The Web of Things: Extending the Web into the Real World

Dave Raggett

W3C/ERCIM,
2004, route des Lucioles, Sophia Antipolis, 06410 Biot, France
email: `dsr@w3.org`
`http://www.w3.org/People/Raggett`

Abstract. Thanks to Moore's law the incremental cost of adding networking to devices is falling rapidly. This creates opportunities for many new kinds of applications. This paper looks at the potential of Web technologies for reducing the complexity for developing such applications, allowing millions of developers to extend the Web out of the browser and into the real world. This is achieved through mechanisms to provide web run times with access to rich models of users, devices, services, and the environment in which they reside. Privacy and trust are key considerations.

1 Introduction

The World Wide Web is today an essential global infrastructure with a myriad of roles, e.g. news, entertainment, commerce, and education, to name just a few. The Web's origin lay in hypertext, the means to link text-based documents. The Internet has given us the means to follow such links across servers all over the World, continent hopping at a single click. The Web rapidly evolved to include images, forms, and scripting, and soon attracted a new generation of developers, which now vastly outnumber the old school developers for traditional platforms like Microsoft Windows.

Today, the Web is a global platform for information-based applications, but that is about to change. What is driving this is the rapidly dwindling incremental cost of networking for all kinds of devices. This is a happy side-effect of Moore's law which describes the ongoing exponential improvements in integrated circuitry which by now has been happening for more than half a century. It is now easy to integrate radio-frequency components alongside digital circuitry for microcontrollers.

We are in the midst of a proliferation of devices that are largely invisible as they are embedded within everyday objects from toasters to cameras and cars. Microcontrollers are the fastest growing segment of the computer industry, with hundreds in every home. These devices are programmed to serve a single purpose, and today are mostly isolated. Networking them will allow many new kinds of application that add value in ways that the original manufacturer may not have envisaged.

Barcodes have been part of the everyday environment for many years, and used for tracking library books, identifying items at supermarket checkouts and routing luggage at airports. Two dimensional barcodes contain a lot more information and can be used to link physical objects to their virtual equivalents on the Web. Electronic versions of barcodes, RFID tags, are starting to take over, and appearing in passports, clothing and warehouses. The ability to scan multiple tags at one go, and to be able to write information back to the tag, provide obvious benefits over old fashioned barcodes.

The ability to connect applications running in the Web to devices in the real World has all kinds of benefits. Here are just a few:

- *Reduced energy costs* through management of home/office environments (sensor nets, actuators, ease of configuration, short/long term adaptation to human behaviour patterns)
- *Improved security* through remote monitoring of sensors and actuators (motion, thermal, doors, windows, locks, videocams, energy usage, RFID)
- *Reduced downtime* of household devices through continuous monitoring for signs of wear and proactive scheduling of repairs as part of maintenance contracts (washing machines, air conditioning, plumbing, cars, etc.)
- *Improved standard of care* for the elderly through monitoring patterns of activities as well as explicit health sensors, with the means to alert carers and emergency services

The remainder of this paper will look at some of the challenges in the way of realizing the Web of things, and proposals for how to address them.

2 Challenges

The biggest challenge is the difficulty in developing applications using traditional programming languages such as Java and C++. These involve steep learning curves for the language itself and especially for the complex application programming interfaces needed. This restricts the number of people who can develop applications as well as pushing up the costs.

Another challenge is the rapid pace of change for technologies used for networking devices. Some examples are Ethernet over twisted pair cable, DSL over copper phone lines, Ethernet over building power lines, WiFi and WiMax, BlueTooth, ZigBee, GSM and cellular packet radio, as well as interconnect solutions such as USB and Firewire. This multiplicity of technologies imposes complexity for developers, especially when different devices are using different solutions.

To add further difficulties, devices are likely to have a wide variety of ages, depending on their expected life time and frequency of replacement. Cell phones are replaced in a matter of months, whilst television sets and cars last for many years. The infrastructure built into buildings lasts even longer. How is it possible to create applications that work with a mix of generations of devices and technologies? How can we create applications today that will keep working when next year's devices come along?

Security is clearly important when it comes to controlling real world objects. Imagine a home security system: you wouldn't want a hacker to be able to open your front door and walk right in and take your prized possessions! Security firewalls protect against unwanted access, but also make it hard to configure applications. Few users are prepared to configure their browsers let alone opening ports to allow applications to connect through their firewalls. Usability is a critical factor in ensuring effective security.

If web applications are able to access sensors throughout the environment in public and private spaces, this would enable companies to track your life in immense detail. This calls for strong safeguards on privacy with negotiable obligations on how long personal data is held for, and for what purposes it can be used. You should be provided with a means to access any personal data held on you, to update it and to request its deletion. This means that privacy is not just a bolt on feature, and instead needs to treated as a core component with strong requirements on server middleware.

A final challenge is enabling applications to dynamically adapt to the context which includes user preferences, device capabilities, and the environment in which the application runs. This is especially important for people with sensory or cognitive impairments.

3 Web Run-Time

A Web run-time (WRT) is a component that provides an execution environment for markup languages, style sheets, scripts and related resources. A browser can be implemented as a container for a Web run-time that adds the browser user interface (some times referred to as the browser chrome) and associated components, such as the navigation toolbar, browser history and bookmarks. Widgets are another class of containers for Web run-times, where a single web application runs on its own as a locally installed application, where all of the associated markup, scripts, style sheets and media resources are installed as a package.

Web run-times are not restricted to HTML, and other kinds of markup languages may be appropriate. One example is SVG (Scalable Vector Graphics). Another is VoiceXML, a markup language with similarities to HTML (having forms and links) that is designed for use with telephones, together with speech synthesis and recognition, and is gaining widespread adoption in call centres. Yet another example is SCXML (State Chart XML) which is a markup language for event driven state machines. Whilst most people still think of the Web in terms of graphical user interfaces and HTML browsers, the Web is actually much broader than that.

Web run-times don't need to be associated with a local user interface, and instead user interaction could be mediated through an exchange of events with other devices that act as sensors or actuators. The user could therefore interact with an application using multiple devices and multiple modes (aural, tactile, visual). Users may even be able to control applications through gestures with their hands or by moving objects that lack electronics, and which are sensed by cameras or other devices. The ability to project synthetic images into the user's visual field allows for applications that augment reality.

The means for Web run-times to run remotely allows them to function as agents that act on behalf of their users. Such Web agents could run on computing resources provisioned as part of the cloud. This allows users to run applications 24 by 7. Such applications are always available, something that isn't the case for applications running on devices that may be turned off. When it comes to sharing private information between Web agents, there are challenges for dealing effectively with privacy and trust.

4 Device Coordination

When new devices are added to the environment, they need to obtain an IP address and to advertise their presence. There are a variety of mechanisms available for doing this without the need for manual intervention. These are loosely referred to as "zero configuration networking" or zeroconf, which includes techniques such as UPnP, Bonjour, Avahi, mDNS and SSDP. From the perspective of the Web of Things, these are low level techniques, and local software agents are needed to track devices and to expose the services they offer to Web applications. This involves maintaining live models of the context, something I expand on in a later section. Devices may be shared across multiple applications and users, or may be restricted.

4.1 Virtual objects as proxies for things

Web developers shouldn't be burdened with unnecessary details of lower level transport mechanisms. A way to achieve this is for Web run-times to support virtual objects as proxies for real world things. This allows the application respond to input from a sensor via registering an event handler on the virtual object, and to operate an actuator by targeting an event at the virtual object. The details of how the virtual object communicates with the real world thing are hidden from the developer.

In a markup language, the virtual object could be represented as a markup element. This could name the real world thing with a URI in an attribute, or it could describe the capabilities and the context with markup in the content of the element. The Web run-time invokes a broker to bind the object to the thing. The broker could be implicit in the run-time environment, or it could itself be named with a URI. Successful binding would be signaled by an event (load) with a corresponding event (unload) when the binding is broken, e.g. when the device is removed. An error event would signal a failure in binding and supply some details on why, e.g. unauthorized.

Such objects could also be created from Web page scripts, e.g. using JSON (JavaScript Structured Object Notation) to pass descriptions of the capabilities and the context for the desired service. The virtual object can support properties that can be accessed synchronously by the Web run-time, where the implementation of the object hides the messaging needed to synchronize the object with the real world thing it represents. This allows Web developers to use properties and events for a mix of synchronous and asynchronous control. The implementation of the virtual objects could be provided as part of the run-time environment, or it could be a dynamically loadable extension. This could involve trusted scripts with access to lower level services.

4.2 Composition and coordination

The virtual object as seen by a Web developer might be realized as a composition of several devices or services, for example, a scanner could be connected to printer to serve as a copier. The process of setting up the binding to the virtual object involves sending commands to the devices/services involved to connect them together, so that communications flow efficiently rather than via a remote server. Coordination may be needed to ensure fair access to a service, or when things have to happen in a particular order or where access control needs to be managed centrally. Strong coordination involves routing all control messages through an agent that controls and coordinates all of the devices/services it manages. Weak coordination is where the controller arranges a contract between the managed devices/services, but thereafter leaves them to communicate directly.

4.3 Distributed processing

Sensors may perform generic functions, e.g. a microphone captures an audio stream, whilst a camera captures a video stream. This data can be processed for some purpose, e.g. speech recognition on an audio stream, and object and gesture recognition on a video stream. This will often involve some kind of distributed processing model that transforms data and adds metadata. It could also combine data from multiple devices. This could be arranged on behalf of the Web application by the broker as part of a composition of devices and services. This suggests a market for such compositions as something you might pay for.

4.4 Cloud of things

Web applications will be able to make use of devices you own and control, but with the spreading ubiquity of networked devices, you may want turn to a supplier that provisions sensors and actuators according to your dynamic needs, e.g. electronic advertising hoardings on the side of buildings, and placed throughout the city. The "cloud of things" seems like an appropriate term, drawing upon cloud computing which deals with dynamic provisioning of computing resources.

5 Context Awareness

This involves maintaining a live model of users, things and the environment in which they reside. For example, what devices are present in a given room, what rooms there are in a building, and the location of that building on a map of the city. A device can be modelled in terms of the features it exposes to Web applications, e.g. what properties it has, what events it raises and responds to. The behaviour of some devices depends on the state they are in, and this can be modelled as a declaration of a state machine.

In addition to up to date descriptions (metadata), the context may include live objects as proxies for real world things. This allows applications to sense or interact with the world through the models. The context models allow users to browse through descriptions and to follow links, as well as to carry out searches. Models can be represented in a variety of ways including OWL ontologies and XML. A shared underlying ontology is needed to ensure that different representations share the same models.

Web developers shouldn't normally need to deal with the details of how the context models in the cloud are synchronized with the real world. A variety of mechanisms can be utilized as appropriate.

These models can be used to adapt applications to suit the needs of the particular context in which a Web application is being used. User models can describe general preferences as well as preferences for specific applications. General preferences are especially useful for users with sensory or cognitive impairments, e.g. a hearing impaired user may require captioning with video. User preferences also extend to privacy and trust, see the later section in this paper.

Infrastructure providers will want to ensure that the systems they provide are secure. This is likely to involve automatic means to configure safe communications through firewalls, based upon some form of cryptographic credentials. This will often have to be done on behalf of users. The context models can be likened to ice bergs where the visible part is accompanied by a much larger part hidden below the surface of the sea. The hidden parts of the context models are needed to figure out how to route communications to real world things, how to configure intervening security barriers, and how to keep the models live as the context changes.

6 Authoring Frameworks

In principle, there are many possibilities for authoring applications for the Web of Things, including markup languages, scripts and mixes of the two. Current practices for Web applications emphasise scripting in the web page and in the web server. Developers concentrate on the most popular browsers due to widespread variations in the detailed behaviour of different of browsers from different vendors, and also different versions from the same vendor. The advent of mobile browsers is making this problem even worse with variations in display size and device capabilities.

As a result, most sites pay only lip service to the needs of people with impairments. Browsers offer external APIs for *assistive technology*, such as screen readers, but there are significant challenges for providing a usable interface. A web page designed for a high resolution display will have many parts, but the roles of these aren't identified in a standard way that the assistive technology can rely on. This is bad enough for static web pages, and becomes even worse when the pages make heavy use of scripting. W3C's work on ARIA seeks to rectify this situation by defining ways for developers to annotate markup and scripting objects with roles and states that the assistive technology can exploit.

Current web practices fail to separate application logic, data models and presentation. This contributes to the cost of maintaining websites. A small change made for one web page could break many others. This is also bad news for providing a quality user experience on different devices, and made worse by the wide variability of mobile devices. This motivates taking a step back and looking at what alternatives there could be.

6.1 The Cameleon Reference Framework

Markup and scripting languages can be hand edited in text editors. Syntax colouring helps a bit, but it is still error prone and hand editing involves a fairly steep learning curve. It is still however much easier than languages like Java and C++. This allows beginners to get a considerable way by copying and modifying other people's examples and trying things out. Unfortunately, high level web authoring tools have been slow to evolve, and there is a long way to go.

Rather than designing markup languages for the browser as now, we should instead design them to suit the needs for effective high level authoring tools, and not be concerned about direct human editing of the markup. This allows us to focus on cleanly separating out different concerns without worrying about the resulting complexity of the markup. Yes, it should be simple for computers to deal with, but no, it isn't important to address the concerns voiced by people editing markup directly in text editors.

The Cameleon Reference Framework defines several levels of abstraction:

Application domain and task models: these define data models and task models in a way that is independent of the user interface. Task models describe how tasks decompose into subtasks and partial ordering between them. Richer task models add details on how events trigger actions according to associated conditions, along with pre- and post-conditions.

Abstract User Interface: these describe the user interface, but at a level above that of modalities, for example, a selection from a set of alternatives which might be ordered. There could be integrity constraints across fields as well as dependencies where further details are required depending on prior choices.

Concrete User Interface: this makes commitments to modes of interaction and user interface controls such as radio buttons, and image controls for making selections. There is limited control over layout, but the details are left to the next level down.

Final User Interface: this is normally generated automatically from the concrete user interface guided by rules provided by the developer. You can think of this as skinning the user interface. This approach makes it easy to generate the user interface for different platforms, e.g. HTML, SVG, Java, .Net and Flash.

Each level can be formally treated as graphs, and represented as markup[1]. The relationship between adjacent levels can then be expressed in terms of graph transformations. This applies to the representation of the user interface components, and to the representation of events at each level. The transformations are conditional on the context. This means that developers can control how the application adapts to fulfil user preferences and variations in device capabilities and environmental conditions.

6.2 Mashups and pluggable authoring tools

The authoring tool needs to manage the models for each level as the developer works on the application. Imagine selecting a library of UI controls and then dragging them onto a canvas and performing further operations to link them up and adjust their properties. The authoring tool provides an API for libraries to plug into. Adding a group of radio buttons will also add a selection list at the abstract UI level, and connect this up to the application data model, directly the user makes the appropriate setting.

The authoring tool should allow developers to work top-down, bottom-up or middle-out. This raises challenges for the user interface for developers to relate models at different levels. This can be addressed through a combination of techniques including direct manipulation of graphs, property lists, and browsing mechanisms. An agenda mechanism can be used to guide developers on outstanding design tasks.

The Web of Things will involve an ecosystem with vendors of authoring tools, vendors of extensions for authoring tools, traditional web developers, and end-users. The aim will be to give end-users the means to easily create mashups of different services. This includes the ability to personalize services based upon tracking the user's previous interactions, and also those of others who may be considered to be like the user in some way. Some kinds of application lend themselves to social interaction with peers, for example, leaving virtual post-its on real world objects for your friends to come across later.

7 Privacy and Trust

Modern technology makes it possible for companies and governments to track our lives at a level of detail that would be unimaginable a generation ago. The Web

[1] The markup can be supplemented with event driven scripts where extra flexibility is needed.

of Things threatens to take this to a whole new level. Imagine your refrigerator tracking your usage of every item and forwarding this onto the supermarket. They already know what you buy from the widespread use of payment cards. The next step would be for them to sell this data onto your health insurance company, and for you to see the effect on your premiums and level of cover. Of course, that scenario is unlikely to be realized so long as there is sufficient pressure from consumers.

In Europe, there are laws that are intended to safeguard user's privacy by giving them control over what companies can do with personal data. Essentially, users (data subjects) should be able to negotiate their preferences for how long a company (data controller) holds onto personal data, and for what purposes it is put to. This further covers exploitation of personal data by third parties (downstream data controllers). Users should be able to find out what personal data of theirs is being held, to make corrections and to request its deletion. Unfortunately, we are some way from realizing this in practice.

The Web of Things expands the range of personal data that applications have access to. This makes it imperative to consider privacy from the very start when designing the application infrastructure. Protocols are needed for negotiating data handling obligations and for notifying users as part of those obligations. Server-side middleware is needed to track the binding between personal data and the data handling obligations agreed with the data subject.

Privacy goes hand in hand with identity management and access control. The current trend is for websites to identify users via their email address or a web page address (for OpenID). These are globally unique and make it easy to track users across different websites. Privacy has taken a back seat, although most websites do provide a human readable legal disclaimer for their privacy policies, this may be hard to find. Practical deployment of a more flexible approach to privacy will have to wait until better technical solutions are available, and there is sufficient pressure on companies to adopt these.

Researchers are developing cryptographic techniques that minimize the amount of personal data disclosed for a given access control decision. Instead of having to disclose your date of birth, you would instead offer a credential from a trusted third party that you are legally an adult. Another example is where you are required to provide a credential as proof that you are a citizen of a given country, rather than say disclosing your address. Websites will be able to use policy engines to determine what credentials are needed for a given decision, along what data handling policies the site is prepared to implement.

7.1 Call a friend or ask the audience

Users will want to control who can have access to what things under their control. In some cases this is relatively straightforward, for example, in terms of your social network of family, friends and colleagues. In other cases, it is much less clear. Which websites is it reasonable to grant access to your GPS location as determined from your cell phone? Most users don't really know enough to make an informed decision. Public key certificates were proposed as a solution that establishes a chain of trust from certification authorities. Unfortunately, this failed in practice and has been found to be largely unusable.

A likely solution would be to allow users to base their decisions on the advice of friends, trusted authorities, or even the wisdom of crowds (and perhaps based upon a reputation system). This suggests the need for delegation mechanisms where a user agent can consult a trust management service. In one approach, a Web run-time is coupled with a local policy engine that is invoked when an application wants to access a restricted service. The local policies may result in a decision, perhaps by asking the user, or they may invoke a remote trust management service, passing it a description of the security context. The trust management service evaluates the request and returns a policy back to the local engine, which then evaluates it and returns the decision to the Web run-time.

A related problem is when a user needs to authenticate herself with a website, but wishes to minimize any personal data disclosed to that site. Instead of logging in directly with the website, the user signs in with her privacy provider, who in turn passes the relevant credentials to the target website. This is made effortless through HTTP redirection. Users only have to sign on once with their privacy provider and not once per website. The user may further delegate negotiation of privacy preferences to her privacy provider, which becomes a safe place for her to keep her personal data. This raises challenges for how to switch providers, but these should be manageable with the appropriate standards and legal framework.

8 Concluding Remarks

The Web of Things is about exploiting Moore's law to extend the Web out of the browser and into the real world. There are many challenges to overcome, including how to maintain live models of the context, how to simplify authoring, and how to ensure an effective treatment of privacy and trust. Many of the key technologies are becoming available and we can look forward to new opportunities and new businesses.

9 Further Reading

This section provides some pointers to further reading, but is not intended to be comprehensive. You are encouraged to explore further.

9.1 Moore's Law

The doubling of the number of transistors on a chip every 2 years which essentially reflects a steady learning curve and relentless competition:

- `ftp://download.intel.com/research/silicon/moorespaper.pdf`
- `http://en.wikipedia.org/wiki/Moore%27s_law`

9.2 The march of microcontrollers

These inexpensive single chip computers are appearing everywhere in all kinds of devices. The incremental cost of adding some form of networking is dwindling. This is however a case of chicken and egg. Many of the devices that use microcontrollers often have low profit margins. What's in it for device vendors for adding connectivity if the cost of developing applications is too high? The Web of things alters the economics by reducing costs and providing a vast pool of developers.

– `http://en.wikipedia.org/wiki/Microcontroller`

9.3 The Internet of things

This is generally used for the widespread use of RFID tags as a way to identify all kinds of everyday objects. It also includes self organizing low power networks of sensors, and the spread of low cost microcontrollers connected via a rapidly evolving pantheon of technologies.

– `http://en.wikipedia.org/wiki/Internet_of_Things`
– `http://en.wikipedia.org/wiki/Wireless_personal_area_network`
– `http://en.wikipedia.org/wiki/Ambient_network`
– `http://en.wikipedia.org/wiki/Zero_configuration_networking`
– `http://en.wikipedia.org/wiki/IEEE_802.15.4`

9.4 Semapedia - hyperlink your world!

This is one example of an easy way to form links from real world objects to websites. In this case through printing 2D barcodes that link to Wikipedia entries.

– `http://en.semapedia.org/`

9.5 Privacy and trust

Everyone should be concerned about privacy in the digital age. How can we safeguard our privacy as our values continue to change? This is an area where privacy enhancing technologies are still in their infancy. The legal principles have been laid down, but we still have to evolve the technologies and practices to

– *The Spy in the Coffee Machine - The End of Privacy as We Know It*, Kieron O'Hara and Nigel Shadboult, Oneworld Publications, 2008.
– `http://www.w3.org/P3P/`
– `http://ec.europa.eu/dataprotectionofficer/index.cfm?TargetURL=D_INTRO%20EUROPA`
– `https://www.privacyos.eu/`
– `https://www.prime-project.eu/`
– `http://www.primelife.eu/`

9.6 Web Widgets

These are web applications that can be installed and executed within HTML-based web pages or as locally installed applications. W3C is developing a suite of specifications for widgets, covering packaging, digital signatures, scripting APIs, and access control.

- http://en.wikipedia.org/wiki/Web_widget
- http://www.w3.org/2008/webapps/wiki/PubStatus#Widgets_Specifications

9.7 Context awareness

This seeks to make computer applications aware of the environment in which they are operating, for instance user preferences, device capabilities, and the environment in which devices are situated, e.g. what devices are present in a particular room, and the location of that room in a building, and that building in the city. The context can play an important role in interpreting input from sensors, including recognizing users' intentions from their actions.

- http://en.wikipedia.org/wiki/Context_awareness

9.8 Model-based UI design

This is a field of research on layered architectures for user interface design that seek to separate different design concerns via models at different levels of abstraction. This makes it easier to maintain user interfaces and to adapt to changing contexts.

- http://www.isys.ucl.ac.be/bchi/research/cameleon.htm
- http://www.w3.org/2005/Incubator/model-based-ui/wiki/Main_Page

9.9 Assistive technology and designing for accessibility

- http://en.wikipedia.org/wiki/Assistive_technology
- http://www.w3.org/WAI/