villegas‡*
* *Department of Computer Science. University of Victoria. Victoria, Canada email: {lcastane,hausi}@uvic.ca*
‡ *Dept. of Information and Communication Technologies. Icesi University. Cali, Colombia email: nvillega@icesi.edu.co*
† *IBM Canada Software Laboratory, CAS Research. Markham, Canada*

***Abstract*—Nowadays, users utilize web applications to perform everyday tasks in order to achieve personal goals. Personalized Web-Tasking (PWT) is the automation of such web interactions while exploiting personal context to enrich users experience. However, web-tasking is affected by unpredictable context behaviour—environment, user, and infrastructure—and situational changes. Given that current web systems are challenged to respond effectively to such changes, we proposed to design PWT applications as self-adaptive software systems that exploit personal context to deliver user-centric functionalities. This paper presents our first approach implementing PWT applications using a grocery shopping web-tasking scenario. Our prototype PWT system transforms web-tasking knowledge information (i.e., user's web interactions) into RDF graphs (i.e., runtime models that contain the user's web-tasking). We conclude our paper with a discussion about our results and implementation challenges.**

***Keywords*-Personalized Web-Tasking, Self-Adaptive Systems, Runtime Models, Semantic Web, Context management.**

I. Introduction

At the core of smart web applications is a user-centric model—users and their concerns become the centre of all web interactions to improve the users' experience by providing personalization and automation for web interactions [1]. We define personalized web-tasking as the automation of repetitive tasks, while exploiting contextual information—about the users and the environment [2]. Context representation is highly relevant in the realization of personalized web-tasking systems [3].

In our vision, smart web applications implemented as situation-aware self-adaptive software systems are important to support personalized web-tasking. First, situation-awareness is the capability of the system to not only be aware of contextual changes, but also capable to reason about them and provide decision-making support [4], [5]. Second, there is a need for more flexible and dynamic software systems that are able to respond to unexpected context changes, which are common in user's daily web-tasking. A self-adaptive software system responds to uncertainty by performing changes over itself (structural or behavioural) at runtime, to maintain the relevance of the system with its objectives [6], [7].

Dynamic systems such as self-adaptive software, require monitoring and adaptation tasks to be performed at runtime. That is, sensing context, reasoning about adaptation, and performing changes must not interrupt the normal execution of the system. To achieve this dynamism and flexibility, self-adaptive systems require proper models that can be monitored and adapted at runtime. That is, runtime models that represent the structure and behaviour of the system to provide information about the current state of the system [8], [9].

We proposed a PWT application as a self-adaptive software system, based on the DYNAMICO reference model proposed by Villegas et al. [10], and on our runtime models [11]: (1) our PWT model to represent the concepts of personalized web tasking, and (2) our goal-oriented context-sensitive web-tasking (GCT) model to represent the evolving web-tasking personal goals of the user, as well as the relevant context. Our PWT System comprises four software modules: (i) Web-Tasking Knowledge Infrastructure, which provides a mechanism to express personal user goals in the form of a GCT model; (ii) PWT Model Processor, to analyse our PWT models and generate the corresponding RDF graphs that contain the information about the user's web-tasking; (iii) Personalization Engine, which uses personal context information to improve the execution of the web-tasking; and (iv) Web-Tasking Effector, which executes the sequence of web interactions on behalf of the user while managing the life-cycle of the web-tasking . Additionally, we take advantage of the SmarterContext Monitoring Infrastructure developed by Villegas [12], to provide context-awareness support.

To demonstrate the feasibility of our PWT system and runtime modelling approaches, we implemented a PWT grocery shopping prototype. The realization of our prototype consisted of four major stages: (1) the implementation of our models, (2) the definition of the information about grocery shopping web-tasking, (3) the implementation of the PWT system's software modules, and (4) the generation of the personalized web-tasking instance of a user in the grocery shopping scenario.

The remaining sections of this paper are organized as follows: Section II presents our grocery shopping web-tasking scenario in which changing context and personalization are required for web-tasking decision making and execution. This scenario presents the interaction of multiple sources of context information as well as the interconnections of heterogeneous systems and devices. Section III describes

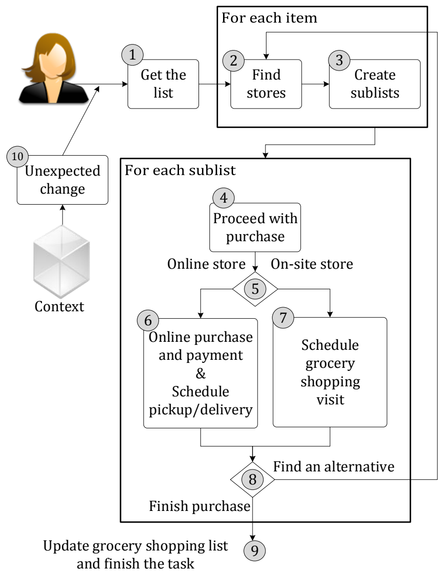

Figure 1. Manual web-task process of a Grocery Shopping scenario

our approach building PWT systems: our personalized web-tasking (PWT) model; our goal-oriented context-sensitive web-tasking (GCT) model; and our PWT System design as a context-aware self-adaptive software system. Section IV describes our prototype implementation in order to validate our approach, and discusses the results. Finally, Section V concludes this paper with a summary and a discussion over the direction of our research.

II. Grocery Shopping Web-Tasking

A compelling application for personalized web-tasking is grocery shopping because its activities must be performed regularly and involves the usage of multiple services and diverse data sources from the web. Figure 1 illustrates the process of the manual web tasks that a user typically follows to accomplish a grocery shopping goal. These web tasks are identified in the figure using rounded labels numbered 1 through 9, and are depicted as an ordered sequence of activities. In most cases, these linked web tasks are dependent on each other (e.g., the outcome of one is the input of the next one).

For our scenario, the details of these activities can be summarized as follows. In Task ① the user logs into her preferred grocery list application to find her shopping list (i.e., a list of items, quantities, and expected purchase prices), through a web service accessed from her browser or mobile device. In Tasks ② and ③, for each item—and taking into account different restrictions, such as prices, deals, localizations, and personal preferences—the user manually selects the proper grocery stores to purchase every item, and creates independent sublists labelling the items with a selected store for its purchase. For this, the user consumes web services from every preferred store's website to obtain location information and product prices. Other web services might be consumed for special deals, coupons, or discounts. In Task ④, for every sublist the user performs a purchase. According to Task ⑤, the user must decide between performing Task ⑥ or ⑦ based on whether the store provides online purchase services. If the user performs Task ⑥, she checks out the desired items using traditional online payment web services, and schedules the groceries pick-up (or delivery if available) and saves it as a calendar event. If the user performs Task ⑦ by shopping the grocery items directly in the store's locations, she schedules calendar reminders to visit the stores of the lists, and decides about the best time according to her availability and the stores' hours. Finally, using mapping applications, she plans the best route for visiting the grocery stores.

Nevertheless, it can happen that not all items can be purchased at Task ⑤. In this case, the user must perform Task ⑧ to find an alternative, which implies to repeat the process starting from Task ② for those items, but selecting alternative criteria. Finally in Task ⑨, the user decides whether the process is finished and manually updates the grocery shopping list application. For this, she must remove those items that where purchased, update prices and budget, and add relevant information for the next grocery shopping process (e.g., weekly deals, calendar events, or budget).

Grocery shopping activities are affected by many contextual variables (cf. Label ⑩ in Figure 1). Moreover, unpredictable changes associated with context entities—such as social events in the user's calendar, availability of the items, or budget restrictions—may require manual adaptations of the grocery shopping process. In certain cases, the user may be unaware of the new requirements, such as, upcoming sales or special offers, which affect the user's satisfaction. At the end of the process, the user should be satisfied with the result of the grocery shopping tasks—the majority of the items in the shopping list where purchased and the total spent was within her budget restrictions.

III. Our PWT System Approach

Recently we proposed two runtime modelling approaches [11]: (1) our Personalized Web-Tasking (PWT) model, and (2) our Goal-Oriented Context-Sensitive Web-Tasking (GCT) model. These runtime models define the elements and concepts of personalized web-tasking, as well as the representation of the personal goals of the user and relevant context.

We proposed an abstract representation of its architecture for PWT applications as self-adaptive software systems as depicted in Figure 2. In our approach, we follow the DYNAMICO reference model proposed by Villegas et al. [10] to guarantee context-driven self-adaptation support. We defined four software modules (i.e., Web-Tasking Knowledge Infrastructure, PWT model processor, Personalization Engine, and

Web-Tasking Effector) to realize the instantiation, execution and runtime adaptation of the personalized web-tasking. Our runtime models (PWT model and GCT model) communicate with all three DYNAMICO layers to provide runtime adaptation capabilities (i.e., changes in the web-tasking conceptual elements, the PWT system infrastructure, and the system requirements). Finally, in the lower layer of our design we take advantage of the existing SMARTERCONTEXT Monitoring Infrastructure proposed by Villegas [12] to support context-awareness capabilities to our system [11]. The *Web-Tasking Knowledge Infrastructure* is the platform to understand the personal user goals. The *PWT Model Processor* analyses our models to extract all the information about the user web-tasking and the *Personalization Engine* will exploit personal context of the user to select the web-task sequence that is to be executed. Finally, the *Web-Tasking Effector* executes such web-task sequence on behalf of the user while acting as a controller of the system.

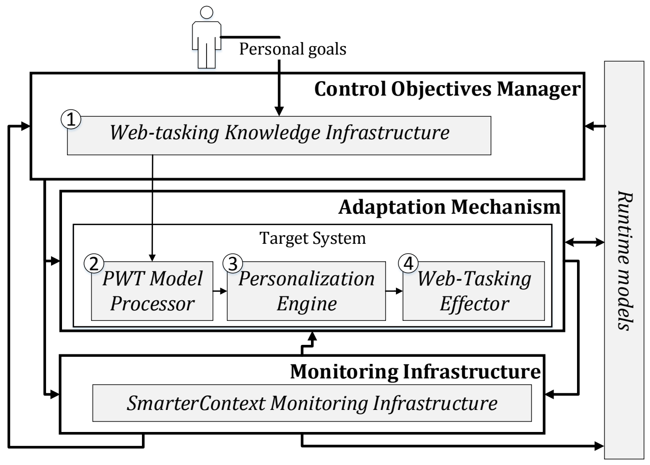
Personal goals
Control Objectives Manager
1
Web-tasking Knowledge Infrastructure
Adaptation Mechanism
Target System
2
PWT Model Processor
3
Personalization Engine
4
Web-Tasking Effector
Monitoring Infrastructure
SmarterContext Monitoring Infrastructure
Runtime models

Figure 2. Our PWT system approach as a self-adaptive software system.

IV. PWT APPROACH IMPLEMENTATION

To validate the feasibility of our approach we implemented a prototype for the grocery shopping scenario. We realized our implementation in four stages: first, we implemented our models using OWL2/RDF[1] using the Protégé ontology editor tool[2] (cf. Figure 3). Second, we analysed and selected the input data of the *Web-Tasking Knowledge Infrastructure* component, that is, the information about the web-tasking of the user required to achieve a grocery shopping goal. Third, we implemented a preliminary version of our *PWT Model Processor* component as an Apache JENA application[3] to create our GCT model of the user's web-tasking by analysing the data from the Web-tasking Knowledge Infrastructure, and the PWT model OWL/RDF file. Finally, to asses our approach we implemented a preliminary version of the *Personalization Engine* to generate the corresponding RDF graphs that will contain the information of the user's web-tasking[4] (i.e., web-task sequence, web services, and preferences).

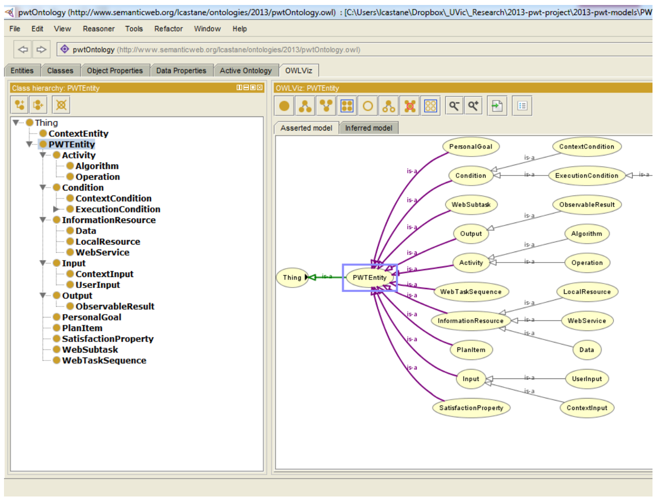
pwtOntology (http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl) : [C:\Users\lcastane\Dropbox_UVic_Research\2013-pwt-project\2013-pwt-models\PW
File Edit View Reasoner Tools Refactor Window Help
pwtOntology (http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl)
Entities Classes Object Properties Data Properties Active Ontology OWLViz
Class hierarchy: PWTEntity
OWLViz: PWTEntity
Asserted model Inferred model
Thing
ContextEntity
PWTEntity
Activity
Algorithm
Operation
Condition
ContextCondition
ExecutionCondition
InformationResource
Data
LocalResource
WebService
Input
ContextInput
UserInput
Output
ObservableResult
PersonalGoal
PlanItem
SatisfactionProperty
WebSubtask
WebTaskSequence
is-a

Figure 3. Protégé interface of the PWT model ontology

A. Web-Tasking Knowledge Information

To populate the information of the web-tasking knowledge infrastructure we defined two personal goals for the grocery shopping expressed in observable results as represented in our PWT model [11]: (1) The numbers of items left in the grocery shopping list application at the end of the web-tasking execution should be less than 20% of the original. (2) The total amount spent in the groceries purchase should be within a $200 budget.

Then, we defined the elements in every web-task. That is the execution plan, web-subtasks activities, and resources. For instance, Table I defines the first three steps of the execution plan, and for each step its corresponding predecessor. In this examples web-subtasks 1 and 2 can be executed in parallel, while 3 depends on the execution of 2.

Table I
EXAMPLE VIEW OF THE EXECUTION PLAN INFORMATION LOCATED IN THE WEB-TASKING KNOWLEDGE INFRASTRUCTURE

Exec. Order	Predecessor	web sub-task description
1	0	Get personal goal metrics
2	0	Get user's grocery shopping list application
3	2	Connect to the Grocery Shopping list application (Authenticate) and retrieve grocery list items

Similarly, the information of the resources required in the web-tasking includes the resource name (as identified

[1]http://www.rigiresearch.com/research/pwt/pwtOntology.owl
[2]http://protege.stanford.edu/
[3]http://jena.apache.org

[4]http://www.rigiresearch.com/research/pwt/GroceryShopping.rdf

by the PWT System), source (i.e., the IP address of the web services, or the name of the function) and type (i.e., resources can be web services, data files, or local services). Table II is an example of three resources of our scenario. For instance *GSApp* represents the grocery list application web service, and *CSApp.list* is the data that represents the items in the grocery list, such as product description and price. Also the *SCCore* represents the SMARTERCONTEXT web service that provides user's personal context.

Table II
EXAMPLE VIEW OF THE INFORMATION RESOURCES

Name	Source	Type
GSApp	http://www.groceryWebApp.com/GList	Web service
GSApp.list	GS_App.getShoppingList_m	Data
SCCore	http://200.3.193.34:9080/CtextProvider Component /ContextProvider	Web service

B. Context Information

The personal context used by the Personalization Engine is described in Table III. This information is extracted from the SMARTERCONTEXT Reasoning Engine developed by Villegas [12], based on the preferences of the user and past web-tasking interactions.

Table III
PERSONAL CONTEXT INFORMATION REQUIRED OF THE PERSONALIZATION ENGINE COMPONENT

Information	**Details**
Grocery Shopping list application	URL of the web service, and authentication credentials
Grocery Shopping Stores	URL of the stores' websites in order of preference (most preferred first)
Purchase criteria information	The rules of purchase in order of importance (i.e., decide stores by lowest price; purchase urgent items first; purchase remaining items first the store with the largest list)
Payment information	Type of payment (Credit card information)
Personal goal metrics	Personal goals observable results, satisfaction properties, and thresholds

For the SMARTERCONTEXT Reasoning Engine to understand our application domain, we extended the Shopping Ontology proposed by Villegas [12], by adding the class Grocery and its subclasses (e.g., Vegetable, Beverages, Diary, Frozen, Paper, Personal, and Baby) (cf. dashed box in Figure 4) .

C. Observations and Results

Figure 5 presents a visualization of the resulting RDF graph of our system. This visualization allows us to understand the final outcome of our PWT system.

Each node of the RDF graph corresponds to web-tasking information as defined by our GCT model. For instance, the cluster of nodes in the upper right corner correspond

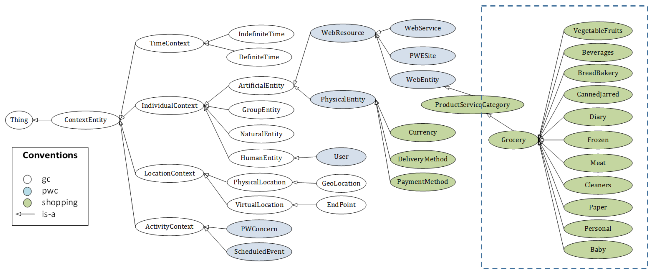

Figure 4. Ontology subset featuring the online grocery shopping scenario. The dashed box represents our extension

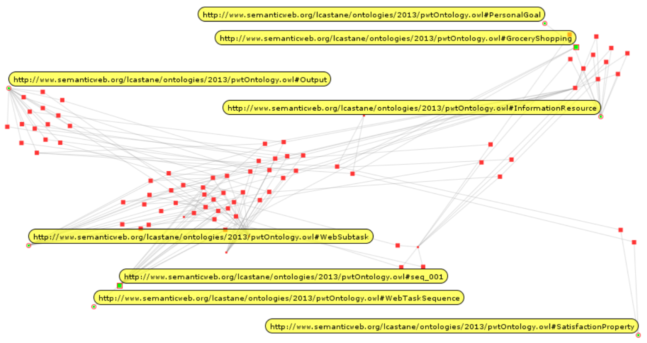
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#PersonalGoal
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#GroceryShopping
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#Output
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#InformationResource
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#WebSubtask
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#seq_001
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#WebTaskSequence
http://www.semanticweb.org/lcastane/ontologies/2013/pwtOntology.owl#SatisfactionProperty

Figure 5. Visualization of an RDF graph that represent web tasks to be executed by our PWT system

to the information resources of our scenario. The clusters in the middle represent the sequence of web-subtasks that will be executed on behalf of the user. Each element in the sequence is also connected to its corresponding inputs and activities. The upper left cluster of nodes are the outputs of the web-subtasks in the sequence. Finally the two isolated nodes in the lower right corner are represent two satisfaction properties (i.e., the grocery list number of items and the total expenditure) of the web-tasking.

Using this visualization we were able to identify all the nodes, their relation with each other (i.e., the edges of the graph), and their stored information (i.e., PWT information). We manually tracked the web-task sequence to validate its representation of the web-tasking of the user according our scenario, and if the information provided was sufficient for the system to execute the web-task sequence on the user's behalf.

Our prototype demonstrates that it is possible to represent personalized web-tasking for simple scenarios, such as grocery shopping, using our runtime models: the PWT model and the GCT model. However, more complex scenarios would require to model additional elements that are not currently supported by our models such as conflict resolution mechanisms, data type translation between web-subtasks and web services. Likewise, in our implementation each component in our PWT system functional behaviour is sufficient to transform web-tasking knowledge into the RDF graph.

V. Conclusions

To demonstrate the feasibility and effectiveness of using context-aware self-adaptive software in personalized web-tasking, we presented an online shopping web-tasking scenario, our PWT system, and our two modelling approaches (i.e., the personalized web-tasking (PWT) model and the goal-oriented context-sensitive web-tasking (GCT) model).

We presented our PWT System prototype which was developed using different tools and technologies (i.e., Protégé ontology editor, Apache Jena, and the SmarterContext reasoning engine)

Based on our observations and results, we posit the following technical challenges to achieve the next level of web-tasking: (1) Refine the Web-Tasking Knowledge Infrastructure: To define the technology requirements for the infrastructure to reason about a user's personal goals, and represent them as web interactions specified in our GCT runtime model. An important aspect in personalized web-tasking is to understand the user's behaviour and preferences including past web interactions to be able to automate his or her web-tasks. Despite current technologies that enable us to record web interactions, there is no understanding of the user's intentions or personal goals while performing those interactions. In fact, there is no instrumentation to measure satisfaction that can be associated with the achievement of personal goals.

(2) Resolve the connections between our PWT and GCT models: We need to develop methods to specify the causal connection between these two models explicitly and operationally, and automate this process. Moreover, manage the propagation of changes given this causal connection. This will also affect our personalized shopping RDF graphs that encode information from past shopping experiences (e.g., preferred stores for particular items, discount cards, air miles cards, and credit cards used for particular stores, coupon servers used, or preferred brand items versus least expensive items including sale-only items purchased).

(3) Broaden our context- and situation-aware instrumentation: Although our system supports the monitoring and reasoning of context, we still need to extend its instrumentation in order to gather information from other sources, such as social media and other devices in order to provide better levels of personalization and automation. Likewise, our vision is to create situation-aware systems which require instrumentation to understand users changing situations (i.e., her personal goal changes) in which case, the PWT system needs to maintain its relevance.

In conclusion, we were able to realize a implementation of our personalized web-tasking system approach. In our ongoing research, we investigate other sources of context that can enhance the personalization of the web-tasking. For example, social context which is all relevant information gathered from the social relationships of the user. Also, we aim to explore techniques to provide automation support to the PWT system based on predictive analytics or data mining algorithms.

Acknowledgments

This work is funded in part by University of Victoria (Canada), IBM Canada Software Laboratory (CAS Research), the National Sciences and Engineering Research Council (NSERC) of Canada under the NSERC Strategic Research Network for Smart Applications on Virtual Infrastructure (SAVI - NETGP 397724-10), and Icesi University (Colombia).

About the Authors

Lorena Castañeda is a PhD student in the Department of Computer Science at the University of Victoria, Canada. She is a CAS student at the Center for Advanced Studies at the IBM Toronto Laboratory. She holds a double degree in Engineering: Computer Engineering (2007) and Telecommunications Engineering (2007), and a Master in Information and Communications Management (2012) from Icesi University, Colombia. Her research interests include smart applications, self-adaptive systems, situation- and context-aware systems, user-centric applications, and personalized web-tasking. Email: lcastane@uvic.ca.

Norha M. Villegas is an assistant professor at Icesi University, Colombia and research associate at University of Victoria, Canada. She received her PhD degree in Computer Science from University of Victoria, Canada. She was a CAS student at the Center for Advanced Studies at the IBM Toronto Laboratory. Her research interests focus on context-aware self-adaptive software systems. She received a Diploma Degree in Computer Systems Engineering and a Graduate Degree in Organizational Informatics Management in 2002 and 2004, from Icesi University, Colombia. Email: nvillega@icesi.edu.co.

Hausi A. Müller is a Professor, Department of Computer Science and Associate Dean of Research, Faculty of Engineering at University of Victoria, Canada. He is a Visiting Scientist at the Center for Advanced Studies at the IBM Toronto Laboratory (CAS). Dr. Müller's research interests include software engineering, self-adaptive and self-managing systems, context-aware systems, and service-oriented systems. He serves on the Editorial Board of Software Maintenance and Evolution and Software Process: Improvement and Practice (JSME). He served on the Editorial Board of IEEE Transactions on Software Engineering (TSE). He is Chair of the IEEE Technical Council on Software Engineering (TCSE). Dr. Müller received a Diploma Degree in Electrical Engineering in 1979 from the Swiss

Federal Institute of Technology (ETH), Zürich, Switzerland and MSc and PhD degrees in Computer Science in 1984 and 1986 from Rice University in Houston, Texas, USA. Email: hausi@uvic.ca.

References

[1] J. W. Ng, M. Chignell, J. R. Cordy, and Y. Yesha, "Smart interactions," in *The Smart Internet*. Springer, 2010, pp. 59–64.

[2] L. Castañeda, N. M. Villegas, and H. A. Müller, "Towards personalized web-tasking: Task simplification challenges," in *Proceedings 1st Workshop on Personalized Web-Tasking (PWT 2013) at Ninth IEEE World Congress on Services (SERVICES 2013)*, 2013, pp. 147–153.

[3] N. Villegas and H. Müller, "Managing dynamic context to optimize smart interactions and services," in *The Smart Internet*. Springer, 2010, vol. 6400, pp. 289–318.

[4] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems: Situation awareness," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[5] ——, *Designing for Situation Awareness: An Approach to User-Centered Design, Second Edition*. CRC Press, 2011.

[6] H. A. Müller, H. M. Kienle, and U. Stege, "Autonomic computing: Now you see it, now you don't. Design and evolution of autonomic software systems," in *Software Engineering*. Springer, 2009, vol. 5413, pp. 32–54.

[7] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, , and J. Whittle, "Software engineering for self-adaptive systems: A research roadmap," *Software engineering for self-adaptive systems*, vol. 5525, pp. 1 – 26, 2009.

[8] N. Bencomo, A. Bennaceur, P. Grace, G. Blair, and V. Issarny, "The role of Models@run.time in supporting on-the-fly interoperability," *Computing*, vol. 95, no. 3, pp. 167–190, 2013.

[9] B. Morin, O. Barais, J. Jezequel, F. Fleurey, and A. Solberg, "Models@run.time to support dynamic adaptation," *Computer*, vol. 42, no. 10, pp. 44–51, 2009.

[10] N. Villegas, G. Tamura, H. Müller, L. Duchien, and R. Casallas, "DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems," in *Software Engineering for Self-Adaptive Systems II*. Springer, 2013, vol. 7475, pp. 265–293.

[11] L. Castañeda, N. M. Villegas, and H. A. Müller, "Self-adaptive applications: On the development of personalized web-tasking systems (SEAMS 2014)," *ACM/IEEE*, (To Appear) 2014.

[12] N. M. Villegas, "Context Management and Self-Adaptivity for Situation-Aware Smart Software Systems," Ph.D. dissertation, University of Victoria, Canada, February 2013.