Model-Driven Architectures for Optimizing Mobile Application Performance

Chris Thompson, Jules White, Brian Dougherty, Hamilton Turner, Scott Campbell,
Krzysztof Zienkiewicz, and Douglas C. Schmidt

Department of Electrical Engineering and Computer Science,
Vanderbilt University, Nashville, TN USA
{jules, briand, schmidt}@dre.vanderbilt.edu & {chris.m.thompson, hamilton.a.turner, scott.h.campbell,
krzysztof.k.zienkiewicz}@vanderbilt.edu

Section 1. Introduction

Emerging trends and challenges. Mobile devices, such as smartphones, mobile internet devices and web-enabled media players, are becoming pervasive. These devices possess limited resources, which motivates resource optimizations, such as memory footprint minimization and battery usage minimizations. Mobile application developers must therefore understand the tradeoffs between performance and battery life. In conventional mobile device systems, however, it is hard to predict the effects of these optimizations until systems have been completely implemented, which makes it hard to test power consumption and performance until late in the software lifecycle, e.g., during implementation and testing. Changes made at this point usually result in far-reaching consequences to the overall design of the system and cost much more compared to those made during earlier software lifecycle phases, e.g., during architectural design and analysis.

Conventional techniques for developing mobile device software are not well-suited to identifying mistakes during earlier phases of the software lifecycle since it is hard to compare the power consumption of one architectural design vs. another without implementing and testing each device application. Moreover, for each function an application performs, there are a number of possible architectural designs for accomplishing the same task, each differing in terms of operational speed, battery consumption and accuracy. These architectural design decisions can have significant consequences for overall device performance and other applications contending for system resources. It is therefore essential that developers of mobile devices understand the implications of every architectural choice on power consumption and performance.

For example, if a mobile application communicates with a server it can do so via several protocols, such as HTTP, HTTPS, or other socket connections. Developers can also elect to have the application and/or mobile device infrastructure submit data immediately or in a batch at periodic intervals. Each design option can result in differing power consumption profiles. The combination of these options results in too many possible variations to implement and test each one within a reasonable budget and production cycle.

Solution approach → *Emulation of application behavior through model-driven testing and auto-generated code*. Model-Driven Architectures (MDAs) provide a potential solution to the challenges described above. MDA relies on system-independent modeling languages, such as UML or domain-specific modeling languages (DSMLs), to visually represent various aspects of application and system design. These models can then be used to generate code and analyze performance. By creating a model of candidate solution architectures, instrumented architectural emulation code can be generated and run on actual mobile devices which enables developers to

quickly emulate a multitude of possible configurations and provides them with actual device performance data without investing the time and effort manually writing application code.

The generated code emulates the modeled architecture by consuming sensor data, computational cycles, and memory as specified in the model, as well as transmitting/receiving faux data over the network. Since wireless transmissions consume most of the power on mobile devices [3] and network interaction is a key performance bottleneck, large-scale power consumption and performance trends can be gleaned by executing the emulation code. Moreover, as the real implementation is built, the actual application logic can be used to replace faux resource consuming code blocks to refine the accuracy of the model. This model-driven solution has been utilized previously to eliminate some inherent flaws with serialized phasing in layered systems, specifically as they apply to system QoS and to identify design flaws early in the software production lifecycle [9]. Some prior work [8] also employs model-driven analysis to conduct what-if analysis on potential application architectures.

By utilizing model-driven analysis, mobile software developers can quantitatively evaluate key performance and power consumption characteristics earlier in the software lifecycle (e.g., at design time) rather than later (e.g., during and after implementation), thereby significantly reducing software refactoring costs due to design flaws. Moreover, since emulation code is automatically generated from the model, developers can quickly understand key performance and power consumption characteristics of potential solution architectures without investing the time and effort to implement them.

Outline of the Rest of the Proposed Chapter

Section 2: Motivating Example. Managing system resources properly can significantly affect device performance and battery life. For instance, reducing CPU instructions not only speeds performance but also reduces the time the process is in a non-idle state; reducing network traffic also speeds performance and reduces the power supplied to the radio. To demonstrate the importance of proper resource management and the value of model-based resource analysis, this section describes a motivating example of a mobile application, called *Wreck Watch*.

Wreck Watch runs on Google Android smartphones to detect car accidents by analyzing data from the device's GPS receiver and accelerometer and looking for sudden acceleration events from a high velocity indicative of a collision. Car accident data is then posted to an HTTP server where it can be retrieved by other devices in the area to help alleviate congestion, notify first responders, and provide accident photos to an emergency response center. Users of Wreck Watch can also elect to have certain people contacted in the event of an accident. Figure 1 shows this behavior of Wreck Watch.

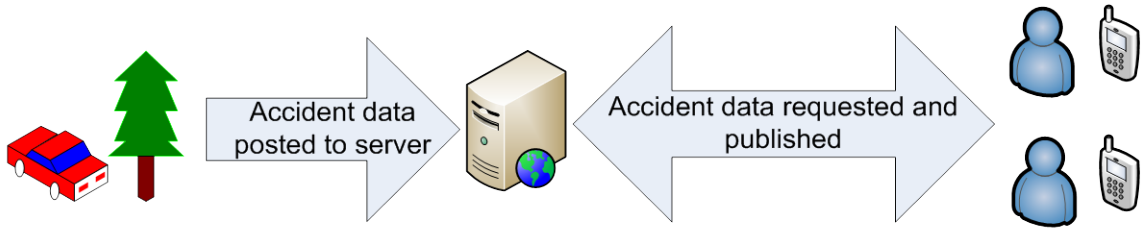

Figure 1: Wreck Watch Behavior

The Wreck Watch application must maintain conservative power consumption. The application needs to run at all times, will consume a great deal of sensor information to accurately detect wrecks, and frequently utilizes network connections. If not designed properly, therefore, these characteristics could result in a substantial decrease in battery life. In the case of Wi-Fi for instance, the radio consumes nearly 70% of device power [2] and in extreme cases can consume 100 times the power of one CPU instruction to transmit one byte of data [3].

The amount of power consumed by the network adapter is generally proportional to the amount of information transmitted [1]. The framing and overhead associated with each protocol can therefore significantly affect the power consumption of the network adapter. Prior work [5] demonstrated that significant power savings could be achieved by modifying the MAC layer to minimize collisions and maximize time spent in the idle state. This work also recognized that network operations generally involved only the CPU and transceiver and by reducing client-side processing, they could substantially reduce power consumed by network transactions. Similarly, other work [7] demonstrated that such power savings could also be achieved through transport layer modifications.

Although MAC and transport layer modifications are typically beyond the scope of most software projects, especially mobile application development, the data transmitted on the network can be optimized so it is as lightweight as possible, thereby accomplishing, on a much smaller scale, some of the same effects. The remainder of this chapter will use the Wreck Watch application to showcase key design challenges that developers face when building power-aware applications for mobile devices.

Section 3: Design and Behavioral Challenges of Mobile Application Development. Despite the ease with which mobile applications can be developed via advanced SDKs (such as Google Android and Apple iPhone) developers still face many challenges related to power consumption. If developers do not fully understand the implications of their designs, they can substantially reduce device performance. Battery life represents a major metric used to compare devices and can be influenced significantly by design decisions. This section describes how designing mobile applications that are cognizant of battery performance presents the following challenges to developers:

Challenge 1: Accurately predicting battery consumption of arbitrary architectural decisions is hard. Each instruction executed can result in the consumption of an unknown amount of battery power. Accurately predicting the power consumed for each line of code is hard given the level of abstraction present in modern SDKs, as well as the complexity and numerous variations between physical devices. Moreover, disregarding language commonalities between completely unrelated devices, mobile platforms, such as Android, are designed to operate on a plethora of hardware configurations, which may affect the perceived effects of different architectures.

Challenge 2: Trade-offs between performance and battery life are not readily apparent. Although performance and power consumption are generally design tradeoffs, the actual relationship between the two metrics is not readily apparent. For example, when comparing two networking protocols, plain HTTP and SOAP, plain HTTP might operate much faster requiring only 10 ms to transmit the data SOAP requires 50 ms to transmit. At the same time, HTTP might consume .5 mW, while SOAP consumes 1.5 mW. Without the context of actual performance in a physical device it would be difficult to predict the overhead associated with SOAP. Moreover, this data may vary from one device to the next.

Challenge 3: Effects of transmission medium on power consumed are largely device, application, and environment specific. Wireless radios consume a substantial amount of device power relative to other mobile-device components [6], where the power consumed is directly proportional to the amount of information transmitted [1]. Each radio also provides differing data rates, as well as power consumption characteristics. Depending on the application, developers must choose the connection medium best suited to application requirements, such as medium availability and transmission rate. The differences between transmission media are generally subtle and may even depend on environmental factors [10], such as network congestion and signal quality that are impossible to accurately predict. To reliably and accurately quantify performance, therefore, testing must be performed in environmentally-accurate situations.

Challenge 4: It is hard to accurately predict the effects of reducing sensor data consumption rates on power utilization. To provide the most accurate readings and results, device sensors would be polled as frequently as they sample data. This method consumes the most possible power, however, by not only requiring that the sensor be enabled constantly, but by also increasing the amount of data the device must process. In turn, reducing the time that the sensor is active significantly reduces the effectiveness and accuracy of the readings. Determining the exact amount of power saved by a reduction in polling rate or other sensor accuracy change is difficult.

Challenge 5: Accurately assessing effects of different communication protocols on performance is impossible without real-world analysis. Each communication protocol has a specific overhead associated with it that directly affects its overall throughput. The natural choice would be to select the protocol with the lowest overhead. While this decision yields the highest performance, it also results in a tightly coupled architecture and substantially increases production time. That protocol would only be useful for the specific data set for which it was designed, in contrast to a standardized protocol, such as HTTP. Standardized protocols often support features that are unnecessary for many mobile applications, however, making the additional data required for HTTP transactions completely useless. It is challenging to predict how much of a tradeoff in performance is required to select the more extensible protocol.

Section 4: Model-based Testing and Performance Analysis. This section addresses the challenges posited in Section 2 by presenting MDA techniques that can be used to visually model a prospective system, generate instrumented architectural emulation code for multiple platforms, simulate normal device usage, and generate performance analysis reports that can be utilized to streamline architecture optimization. In particular, this section will:

- Demonstrate DSML construction for the Wreck Watch application
- Construct models representing possible configurations of the Wreck Watch application
- Generate instrumented code for multiple mobile device platforms
- Perform performance analysis and optimization based on reports generated by test code

These key elements of model-driven analysis will provide readers with an understanding of the processes and practices associated with creating a model that can be utilized to perform performance analysis. We will present a step-by-step analysis of the modeling process and finally, provide some cursory analysis of a sample configuration. We will also attempt to highlight potential points of variability with a special focus on power-intensive operations and performance bottlenecks.

Section 5: Case Study: Applying MDA to Optimize Wreck Watch Application Power Consumption. This section shows how the MDA techniques presented in Section 4 can be applied to

the Wreck Watch application presented in Section 2. We will show several potential system architectures with special attention paid to the points of dissimilarity (such as Wi-Fi versus EDGE and HTTP versus custom protocols) and potential for performance variation. These models will be used to generate sample code that we will benchmark on Android devices with a focus on the methodology rather than the resultant data. This section also highlights the specific advantages to model-based analysis of model-driven architectures and the speed at which a multitude of configurations can be tested accurately.

Section 6: Concluding Remarks. This section presents lessons learned and concluding remarks.

Relevance to the Book

This chapter outlines some of the benefits and applications of MDAs to mobile application development with special attention paid to the role of using modeling in performance analysis and resources utilization optimization. This chapter will provide readers with a firm understanding of DSML construction and model-based system analysis for mobile device power consumption and performance. This chapter will also emphasize the platform-independent view of the OMG MDA and utilize custom DSMLs to generate platform-specific emulation code.

References

1. Feeney, L. & Nilsson, M., "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," *IEEE INFOCOM*, **2001**, *3*, 1548-1557
2. Liu, T.; Sadler, C.; Zhang, P. & Martonosi, M., "Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet" *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, **2004**, 256-269
3. Pering, T.; Agarwal, Y.; Gupta, R. & Want, R., "Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces," *Proceedings of the Annual ACM/USENIX International Conference on Mobile Systems, Applications and Services (MobiSys)*, **2006**
4. Poole, J., "Model-driven architecture: Vision, standards and emerging technologies," *Workshop on Metamodeling and Adaptive Object Models, ECOOP*, **2001**
5. Chen, J.; Sivalingam, K.; Agrawal, P. & Kishore, S., "A comparison of mac protocols for wireless local networks based on battery power consumption," *IEEE INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, , 1*
6. Krashinsky, R. & Balakrishnan, H., "Minimizing energy for wireless web access with bounded slowdown," *Wireless Networks, Springer*, **2005**, *11*, 135-148
7. Kravets, R. & Krishnan, P., "Application-driven power management for mobile communication," *Wireless Networks, Springer*, **2000**, *6*, 263-277
8. Paunov, S.; Hill, J.; Schmidt, D.; Baker, S. & Slaby, J., "Domain-specific modeling languages for configuring and evaluating enterprise DRE system quality of service," *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on*, **2006**, 10
9. Hill, J.; Tambe, S. & Gokhale, A., "Model-driven engineering for development-time QoS validation of component-based software systems," *Proceeding of International Conference on Engineering of Component Based Systems*, **2007**

10. Carvalho, M.; Margi, C.; Obraczka, K. & others, "Garcia-Luna-Aceves. Modeling energy consumption in single-hop IEEE 802.11 ad hoc networks," *Thirteenth International Conference on Computer Communications and Networks (ICCCN'04*, **2004**, 367-37

Author Biographies

Chris Thompson is an undergraduate Research Assistant at Vanderbilt University. He is currently pursuing a Bachelor of Engineering degree in Computer Engineering. He is currently focusing on mobile device performance analysis and in the past has worked with constraint-based deployment configuration and automating hardware and software evolution analysis.

Dr. Jules White is a Research Assistant Professor at Vanderbilt University. He received his BA in Computer Science from Brown University, his MS in Computer Science from Vanderbilt University, and his Ph.D. in Computer Science from Vanderbilt University. Dr. White's research focuses on applying a combination of model-driven engineering and constraint-based optimization techniques to the deployment and configuration of complex software systems. Dr. White is the project leader for the Generic Eclipse Modeling System (GEMS), an Eclipse Foundation project.

Brian Dougherty is a Ph.D candidate in Computer Science at Vanderbilt University. Brian's research focuses on hardware/software co-design, heuristic constraint-based deployment algorithms, and design space exploration. He is the co-leader of the ASCENT project, a tool for analyzing hardware/software co-design solution spaces. Brian is also a developer for the Generic Eclipse Modeling System (GEMS). He received his B.S. in Computer Science from Centre College, Danville, KY in 2007.

Hamilton Turner is a Research Assistant at Vanderbilt University. He has worked with the Institute for Software Integrated System (ISIS) for two years, and currently focuses on Mobile Computing. He is working on his B.E. in Computer Engineering from Vanderbilt University, and has previously focused on Unit Testing Non-Functional Properties of Large Scale Computing Systems.

Scott Campbell is a Graduate Research Assistant at Vanderbilt University. He has worked with the Institute for Software Integrated System (ISIS) for two years, and currently focuses on Mobile Computing. He received his BS in Computer Science from Vanderbilt University, and has previously focused on remote collaboration in the Generic Eclipse Modeling System (GEMS).

Krzysztof Zienkiewicz is an Undergraduate Research Assistant at Vanderbilt University. He works for the Institute for Software Integrated Systems (ISIS), focusing on application development on mobile devices. He is currently working on his B.S. in Computer Science and Mathematics from Vanderbilt's School of Engineering and School of Arts and Science respectively.

Dr. Douglas C. Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 400 papers that cover a range of topics, including patterns, optimization techniques, and empirical analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications. Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are open-source middleware frameworks and model-driven tools that implement patterns and product-line architectures for high-performance DRE systems.