Towards Smarter Task Applications

Przemek Lach
Department of Computer Science
University of Victoria, UVic
Victoria, British Columbia
Email: przemek@uvic.ca

Hausi A. Müller
Department of Computer Science
University of Victoria, UVic
Victoria, British Columbia
Email: hausi@cs.uvic.ca

***Abstract*—Mobile devices offer an unprecedented amount of context about their users. Management of this context is like trying to find the signal in the noise. Those applications that can find the signal open themselves up to new business opportunities. These business opportunities come about as a result of emergent behavior and are better at satisfying user utility. Applications need to become smart applications and as software engineers we can make this happen by looking at the lessons learned from self-adaptive systems. Data structures, models, and an unfettered resolve to simplifying the user experience will help us get there.**

***Keywords*-Smart Applications, Self-Adaptive Systems, Mobile Devices, Personalized Content, e-Commerce**

I. Introduction

Every morning, still half asleep, millions of people try to plan out their day. In their mind they stitch together a series of events that need to occur in order for them to complete all the tasks for that day. Some of these tasks are daily tasks which do not require much cognitive effort, such as dropping your kids off at school, while others are less routine and require more mental effort, such as taking your car in for repairs. This daily activity has been part of our culture for hundreds of years but it is amazing how many times we still get it wrong. To a certain extent it is our brains that fail us: overcommitment, lack of caffeine, lack of sleep, and difficulty in parallel thinking contribute to less than optimal results.

But don't worry, there's an App for that! A quick search on the Apple[1] or Google[2] stores reveals a plethora of task oriented applications designed to help you overcome the shortcomings of your cerebral cortex. These task oriented applications range from a simple to-do list,[3] to applications that attempt to instill habits by constantly nagging you,[4] to applications that use contextual information, such as your location, to help you get your tasks done more effectively.[5] In fact, there are so many of these types of apps that there are applications designed for the sole purpose of managing all these other applications.[6]

Granted these kinds of applications, such as GoTask, are an improvement compared to trying to organize tasks in your head while you are still half asleep, they still lack automation and intelligence. Even for the more sophisticated task oriented applications, such as GoTask, you still need to manually enter all your tasks, assign dates, and assign locations. Furthermore, most of these tools do not go out and find solutions to your tasks and some force you to create a login and profile. Current applications for task management, inherently, add even more tasks. In other words they are a glorified sticky note and not very smart.

In this position paper, we present several ideas for what it will take for task applications to become smarter task applications and how this additional intelligence creates e-commerce opportunities. Task applications can become smart by making context and feedback loops first class entities. By doing so, smart task applications can simplify the work done by the user while at the same time suggest better solutions. The line of thinking required to make this leap is analogous to the reasons for and the way how self-adaptive systems have evolved. We first present a brief history of self-adaptive systems and use those lessons as a motivation for smart applications.

II. An Increase in Complexity

Over a decade ago it became apparent that IT practices, specifically those around Information Systems (IS), were at odds with what organizations needed. There are countless examples of expensive and sometimes spectacular software projects failures over the last 50 years. Some of these failures were a result of simple mismanagement but most others can be traced back to a misalignment between organizational goals and how software systems were designed and implemented. This realization forced some people to take a long hard look at what organizations wanted and how it was being delivered [1]. The result of these reflections was the concept of emergent behavior: the idea

[1] https://itunes.apple.com/ca/genre/mac-productivity/id12014?mt=12
[2] https://play.google.com/store
[3] GTask
[4] Habit Streak
[5] GeoTask
[6] Status Board

that the needs of the organization are constantly changing and hence should the requirements of the software.

With this realization it became apparent that IS goals needed to change. The traditional approach was static in nature. The thinking at the time was that users knew exactly what they wanted, that systems could be completely specified, and once deployed the requirements of a system would never change. This way of thinking was obviously at odds with the concept of emergent behavior and the results spoke for themselves. The U.S. Department of Defense (DoD) also experienced similar problems and the results of several studies coalesced in a book called *Ultra-Large-Scale Systems The Software Challenges of the Future* [2]. One of the findings published in this book was that only an abysmal 16-34% of software projects were deemed successful. These failures were in line with what other organizations experienced. The report also identified the true characteristics of these systems: decentralized, described using conflicting and diverse requirements, under constant evolution and deployment, composed of heterogeneous changing elements, no clear boundary between system and users, and the need to function during expected failures. These findings were completely at odds with software systems assumptions.

A. Self-Adaptive Systems and Feedback Loops

The shift in the approach to software development resulted in software that was more flexible and a software development life cycle that allowed for organizations to take advantage of emergent behavior more easily. You no longer had to define a rigid set of requirements, clearly identify who a user is, or make any guarantees about the environment in which the software is to operate. This change in perspective was an admission that there is a problem, which is the first step, but admission is not a solution.

This new approach led to the development of systems that moved the decisions usually made at design time to runtime. This evolution impacted the amount of complexity that had to be managed at runtime. Several solutions were presented for managing this complexity one of which was the autonomic manager (AM) [3]. An AM uses dynamic policies, rather than hard and fast rules, defined either by users or other systems to self-adapt or adapt other systems under its control. A fundamental concept that allowed AMs to effectively manage complexity was the idea of the feedback loop.

Feedback loops are not a novel concept. They have existed as part of control theory and used in other engineering disciplines for decades but it wasn't until self-adaptive systems that they began showing up in software engineering. This concept has gained momentum both in theory and in practice, and it is quickly becoming the central component around which self-adaptive systems are being built [4], [5]. AMs and feedback loops allow for the definition of the characteristics associated with self-adaptive system capabilities otherwise known as Self*: self-configuring, self-optimizing, self-healing, and self-protecting.

B. Mobile Devices and the Mobile User

The need for self-adaptive systems arose as a result of an increase in system complexity. System complexity increased because organizations wanted to take advantage of emergent behavior. Today this pattern still holds true but what has changed is the source of the complexity. No longer is this complexity as a result of managing a huge interconnected Ultra Large Scale (ULS) system. Quite the opposite. It's about managing the user.

Over the last ten years we have seen a surge in the use of mobile devices. The latter part of this uptake has been dominated by smartphones and tablets. The idea of the smartphone has been around since the introduction of IBM's Simon in 1993 but it was not until the introduction of the first iPhone where the idea of the post-PC era began to take shape. Recent studies predict that smartphone sales will hit the one billion mark by 2014. In other words, one seventh of humanity will have a new smartphone in their pocket. CISCO predicts that by 2016 there will be 1.4 mobile devices per capita [6].

The proliferation of smartphones and tablets has given rise to a new set of user expectations and it is, in part, these expectations that are driving the complexity. Users expect to have access to their device at all times, they want their applications to be personalized, and they assume that they have access to their data wherever they go. These expectations are already having a profound affect on how organizations do business [7]. One such manifestation of this is the concept of bring your own device (BYOD). Users are bringing their own devices to work and to universities and IT departments have had to adjust. The same applications that work at home now have to also function at work and this can be a tricky sea to circumnavigate.

Another consequence of this mobile revolution has been the rise of the cloud. The big players in this field, including Apple, Google, Amazon, IBM, and Microsoft, have made substantial investments in cloud infrastructure to meet consumer demands. Gartner predicts that by 2014 the personal cloud will surpass the PC as the digital hub [8]. Furthermore, by 2016 more than one third of all user data will be stored on the personal cloud [9]. All this is driven by mobile devices and user demands for data synchronization.

There are three aspects to smartphones, and mobile devices in general, that make them so appealing to users: *context*, *power*, and *trust*. First, unlike most of their desktop counterparts smartphones are outfitted with many sensors such as microphones, cameras, GPS, accelerometers, and

gyroscopes that stream a mountain of context from the environment. Second, despite their small size their hardware specs[7] are comparable to what was only available on a big desktop just a few years ago. Third, smartphones go everywhere with the user and are not usually shared with others. Users consider smartphones to be personal devices and as such place significantly more trust in them. These three aspects have changed the role that smartphones play in people's daily lives.

There is another kind of mobile device, in prototype stage right now, that may take mobile devices to a whole new level: Glass, a wearable computer that Google is intent on making the first mass produced ubiquitous device. In addition to having the usual line up of sensors, such as cameras and GPS, Google Glass is a much more personal device since it is something you wear and not just something you put in your pocket. This type of device will once again change user expectations and bring to light new usage scenarios. These scenarios will be on the client side and as such will require smart applications. In order to get a sense of the types of scenarios Google created the #ifihadglass campaign where people could submit their ideas for building applications running on Glass.[8] The response was overwhelming. People are clearly interested in this device.

III. Get Smart

Systems that manage you have to do so differently than systems that manage themselves. The level of complexity is high in both domains but the complexity is inherently different. The problem is no longer how to manage a large, heterogeneous, and decentralized system but how to manage a firehouse of contextual information that in the end is supposed to make your life easier. There are millions of users with just as many mobile devices all of whom want their device to cater to them. Applications that manage this complexity for the user need to get smart and become smart applications.

A. What is Smart?

At some point in your life you where probably told that "The more effort you put into it, the more you get out of it." These are some wise words to live by but with the current state of technology it is the technology that should be putting in the "effort" and not you. Standard operating procedures still require you to put in all the "effort". For example, a social media site requiring you to fill out pages of profile information or spending hours building your friend lists or a task application requiring that you enter a task name, description, priority, due date, reminder date, location, and dependencies on other tasks. Does this sound familiar?

At first glance it may seem like the social media site and task application are smart when in in fact they are just a glorified filter. Increasing the complexity for the user does not make for a smart application. A smart application manages complexity by hiding it and only revealing it when necessary. Smart applications must strive to provide the functionality they claim without incurring any significant burden on the user. They can do so by employing the lessons learned from self-adaptive systems, feedback loops, the context streams, and the the inherent trust that users place in their smartphones. These are some of the characteristics that make up smart applications and why smartphones are an exciting platform for exploring these approaches.

B. Different Levels of Smart

The goal of smart applications is to reduce the amount of work done by the user via automation. Automation can have an effect on the accuracy of results; therefore, a healthy balance needs to exist between the two. To help illustrate this lets build on the task application we just talked about. Suppose a user has three tasks: buy milk, pay a parking ticket, and pay an on-line bill. Each task requires the filling out of several properties. Once complete, the application can go out over the web and return relatively accurate recommendations based on the properties entered by the user.

A smart version of this application would be able to infer these properties by parsing the list of tasks. It would go out over the web and not only find the recommendations but it would also search for places that are near the user's location and take into account the business hours of those locations. This would all be done behind the scenes and the user would only be notified if they are near a location or within a critical time limit. No login or profile is required. The key is that the user does no more work than if they just used a post-it note. In fact, they actually get more out of doing things this way. That's smart.

This approach also decouples the architecture by hiding the complexity from the user. Decoupling allows developers to experiment with different solutions or to take advantage of emergent behavior. For example, if a developer decides to add weather as a property, more context, they can do so without adding any more work for the the user. Furthermore, an application that does not require a login can more easily store anonymous data that can later be mined to make inferences for other users who are exhibiting similar behaviors. In the near future more and more devices will be a source of context, such as your kitchen appliances or your car. With more context smart applications can not only provide solutions to tasks but also generate the tasks for

[7] http://www.android.gs/samsung-galaxy-s4-vs-nexus-4/

[8] http://www.google.com/glass/start/how-to-get-one/

you in the first place. A web enabled, or Internet enabled, appliance can tell you that you are running out of milk and your Internet enabled car can tell that your parking meter is about to expire. A smart application can amalgamate all the context from different sources and auto-generate your task lists along with recommended solutions.

C. Loginless

One common thread that flows through most tools today, especially social tools, is a requirement for a user login or a profile. From a task application point of view, creating a login or profile is just an extra task added to your to-do list. In addition, most users find it a hassle to have to take the extra time to fill out information and having to remember yet another password. For applications, such as banking, these steps may be necessary but for most other types of smart applications there is no need for a login.

The thought of a loginless personalized application may seem odd at first. Yet, if one reflects on the context mobile devices provide and the trust that users place in them, an opportunity to change the approach arises. One such application that takes this concept to heart is YaKit, aka Yakkit. Yakkit is a loginless, location based messaging system that allows users to communicate with one another based strictly on proximity [10]. This approach mimics what happens in real social situations where you can talk to anyone who is near you without having to create a login or ask their permission to be your friend. Yakkit enhances this social norm by providing a smart application that encourages users to socialize in a natural way rather than a way that is dictated by the technical constructs of a computer.

IV. Smart Applications for Smart Scenarios in Commerce

One claim that can be made is that most tasks result in some kind of commerce transaction. After all nothing in life is free. Smart applications are an exciting opportunity to target consumers more effectively. Traditionally this targeting is done via advertising which, considering its cost[9] and difficulty in measuring effectiveness, is far from ideal. It takes away large sums of revenue from businesses, assuming they can even afford it in the first place, and floods consumers with annoying ads that they did not want to see in the first place. Using context from mobile devices and feedback loops, smart applications can be used to match those that require a service with those that can provide it without the need for spam.

Smart applications that use context and feedback loops are the key here. Consumers are more likely to go to a store if they are near it and even more likely if they are near it and actually need to buy something. The need and the location are just pieces of context. The need can be in the form of a task, I need milk, and the location can come from the phone's GPS, near a grocery store. Using these contexts a smart commerce application can detect that there is a customer nearby who needs to buy milk. At this point the application can send a simple advert notifying the consumer where they can buy milk or go even a step further by enticing them with a digital coupon. That same digital coupon can then be used to analyze the effectiveness of the advertising campaign. This is just one example of how smart applications can substantially improve commerce.

Even though Yakkit is primarily a messaging system it lends itself nicely to commerce applications because it makes context and feedback loops, in the form of location, first class entities; no login or profile is required. An added benefit of using smart applications for commerce is that it is easier to blend the original functionality of the application with a commerce opportunity. The location based messages that show up on the Yakkit application are usually messages from other users nearby but they could equally well be messages from businesses. The different types of messages are indistinguishable from one another and are relevant to the user's current situation. There is no special area for ads which the user can simply start ignoring over time. The delivery of all content, including ads, is based on the user's context.

V. Building Smart Applications

Smart applications allow us to reason more effectively about complexity but that complexity still has to be dealt with somewhere. Clearly, one of the primary goals of smart applications is to simplify the life of the user but how one goes about implementing such an application is a tricky problem. In order to be effective, smart applications need to continuously monitor and analyze all sorts of context for every user in the system. This is no small feat especially when you consider a smart application such as Yakkit which if used in a city like New York needs to be able to consider millions of pieces of context for thousands of users almost instantaneously.

Since Yakkit's central piece of context is location a quad tree data structure was used to store user locations [11]. Quad trees are a good and efficient choice for storing two dimensional data which in the case of Yakkit was the latitude and longitude. The quad tree was then implemented over a distributed hash table which allowed for the scalability of the data structure. All this work just to deal with one piece of context: location.

Smart applications, and the subsequent versions of Yakkit, use more than one piece of context some of which are non-deterministic. Analyzing a piece of location context

[9] http://www.gaebler.com/Newspaper-Advertising-Costs.htm

is relatively straight forward but trying to recommend whether one course of action is better than another is a problem that requires a probabilistic approach. What is considered better changes with context and at best we can only assign a probability. For example, Yakkit can be used to produce a list of nearby shoe stores with line ups sorted from shortest to longest. What is considered long for one person is not long for another. Consequently, the best effort that Yakkit can make is to list each of those stores based on a probability, or confidence. This type of reasoning requires data structures and models other than a quad tree.

One such promising model is the use of Bayesian networks [12]. Bayesian networks are useful for modeling problems whose solutions have a degree of uncertainty. A specific class of Bayesian networks called dynamic Bayesian networks have the added benefit that for each variable, or context, they add a time dimension. This type of model is an interesting proposition for modeling dynamic systems, like smart applications, whose context is dynamic. Some work has already been done on Bayesian networks that proposes the use of dynamic Bayesian networks for commerce opportunities [13]. Further research is needed to see how this approach works in a loginless application whose goal is not just commercial but also to maximize user utility.

VI. Challenges and Next Steps

As software engineers we have become used to the notion of logins and profiles. When designing systems we take for granted that the person using the application will create an account or fill out a profile. Consequently we build applications that make assumptions about the existence of predefined pieces of context. One of the challenges moving forward will be in realizing that we must at least offer a solution that requires less, not more, work from our users.

Another challenge is in dealing with the complexity found as a result of the flood of context. We briefly identified a couple of potential solutions, quad tree and dynamic Bayesian networks, as potential candidates for helping to deal with this complexity but further research is necessary. Just like in self-adaptive systems there will be no final or complete solution. As new contexts and expectations emerge, new data structures and models will be added to the bag of tricks. One challenge will be in identifying which tricks work the best for which context and how to architect systems so that the data structures and models are manageable.

One last challenge is in dealing with scalability. This is not a new problem nor is it unique to smart applications; however, smart applications amplify the issues around scalability because of the volume of users involved and the sheer number of variables to consider. One solution, which has traditionally been employed for self-adaptive systems, is the use of sophisticated schedulers for managing the back end services so that quality of service (QoS) is maintained and cost is kept low. This approach has merit with smart applications but another factor to consider is the power found in mobile devices. How can we leverage mobile devices to improve the user experience and at the same time cut costs?

Moving forward we need to tackle some of these challenges. First we plan to identify a set of contexts that is relevant to people's daily routines as well as to commerce. We then aim to further explore how dynamic Bayesian networks can be leveraged to provide relevant recommendations to match the user needs.

References

[1] D. P. Truex, R. Baskerville, and H. Klein, "Growing systems in emergent organizations," *Communications of the ACM*, vol. 42, no. 8, pp. 117–123, Aug. 1999.

[2] Northrop et al., *Ultra-large-scale systems : the software challenge of the future*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

[3] IBM Corp., "An architectural blueprint for autonomic computing," IBM Technical Report, 2006.

[4] Lemos et al., "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*, ser. Lecture Notes in Computer Science, R. Lemos, H. Giese, H. Müller, and M. Shaw, Eds. Springer Berlin Heidelberg, 2013, vol. 7475, pp. 1–32.

[5] Cheng et al., "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science, B. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Springer Berlin Heidelberg, 2009, vol. 5525, pp. 1–26.

[6] CISCO, "Cisco visual networking index: Global mobile data traffic forecast update, 20112016," CISCO, Tech. Rep., 04 2012.

[7] Gartner, "Gartner identifies organizational implications of the rise of mobile devices," *M2 Presswire*, 06-11 2012.

[8] ——, "Gartner says the personal cloud is poised to eclipse the pc as the hub of consumers' digital lives by 2014; key issues facing cloud computing to be examined at gartners infrastructure, operations and data centre summit, may 14-15, in mumbai," *M2 Presswire*, 05-08 2012.

[9] ——, "Gartner says that consumers will store more than a third of their digital content in the cloud by 2016," *M2 Presswire*, 06-25 2012.

[10] R. J. Desmarais, P. Lach, and H. A. Müller, "Yakit: a locality based messaging system using icon overlay," in *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, ser. CASCON '11. Riverton, NJ, USA: IBM Corp., 2011, pp. 148–159.

[11] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[12] F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*, 2nd ed. New York: Springer, 2007.

[13] S. Lee and K. C. Lee, "Context-prediction performance by a dynamic bayesian network: Emphasis on location prediction in ubiquitous decision support environment," *Expert Systems with Applications*, vol. 39, no. 5, pp. 4908–4914, 2012.