

DetectRight.com



DetectRight – how does it compare?

by Chris Abbott and Njal Hansen Wilberg

Mobile Phone Wizards AS

*© 2007 Mobile Phone Wizards AS
Postboks 1803 Håkonsgaten, 5828 Bergen, Norway*

About this document

This document tries to put DetectRight into the context of the other device detection systems you might be using, or have experienced in the past, to explain what problems it addresses in the other solutions, the thinking behind the structure, and an overview of the problem space as we see it. The following represents our understanding of the current state of these solutions. We'd welcome necessary factual corrections should we be in error!

Basics of the problem

DetectRight was created to solve a problem: if a person visits your WAP site or website, how do you adapt the content you serve them? And how do you keep track of the big picture of what's happening on your website? More importantly, DetectRight was designed so that you would have industry-leading detection performance without having to become an expert in device detection!

It's not just a simple device lookup. The problem is more complex than that.

For real-time use, you have to solve a number of problems:

- 1) Matching the headers you see to a particular capability profile (or "device description")
- 2) Serving up data for that match, the more specific the better
- 3) Coping with all devices and browsers sensibly
- 4) Identifying unique users over time, and important factors like Country or ISP.
- 5) Identifying statistical trends, new handsets or new features

Comparison table of solutions to device detection

If this table appears to flatter DetectRight, it's simply because DetectRight was specifically written to address the deficiencies in the publicly available "sacred cow" standards, and to add adjunct features which reflect either the direction in which the industry wants to take device repositories, or which would be invaluable to commercial organisations.

Analyses below will be augmented by detailed accuracy and coverage statistics and data auditing and comparison in a forthcoming white paper.

	DetectRight	WURFL	UAProfile	Browsercaps (.NET)
Identification	Uses all HTTP headers	Relies on the user agent, except for basic handling of the Accept String header.	Uses variants of the HTTP_X_WAP_PROFILE header	Uses all HTTP headers
Data Matching	Uses any method it can to identify device or browser, including user agent analysis, header analysis and UAProfile URL analysis.	Relies on a match with user agent: API allows for limited heuristics by reducing user agent length until match is found. This algorithm is broken for Windows Mobile, Opera Mini, since the relevant information is carried outside the scope of the User Agent. [edit: WURFL now has limited UA analysis, improving things a little]	UAProfile linked to contains data for device. User agents not involved, which is a problem for any device which doesn't carry a UAProfile header, or carries one which is unavailable.	Uses regular expressions on the user agent, and some header matching.
Device Coverage	Every device it sees, it dynamically detects, to the degree possible from the information given.	~75% hit rate of devices on pure WAP sites, worse for web browsers. Figure TBC, based on preliminary analysis.	About 70% of devices visiting WAP sites have UAProfile headers. This means a 30% miss rate: more if you're seeing web browsers as well.	Poor and out-of-date. Companies charged with keeping browsercaps up to date do not do this adequately. Device inheritance fills in some gaps.
User Identification	Unique customers identified: which gives all manner of benefits	Fresh detection per pageview	Fresh detection per pageview	Fresh detection per pageview
Trends and Statistics	Worldmap available giving real-time view of unique visitors,	None	None	None

	DetectRight	WURFL	UAProfile	Browsercaps (.NET)
Server Resources Needed	<p>No additional server or database software required for shared web solution. Clients are encouraged to put all traffic through the server rather than cacheing locally (though clients with dedicated servers get the best of both worlds).</p> <p>DetectRight can, however, be scaled up to semi-dedicated or dedicated solutions potentially handling millions of hits per hour.</p>	<p>In basic form, puts great load on the CPU, disk subsystem or memory, even with multocache. Tera-WURFL reduces this load substantially and has cacheing for subsequent user agent strings.</p>	<p>You'd need to write your own code to isolate the header URL and retrieve the UAProfile, create a database framework to cache them, and build in versioning and import facilities. If you didn't cache the data, performance would be unacceptable in real-world situations, and if you did, you'd need to use some kind of database arrangement, whether file or relational.</p>	<p>Browsercap files are compiled into the application, and runs in the .NET framework: and all that this entails!</p>
Other features	<p>Features soon to be announced are ProfileRight (search engine and web profiles), statistical reporting (ReportRight), and a relaunch of the new DDR core with trust and verification models: conforming to W3C aspirations</p>	<p>WALL available for PHP or Java: WURFL editors and UAProfile converters available.</p>	<p>None</p>	<p>Typically idiosyncratic attempt at WALL-type features, based on textwriters and control adapters.</p>
Direct Cost	<p>From £0 depending on throughput and whether usage is commercial. Standard price: £300 per month</p>	<p>£0 (Open Source)</p>	<p>£0 (UAProfiles available free in a piecemeal fashion)</p>	<p>£0 (comes with .NET).</p>

	DetectRight	WURFL	UAProfile	Browsercaps (.NET)
Cost of Ownership	<p>Subscription cost only Apart from the subscription cost, DetectRight does not need to be updated, and it doesn't require any server resources short of creating a socket to contact the server. DetectRight deals with 95% of requests within less than 20ms of server time.</p>	<p>Manpower and expertise required: potentially expensive. Although it is theoretically possible to use WURFL in a "set it and forget it" fashion, most people using it for commercial purposes would need to maintain their own patch files: which means jumping on the treadmill of collecting device information and converting to the WURFL format.</p> <p>In addition, if you wish to detect web browsers or perform unique customer recognition, then you have to write your own routines. WURFL's costs can be summed up as manpower (both in terms of data collection and programming hours spent solving its problems), and/or hardware. In addition, services such as geolocation which may be crucial to the success of, say, content selling or regional adaptation would have to be bought from elsewhere: potentially more cost and hardware expense.</p>	<p>Manpower and expertise required Most companies that rely on UAProfiles hire between 1 and 10 people to process their data.</p> <p>Factor in the programming and schema-adaptation required, along with the programming to cope with whatever devices don't have UAProfiles, and you're looking at between £20,000 and £100,000 per year: more if the data collection staff are highly qualified, along with substantial programming time and R&D to build a custom solution: and add in more expertise requirements if you're dealing with UAProfiles and WURFL.</p>	<p>Manpower and expertise required to avoid monetary loss through inadequate service to customers. In reality, very few companies using .NET browsercaps spend any time updating their browsercaps collection: mostly because the scope of work is far too large. This is costing them dear, since despite its clever (if overly convoluted) framework, the hit rate is bad: only around 20-30% of devices, although it's good on web browsers and Windows Mobile devices, as you'd expect.</p> <p>The cost of ownership is thus either hiring people to keep the data updated, or losing customers through poor detection of customers.</p> <p>In addition, browsercaps is what it says: a listing of browser capabilities. It does not deal with media types, video, physical or network capabilities, or many other factors which may be important.</p>
Platforms	SOAP standard: specialised client available in .NET and PHP.	XML is platform agnostic. Clients available in most languages.	XML is platform agnostic.	.NET only.

	DetectRight	WURFL	UAProfile	Browsercaps (.NET)
Compensates for lack of data by:	Intelligent heuristics and module analysis (for instance, browser or jvm analysis)	Device Inheritance (main drawback is that too many data fields fall back to the generic device)	Not including it!	Device inheritance, some header analysis biased towards non-mobile browsers.
Updates	DetectRight updates every time its system see a new device in the wild: and since it produces profiles for all clients, it's a "set it and forget it" solution.	Infrequent released updates, relying on voluntary help, necessitating manual patches and expertise in device capabilities.	No organised updates, and no public announcements of UAProf availability.	Infrequently updated with the .NET framework.
Accuracy	Good, since most data is "best effort": no fallback data used, which increases accuracy. DetectRight's ethos is that data has three states: "true", "false", and "Don't know". DetectRight specialises in dynamic data, which also means calculating assumptions about the type of Net connection and the geolocation details.	Good where data exists, though sometimes inconsistent. Model/manufacture presence/consistency is poor, thus limiting its use greatly for any queries which key off manufacturer/model. Very poor where generic fallback data has been used, since this is by definition not related to the make or model of the device. System is incapable of properly detecting Opera Mini and Windows Mobile devices, since this data is dynamic data within the headers.	Much better than it used to be, though the amount of UAProfile revisions and corrections is worrying, and illustrates the need for versioning in any UAProfile-based solution. Smaller manufacturers often produce perfunctory or inadequate UAProfiles, and many don't include it at all.	Data quality not audited, but coverage is very poor.
Notes on schema	DetectRight's current schema has many more derived binary fields than UAProfile, but also has many more "informative" fields than WURFL.	WURFL has taken a very binary approach to its schema, which is consistent with its development as an aid to programming, and making binary if/then decisions. This has resulted in a schema which needs updating in the event of new versions of particular items (such as, for example, DoJa), and a lack of "bag" fields, such as lists of mime-types.	UAProfile's approach to its schema is very different from WURFL's, concentrating much more on lists of supported facilities, and ignoring facilities which aren't supported (thus you're left with much less information).	The schema focuses almost entirely on binary fields specifying the existence of various browser capabilities.

DetectRight's development was heavily weighted towards being useful for real-time analysis of visitors to a website. This makes it very different to any other solution out there, including the expensive enterprise offerings. DetectRight analyses all of the headers it can find: if a handset is using a different browser, or has been upgraded to support new file formats, then DetectRight will detect that. A complete set of heuristically generated but accurate profile items are generated for each customer, tailored to their own device. And in a fraction of a second.

DetectRight can:

- 1) Detect and analyse devices and browsers it hasn't seen before in amazing detail and faster than a local WURFL lookup.
- 2) Efficiently identify unique users using our own algorithm: it's capable of detecting different users coming from the same telco IP addresses, for instance.
- 3) Import and convert new UAProfiles in real-time, and even correct them. DetectRight is the engine of UAProfile.com: the world's biggest repository of UAProfiles, but DetectRight is much more than "uaprofiles in a database".
- 4) Classify spiders and bots as well as mobile and web devices.
- 5) Properly Identify devices that don't have UAProfiles, such as legacy devices, iDen devices and more...
- 6) Provide visitors as an XML feed (for example, to our Global HQ Map)

DetectRight does not:

- 1) DetectRight does not rely solely only on UAProfiles, since only about 70% of mobile devices have them.
- 2) DetectRight does not use any of your CPU time, since it uses our optimised server and client code.
- 3) DetectRight does not require you to write more than three or four lines of code to take advantage of its real-time customer profiling

DetectRight is always improving

Thanks to our experiences at W3C, we have been busy building the first device repository in the world that will harmoniously combine the major device description schemas, while not losing granularity of information. It will include a full trust/verification/validity model and the ability to acquire output in various schema formats. The engine of DetectRight will also be further improved taking advantage of the new DDR core.

The problems of a custom solution

Often an organisation has invested heavily in its own solution, whether this be UAProf-based, WURFL-based with patches, or an entirely new method. These solutions come in two flavours: underfunded and not-fit-for-purpose, or adequate (maybe even brilliant), but resource-heavy. In addition, the R&D needed to keep up with changes and both devices and detection methods is prohibitive even for companies to whom device detection may be a more core activity. For organisations for whom device detection is a drain on the profit-margin of the company, the treadmill of running your own device solution can be an albatross around the neck.

In addition, the resources spent developing a solution will not be reflected in testing and auditing the data: partially because it will almost certainly be in a closed proprietary structure, and partially because to do so would mean admitting the failure of the current solution. This means that the vast majority of solutions out there are assumed to work, but almost certainly have a much higher failure rate than their owners would predict. And often, the only mark of a failure is a client bothering to inform you: a rare occurrence. And after all those resources are spent, do they have any data to show for it? Often the data will not be captured for later validation or analysis, but thrown away, analysis restricted to logfiles of user agents. Valuable business intelligence lost.

DetectRight offers a way off the treadmill: some companies already use it to keep up with the newest devices and help create their WURFL patch files, or take advantage of the system indirectly through downloading UAProfiles from UAProfile.com (something which the planned "triple-option schema output" will immensely speed up). It can augment your system offline, provide valuable data validation and auditing (a service we offer as a consultancy service), and even has the ability to throw http headers at a site in order to record its output for various devices: a facility we plan to use to test other device solutions.

In addition, the new DetectRight DDR Core has been designed to allow private custom schemas which can integrate harmoniously with the largest repository of device data ever compiled: and the trust and verification models ensure your data is given appropriate priority. In addition, you don't need to worry about actually detecting the devices: DetectRight's accuracy means that your data automatically gets attached to the proper device/model/major version/firmware number, and where this data is used out of context, is weighted appropriately.

Soon, DetectRight will offer a home for the private device data of the world, whether it be as a dedicated enterprise-level solution in their organisation, or through the shared web hosting. Our vision is shared with the W3C: to create an ecosystem where companies access shared device data incorporated with their own expertise, through a common API. We're taking a lead in this, but whatever W3C's DDWG/MWI come up with, we'll integrate it.

Chris Abbott
Njal Wilberg
Mobile Phone Wizards AS