Software Selection and Configuration in Mobile Environments: A Utility-Based Approach

Vahe Poladian
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+ 1 412 362 9015
Vahe.Poladian@cs.cmu.edu

David Garlan
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+ 1 412 268 5056
Garlan@cs.cmu.edu

Mary Shaw
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+ 1 412 268 2589
Mary.Shaw@cs.cmu.edu

ABSTRACT

Users of low-power mobile computing platforms make ad hoc decisions when choosing software components among alternatives and configuring those components. We propose applying utility-theoretic models, which can help determine optimal allocation of scarce resources to applications given the user's utility and application resource usage. We believe that taking into consideration resource consumption and applying microeconomic models has the potential of improving the user's satisfaction with the system. In this paper, we formulate the problem, demonstrate the use of a microeconomics-based model on a simple version of the problem, and list possible solutions. Further, we identify issues typical of mobile environments that are not addressed by existing research, and propose ways of tackling these issues.

1. MOTIVATION

Mobile computer users are often faced with choosing among software components that provide similar services at various levels of quality (e.g., which map rendering program to use) and configuring those components (e.g., what portion of the map database to download). When making a selection among alternatives, users usually consider the computational features supported by similar applications and perhaps the dollar cost, but they typically ignore the differences among the resource requirements of the applications (e.g., a feature-rich application is likely to use more memory than a light-weight version). Mobile computers generally have limited resources (e.g., memory, disk bandwidth, battery capacity) as compared to desktop computers, so ignoring these resources can lead to substantially less useful performance than might be achieved.

Most users typically do not apply a systematic method to the configuration decision, and they often ignore, even informally, the burden that an application places on the resources of the mobile computer. The resulting ad hoc decisions might differentiate relatively good solutions from bad ones, but we believe that only a systematic approach will consistently yield optimal or even very good solutions. Thus we explore the potential for utility-theoretic models to be used in the problem of mobile software selection and configuration.

Fourth International Workshop on Economics-Driven Software Engineering Research, May 21, 2002, Orlando, Florida.

2. THE PROBLEM

Let's formulate the problem. Given:

- A mobile computer system, equipped with scarce resources such as processing power, memory, network bandwidth, battery capacity,
- Configurable, fidelity-aware applications, which are capable of providing varying levels of quality in several dimensions of service (e.g., the Map Renderer might have the option to configure the size of its downloadable database, providing more detail and larger geographical coverage when more data is downloaded); and a choice between a unified application providing several services and several leaner, more specialized applications,
- Application profiles, which provide information about the various levels of quality in each service that an application provides,
- For each configuration of every application, resource usage information (e.g., the Map Renderer profile provides information about how much memory is needed in order to look up a certain number of locations),
- The preferences (utility) of a user with respect to all levels of quality for every desired service.[1]

[1] We assume that a mobile computer has only one user. Some of the related research has considered problems of mediating preferences of multiples users.

Determine: a configuration of the applications that provides the best overall utility to the user.

What we mean by fidelity-awareness is the ability of the application to provide varying degrees of quality and resource usage trade-offs. Furthermore, we expect an application to report the quality levels that it provides, as well as the resource requirements for each quality level. We assume that the application will conform to its advertised resource usage, i.e., it will not consume more resources than it should for each level of quality.

The space of quality levels, and hence the corresponding resource usage, can be either fine-grain (e.g., the granularity of the downloadable map database can be in bytes, and with each additional one hundred bytes, the number of locations that can be looked up increases by one), or coarse-grain (e.g., quality levels of low, medium, high of the e-mail application features, where low allows no attachments, medium selectively allows some attachments, and high allows all attachments to be downloaded).

It is possible that various degrees of quality of essentially the same service are provided by more than one application. For example, a map application from one vendor might have a larger database, thus supporting better detail, as compared to a similar application from another vendor. In this case, several quality vs. resource trade-offs are available by the virtue of having different applications, hence choosing the best application amounts to selecting the best quality vs. resource trade-off. We can see that application configuration and application selection problems are intimately related.

3. RELATED WORK

The Amaranth project [1], [5], [8] at Carnegie Mellon University has developed a Quality of Service based Resource Allocation Model (Q-RAM), which enables a computer system to optimally allocate its resources to maximize the system's utility. Our objective and approach are similar, but we address the mobile computing environment. Later we consider issues specific to mobile computers that we believe are not solved by previous research.

The mobile systems group [3], [6] at University College London uses closed-bid Vickrey auctions to solve the problem of allocating bandwidth and other resources to multiple competing users in a mobile environment. This approach aims to mediate the preferences of *multiple* users.

The Nemesis operating system [7] uses a decentralized approach in determining the optimal resource allocation. Processes reveal their preference for resources to the kernel. Using *shadow* prices, the system charges the consumer of a resource a price, which is the cost that other potential consumers of the resource incur by forgoing that resource. These charges are viewed as continuous feedback signals that allow the system to competitively adapt to a state that is optimal from the point of view of all consumers.

Under the umbrella of Project Aura [1], [9] at Carnegie Mellon University, various groups investigate issues related to capturing user intent and providing for a distraction-free computing environment. In the architecture of Aura, the Environment Manager is the component that coordinates applications and configures them for use. We propose implementing utility theoretic models within Aura's Environment Manager.

4. APPROACH

We are exploring the use of a microeconomic model that leverages knowledge of user's preferences, application quality of service vs. resource trade-offs, and resource constraints, to compute an optimal allocation of scarce computing resources among competing applications. The objective is to configure applications so as to maximize the user's utility.

Using an example, we show the economic intuition behind the model. Consider a rather simplified model of computation, consisting of two simple applications: a map rendering application and an e-mail client, which can work in a *disconnected* mode. Assume, for simplicity, that the only resources under consideration are the battery of the device and its memory, e.g., flash memory. The Map Renderer provides directions and maps using a variable size downloadable database. The user can download only a portion of the entire database at a time. The bigger the database, the larger the region covered, and on average, the more locations that can be looked up. The e-mail application uses local memory to store messages for offline processing. The larger the amount of memory allocated to e-mail, the more messages that can be downloaded, read, and composed. The total memory of the mobile computing device imposes one constraint.

The second constraint is imposed by the battery capacity, which is roughly proportional to the length of the time that the device can be used without re-charging. Naturally, the limited lifetime of the battery has to be allotted between the two applications.

There are two configuration decisions to make: (1) how much memory to allocate to each application, and (2) how much energy (and thus, how much time) to allocate to using each application. For simplicity, we ignore the different policies that each application might use internally to manage memory and battery life.

Treat the Map Renderer and e-mail applications as producers of map and e-mail services. As the unit of service provided by the Map Renderer application we consider the number of physical locations that are looked up using the application. Looking up one location requires little memory

and perhaps a few minutes of battery life. Looking up twenty locations requires proportionally more memory and longer battery usage. Similar analysis applies to the e-mail application, but with respect to the number e-mails that are read, responded to, or composed.[2]

The graph in Figure 1 introduces a visualization of the problem and the solution. The vertical axis shows the units of map service, and the horizontal axis, the units of e-mail service. Ignoring for a moment the battery constraint, we identify all the possible combinations of units of map and e-mail service that can be achieved given the memory constraint.

Because of the linearity assumption, this constraint is a straight line, describing all the linear combinations of units of map and e-mail service that use up at most the total memory available.

Similarly, we express the battery constraint (independent of the memory constraint) as another line, which can possibly have a different slope and intercepts.

Notice that when the maximum number of map lookups is done, the memory of the device is entirely used up by the map data, but the battery is not yet drained. Similarly, when the maximum number of e-mail is processed, the battery is exhausted, but there is available memory.

Thus that the map application is more memory-intensive, and the e-mail application is more processor-intensive.

The region that is bounded above by the two bold lines shows all the possible combinations of map lookups and e-mails processed under the memory and battery constraints.

But which is the optimal point?

In order to determine that point, and the optimal resource allocation, we need to consider the utility of the user. As is common in microeconomics, let us consider the contours of equal utility, also called the "indifference curves". Each indifference curve is made up of points that from the point of view of the user yield equal utility.

The higher the curve, the better the utility. Typical shapes of indifference curves are described by equations such as $M^{\alpha} \times E^{(1-\alpha)} = \texttt{const}$, where α is a real number between 0 and 1. In the graph we have one set of such curves, which indicate some utility bias towards the e-mail application.

Although it may seem that the optimal solution should lie at the intersection of the resource constraint lines – where both resources are saturated – in fact it does not. In this case, the optimal solution occurs at a point where one of the indifference curves is tangent to the energy constraint line.

Indeed, those curves that are higher (further away from the origin) are outside of the resource boundaries. And curves that are lower yield lower utility.

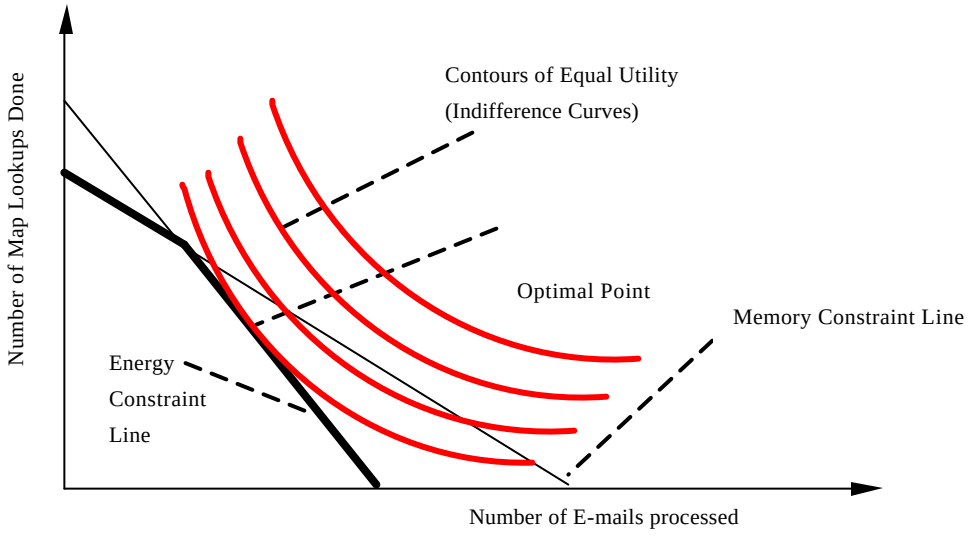

Figure 1: The Optimal Point Yields the Highest Utility While Satisfying the Resource Constraints

[2] We assume, for simplicity, that the level of service is linear in terms of each resource.

Economists use the concept of *marginal utility* to analyze the properties of an optimal solution. Marginal utility of e-mail service with respect to a resource is the incremental utility to the user, when a small amount of that resource is shifted to the e-mail application. Notice that at the optimal point, the marginal utility of e-mail service with respect to energy is equal to that of the map service (indicating that the indifference curve is tangent to the energy constraint line). This insight helps in generalizing the solution.

This approach is also applicable in selecting one application among several alternatives. Simply treat the different providers of the same service as if they were different quality of service vs. resource trade-off opportunities provided by a single application. Apply the same analysis to determine the optimal point. The application that offers the latter combination is the best alternative.

5. SOLUTIONS TO THE PROBLEM

The example just discussed makes several assumptions that may not be realistic in a computing environment, e.g., the quality space is continuous, that the mapping of quality space to the resource space is linear, that the utility is of particular functional form. These assumptions help us in understanding the economic intuition, but may need to be dropped in order to be applicable in the computer systems.

The general problem described in section 2 is similar to the example we discussed, but allows for a larger number of applications and resources. Also, in the general case we drop many of the simplifying assumptions of the example. Classical economics solves the utility maximization problem using calculus. Assuming that the utility functions with respect to service level, and service level with respect to resource are all continuously differentiable and monotonic, then the optimal point has the property that the partial derivatives of the utility of each service with respect to a given resource are either all pair-wise equal, or some of those derivatives are pair-wise equal, and the rest are all zero[3].

In computer systems, it is not realistic to assume that the quality space is continuous, and it may not be practical to make any assumptions about the properties of either quality or utility functions. For example, more resource may not consistently yield higher quality, and higher quality may not yield higher utility. However, we believe that there is value in studying the forms of utility functions that occur in mobile applications, as this can offer valuable insight to understanding the preferences of mobile users, and help in solving the optimization problem (e.g., as we showed above the optimal solution may not be the point where all the resources are saturated). Continuous models may provide reasonable approximations of fine-grained quality spaces.

Lee [5] and Khan [4], in their approaches to the quality of service management, assume that the space of quality levels is a finite (discrete) set. Such an assumption is realistic, as applications typically offer only a finite set of quality vs. resource trade-offs. Given this assumption, the utility maximization problem can be formulated as a variation of a multi-dimensional knapsack problem (another common name for this set of problems is integer programming). It is known that these problems are NP-hard. The optimal solutions are exponential in the size of the input, but there are polynomial approximation algorithms that can find solutions within a threshold of the optimal. These models may provide reasonable approximations of coarse-grained quality spaces.

6. ISSUES SPECIFIC TO MOBILE COMPUTING ENVIRONMENTS

We believe that mobile computing presents interesting problems that have not yet been addressed by previous research: (1) solving implementation issues specifically related to mobile systems, (2) describing the forms of utility functions that occur in mobile computing, and (3) dynamically reconfiguring the system after a change in the environment, resources, or utility.

Let us discuss the latter group of issues in more detail, as it exposes several interesting research problems. Mobile computing environments change often. Such changes can be in resource availability (e.g., network bandwidth dropped, so perhaps a web image download should be deferred), in the environment (e.g., it is darker in one room than another, so the display needs to be brightened), in the utility of the user (e.g., after an urgent message regarding a shift in deadline, the utility of finishing the demo goes up, thus necessitating more resources for the demo).

In response to all these changes, either the constraint or the utility or both changes, so the optimal point moves and the system needs to be reconfigured. We believe that current literature does not directly address this problem. The computation of a new optimal value may not be the best action in response to a change. To see this, imagine an increase in some resource that enables a better version of some application (e.g., that offers higher quality in some service) to run. Shutting down the currently running version and starting the better version incurs overhead cost – hence that configuration may not yield higher utility than the current one. As another example, imagine a decrease in some resource, which necessitates that an application switch to a lower quality level. However, this might cause unnecessary user distraction, and may not be optimal from

[3] Consider the change in total utility when shifting a slight amount of a resource from one service to another. If there are corner cases, then we can assume functions are piece-wise differentiable; however, the corner points need to be checked separately in case they are optimal.

the user's point of view. Accounting for hidden costs such as user distraction becomes an important problem in the reconfiguration, which was not an issue for the static version of the problem. Likewise, the cost of an incremental change to an existing configuration is not addressed by the simple model.

Both the economic model described here and the solutions that we have found in the computing literature solve the static version of the problem. The aforementioned issues comprise only a small sub-set of all those that we have considered.

We believe that it is possible to recast the problem of reconfiguration into utility models (i.e., to an integer programming problem) with the introduction of additional variables to account for overhead resource usage, costs of incremental change, hidden costs accounting for user distraction, etc. Such a reformulation might allow us to solve the problem of reconfiguration using existing solutions. However, existing solutions may not be computationally efficient enough for mobile computers. Ideally, we would like to reuse the computation of an optimal solution for an existing configuration in the process of computing the new optimum. This will make such solutions attractive for use in mobile environments, since computation is relatively expensive on mobile devices, and the changes are frequent. For some variations of integer programming there are solutions that are incremental. For example, in case of only one resource, there is a dynamic programming solution. We would like to explore the integer programming, linear programming, and other optimizations research literature for such solutions and algorithms that are well suited for incremental change.

7. CONCLUSION

We have proposed using utility-theoretic models to determine user-optimal software configurations in mobile computing environments. Using an example, we have demonstrated how utility models can be applied to determine optimal solutions. We then formulated the generalized problem of mobile component selection and configuration, and mentioned approaches that promise to solve the problem. We also enumerated various issues that are characteristic of mobile environments that we believe are not addressed in current research. We propose approaches for tackling these issues, while reusing existing models.

8. ACKNOWLEDGMENTS

This research is supported by the National Science Foundation under Grant CCR-0086003, by the Department of Defense under DARPA grants F30602-00-2-0616 and N66001-99-2-8, by the High Dependability Computing Program from NASA Ames cooperative agreement NCC-2-1298, and by the Software Industry Center at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, DOD, or the U.S. government.

The authors would like to thank Joao Pedro Sousa, Orna Raz, and Dushyanth Narayanan of Carnegie Mellon School of Computer Science for critical feedback on this paper.

9. REFERENCES

[1] The Amaranth Project at Carnegie Mellon University. http://www.cs.cmu.edu/afs/cs/project/ices-amaranth/www/.

[2] The Aura Project at Carnegie Mellon University. http://www.cs.cmu.edu/~aura.

[3] L. Capra, W. Emmerich, and C. Mascolo. A Micro-Economic Approach to Conflict Resolution in Mobile Computing. *UCL-CS Research Note RN/01/38.*

[4] S. Khan. *Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture.* PhD Dissertation, Department of Electrical and Computer Engineering, University of Victoria, 1998.

[5] C. Lee. On Quality of Service Management. PhD Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1999.

[6] The Mobile Systems Group at the University College London. http://www.cs.ucl.ac.uk/research/mobile/.

[7] R. Neugebauer and D. McAuley. Congestion Prices as Feedback Signals: An Approach to QoS Management. *Proceedings of the 9th ACM SIGOPS European Workshop*, September 2000.

[8] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A Resource Allocation Model for QoS Management. *Proceedings of the IEEE Real-Time Systems Symposium*, December 1997.

[9] J.P. Sousa and D. Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. *Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture*, Montreal, August 2002. To appear.