

RAPID and HTML5's Potential

David Torosyan
Jet Propulsion Laboratory, California Institute of Technology
August 21, 2012

Introduction

Just as important as the engineering that goes into building a robot is the method of interaction, or how human users will use the machine. As part of the Human-System Interactions group (Conductor) at JPL, I explored using a web interface to interact with ATHLETE, a prototype lunar rover. I investigated the usefulness of HTML 5 and Javascript as a telemetry viewer as well as the feasibility of having a rover communicate with a web server. To test my ideas I built a mobile-compatible website and designed primarily for an Android tablet. The website took input from ATHLETE engineers, and upon its completion I conducted a user test to assess its effectiveness.

Background

The Conductor group is part of the Operations and Planning Software laboratory (OPS lab), which develops and supports software for mission operations and research technologies. Current projects include the Mars Science Laboratory science operations interface (MSLICE) for Curiosity, the All-Terrain Hex-Limbed Extra-Terrestrial Explorer (ATHLETE) meant for potential moon/asteroid missions, and research into a natural user interface (NUI) for human-robot interaction using Microsoft Kinect, a low-cost sensor with built-in gesture detection. An emphasis of Conductor research is designing control systems that require little training and set-up.

ATHLETE itself rises 15 feet high, stretches 20 feet from one face to the other, and is shaped something like a metal spider. As such, the it spends most of its time in a building dedicated to it on lab, but goes out for drives in the Arroyo Seco and is taken out for field testing in the Mojave desert. The rover was initially developed for transporting lunar cargo on the surface of the moon, but due to changing plans at NASA is being considered for a future asteroid exploration and sample return mission.

ATHLETE is equipped with its own flight software, but in an attempt to standardize robot communications JPL is rolling out the Robot Application Programming Interface Delegate (RAPID), a standard of communication between robot and operator. RAPID works on top of multicast software from Real-Time Innovations (RTI) that allows telemetry from the rover to be communicated easily to whatever system needs to listen. ATHLETE's flight software produces telemetry that a "rapid bridge" translates into RAPID-readable telemetry. In order to receive said messages, a correctly configured computer is required, one where both RTI software and the RAPID workbench are installed. Furthermore, this software cannot be installed on some operating systems and is not available for any mobile device.

Objectives

The lack of mobile support and a lengthy installation process is a deterrent to the growth of RAPID. Limiting support for desktop and laptop environments means limiting the usefulness of the platform. A working example of RAPID on a mobile device can open opportunities for its development. To this end, my project aimed to build an application compatible with tablets, and, if possible, with smartphones as well. The only way to build an application for a wide range of devices is a website, so a successful project would also explore the capabilities of a web page in getting robot information across. Furthermore, a website means the ultimate in portability.

As to the actual focus of the application, there were three options dealing with ATHLETE. The first involved a two-way stream, both sending commands and receiving telemetry from the rover. This is advantageous because it would allow for the most full "driving" experience, but was eliminated for its scope being too large for a summer internship. The second option would be only commanding the rover, and the third only receiving telemetry. I chose the third option, pure telemetry, because of the security protocols that would be necessary with commanding the rover.

Approach

The first component of the puzzle was RAPID, and getting it to work with a server. My model required ATHLETE to publish telemetry to the server, which would then serve web pages detailing the

information. Because using RAPID requires RTI's Data-Distribution Service, some time was spent communicating with their technical support. In order to expedite the process, I attempted to take advantage of RTI's Web Integration software, but as it was still an early-release and needed work as a result, I was not successful. Eventually, however, I did manage to get both RTI and RAPID configured correctly and was able to send messages from one computer to another using the same pipeline the rover uses.

Eventually, the server is planned to be hosted by Amazon or the like. Until then, I used two computers, one for sending RAPID messages (and so acting as a virtual ATHLETE) and a second receiving and re-distributing those messages (taking the place of the server). The server itself is an Apache Tomcat server using a Java servlet to do much of the processing.

The actual web page was written in HTML and Javascript. HTML stands for Hyper-Text Markup Language and is what makes everything on the Internet. Its newest iteration, HTML 5, is full of new features for creating dynamic content natively. For example, there is a new element called "canvas" which essentially acts as a blank sheet of paper on top of the page. This allows the drawing of any shape, and, by redrawing many times a second, can be used for animation. This must be done in conjunction with Javascript, which is a scripting language just for websites. It allows the page to update elements by communicating with the aforementioned server. Finally, good practices with Cascading Style Sheets (CSS) were used to keep the page readable and remove unnecessary Javascript.

The design of the web page was created by a fellow intern, and was implemented by me in collaboration with Manuk Hovanesian. I broke up the telemetry viewers into "widgets", each a dynamic image drawn on a canvas. One important requirement was scalability; as tablets have a wide variety of screen sizes (and because most testing was on a desktop), every part of the web page was built to scale to any proportion.

Results

The final result of the page is a screen split in half: on the left side is a 3D model of ATHLETE that conveys position and orientation, and on the right a set of accordion-style tabs for in-depth telemetry. The first tab details velocity, and consists of a speedometer and a speed vs. time graph. The second shows torque distribution in the form of a radar graph; ATHLETE has six legs and many joints, so the

radar graph shows the six legs from above and the relative amounts of torque on each leg, with the joint displayed an option on the right. The next tab is a camera view with a scrollable set of preview images, all set on the border of a 3D circle. Finally, the leg view has its own set of tabs, one for each leg. There the user can view side-shot of each leg in its physical position and click on joints to get more information, such as torque, angle velocity, and temperature.

One important lesson from this experience is about hardware limitations. The 3D view, for example, requires WebGL to run smoothly, which is a feature only available on select browsers. While this may not be as much of a problem in a desktop environment, it means that tablet users are limited in their choice of browser. Furthermore, rendering complex objects requires a lot of power, regardless of browser. This makes the smartphone, a very convenient device, difficult to develop for.

After the website was built, we conducted some user tests with ATHLETE engineers. Two participants were designated three tasks each, one acting as a driver while the other was a safety observer, and then vice versa. The safety observer had the tablet in each case, and used it to monitor the safety of the rover in addition to keeping an eye on the physical machine. We learned a few things from the tests: one, that the page was not responsive enough to the touch. While the page transitions looked good, they found the slowness frustrating. Second, while I had focused on the visual aspects of telemetry viewing, the operators requested more numbers. Finally, while I worked to simplify the design to be straightforward, the reality is that I did not have full information about what was required to monitor that rover. That means the information density of the design was not quite high enough.

Discussion

The end result of the website roughly matched expectations, barring some technical challenges that proved insurmountable. One discrepancy between the expected and the reality was the nature of the user test; due to versioning problems, the application could not connect to the Athlete Simulator (athsim), let alone the actual rover. As a result, my computer served as a surrogate and I had to manually enter data in accordance with the driver's actions. Even still, the driver and safety observer went along well with the simulation and helped to make up the difference.

One challenge I did not expect was that of designing the interface. While the mock-up provided to me looked nice and the implementation was doable, there were many other considerations that needed to

be taken into account. For example, font that looks good on paper may look very different on a screen, and the same goes for color choices. Furthermore, there was the issue of reliable interaction; buttons had to be large enough to interact with, but not so large as to take up too much space. I believe that with more time and an opportunity to develop further, many of these problems could be solved.

Conclusion

I demonstrated that RAPID can be successfully implemented in a robot/server/user model, and that a mobile application has potential use in a field testing environment. Future work on the website itself would mainly be refinement and more testing, as well as the implementation of some features I did not have time for. Also important is the resolution of version differences in the ATHLETE code that prevents it from communicating with the website.

Acknowledgments

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by (name student program) and the National Aeronautics and Space Administration. I would like to thank Lucy Abramyan, my mentor, Manuk Hovanesian, who I worked with on this project, and his mentor, Jay Torres, who conceived the idea of a RAPID website. I'd also like to thank the Conductor team and the entire OPS Lab, as well as Chris McQuin and John Leichthy who helped test the application. Finally, I owe special thanks to JPL's education office and Caltech's SFP for this opportunity.