

## Architectural Styles for Augmented Reality in Smartphones

**Ben Butchart**

**EDINA, University of Edinburgh**

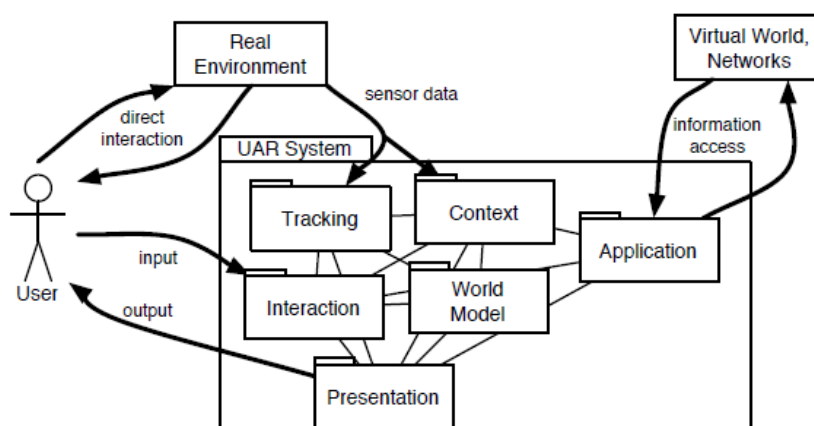
**b.butchart@ed.ac.uk**

### **Abstract**

In this paper we briefly examine the architecture of augmented reality in smartphone applications. While the architecture is fairly consistent across AR browsers / SDKs, there are some important differences. Appreciating these differences will help developers choose a framework or browser implementation that best meets their needs. Understanding the architecture of these systems will also expose the gaps in current implementations and help us to think clearly about the likely evolution of augmented reality browsers over the next few years.

### **A Reference Model**

Thomas Reicher's "Framework For Dynamically Adaptable Augmented Reality Systems" thesis [1, 2] provides a reference architecture for comparing different augmented reality frameworks and this is a good starting point for our discussion. An illustration of Reicher's reference architecture is shown below. (from Asa MacWilliams[4])



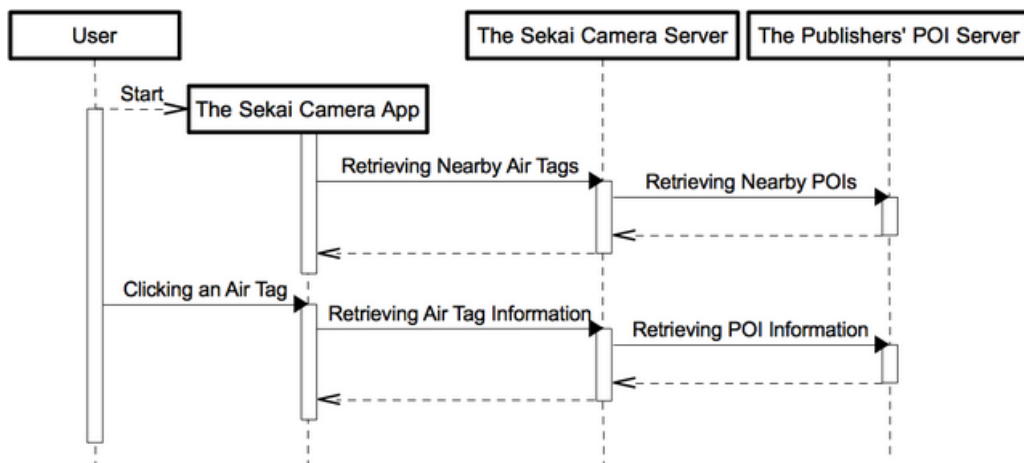
In this model, the augmented reality system is organized into six logical subsystems. A thorough explanation of these subsystems is given by MacWilliams [4]. For convenience below is a brief summary.

## Tracking subsystem

This subsystem is responsible for responding to changes in the user’s location and orientation so that digital objects can be superimposed on the reality view in a way that convinces the user that the digital object is part of the natural environment. For smartphone AR browsers tracking is typically based on location sensors in the device such as GPS, compass and accelerometer allowing 6 degrees of freedom in displaying a digital object. Some browsers have image recognition services available to obtain a more accurate fix.

## Application

This subsystem is responsible for the main control flow logic of the application and coordinating communication between other subsystems. In smartphone browsers this application logic is typically implemented in the client application, as can be seen in the sequence diagram for the Sekai Camera app.



Sekai Camera OpenAir API: <http://support.sekaicamera.com/ja/archives/5963>

This sequence diagram corresponds to the “Gateway” pattern described below and provides a typical example of application logic implemented in smartphone AR browsers.

## World model

This subsystem stores and provides access to a digital representation of world, including points of interest, 3d objects and metadata about the model itself. In the jargon of AR browsers these models are called “channels”, “layers” or “worlds” and metadata is referred to as “publishing information” or “provisioning” data.

## Presentation

This subsystem is responsible for all output, including reality view streams, 2d and 3d rendering and audio and tactile outputs.

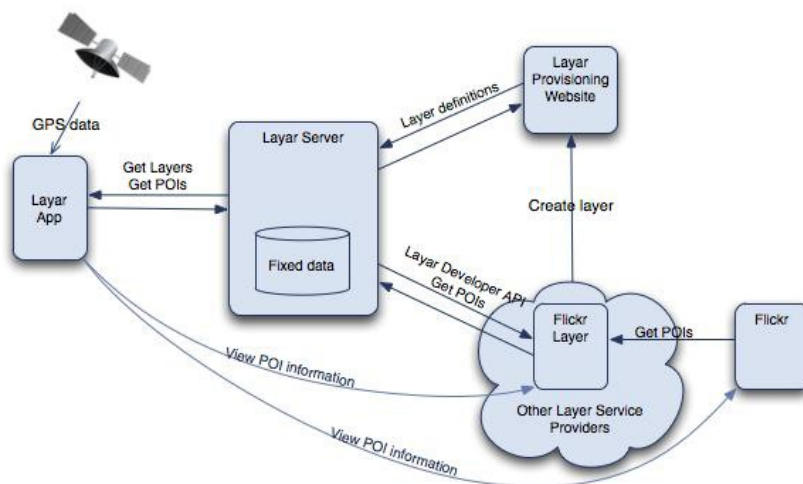
## Context

This subsystem provides the application with context about the status and situation of the user. This includes both static information about the user (e.g. name, avatar, friends) and near real time status ( in office, busy, sitting down, at gym etc.)

Logically, tracking is a special case of context, but as tracking is so important to AR applications, this aspect of user context is afforded a separate sub-system in many AR implementations [3].

## Comparing AR browsers with the reference model

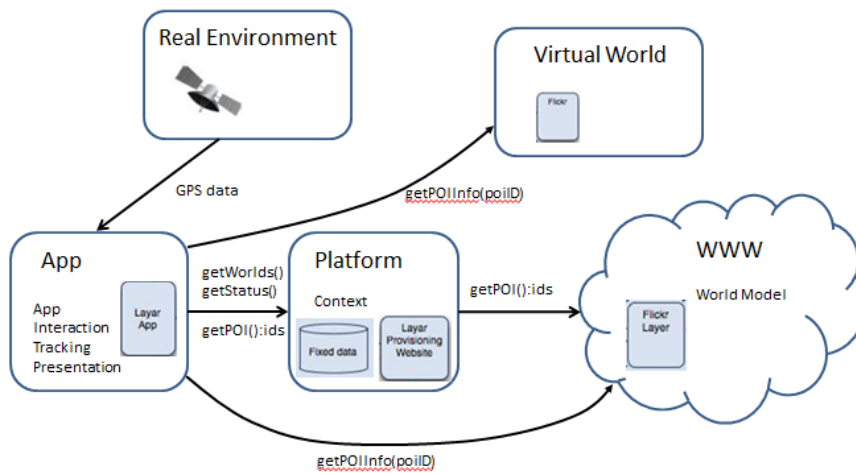
The reference model is suitable for comparing a general class of augmented reality applications, including systems utilising both handheld and head mounted displays. Our concern with applications deployed to handheld devices means we can simplify the model and focus on just those subsystems where architectures differ. To illustrate how we can map the reference model onto a smartphone AR browser let's take a look first at Layar's [5] architecture diagram shown below.



Layar architecture diagram: <http://layar.pbworks.com/w/page/7783214/Layar-Platform-Architecture-Overview>

The Layar architecture diagram above shows how the Layar app communicates over a wide area network with the Layar platform, a server which holds information about Layar channels (“fixed data”) and proxies requests to POI services hosted independently by Layar providers on the World Wide Web.

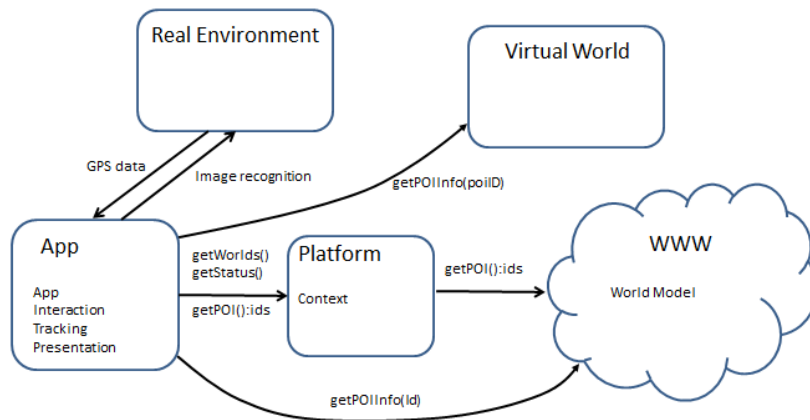
## “Gateway” and “Platform” architecture



Layar architecture mapped onto Reference Model

This modified diagram shows how the Layar application maps on to the subsystems described in the reference architecture, providing a more general model that helps us compare this design with other applications. As can be seen most subsystems in Layar are coded into the App ( i.e. the AR browser) with just the World Model ( the Layar Points of Interest API) and the Context ( Layar Channel API) subsystems implemented outside the browser. Characteristic of this design pattern is the role of a vendor provided web server (the “Platform” in the diagram above) that mediates requests from the AR browser to the World Model. For this reason this design is dubbed the “Gateway” architecture as the Platform acts a gatekeeper to world models published on the World Wide Web. Most of the AR browsers we have discussed adopt some variation of this pattern, principally because the platform provides vendors with opportunities to generate revenue. The platform does offer benefits to the users and developers too, creating a place for the user to discover channels and providing tools for creating and testing channels.

A minor variation of this model includes the use of image recognition for registration and tracking. A general diagram for the Gateway pattern is shown below.

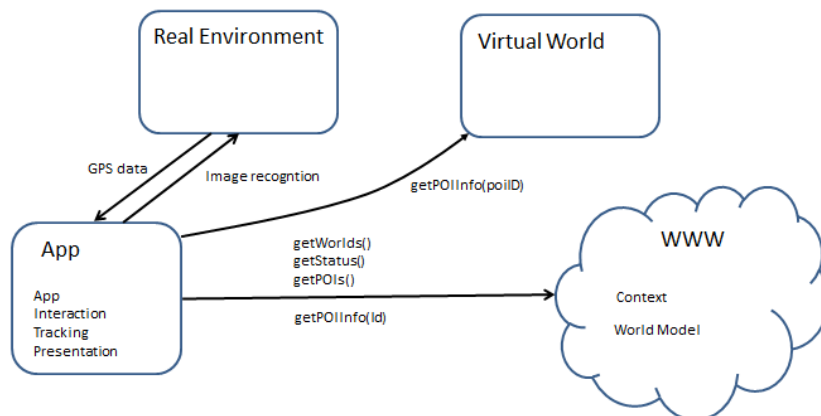


“Gateway” Architecture: (e.g. Layar [5], Junaio [6], Sekai Camera [7] and Wikitude Worlds [8])

A variation on this architecture shifts the World Model sub system from the web into the platform. This “Platform” model is more restrictive for developers but offers vendors more control over content. Typically a publishing API and administration console is provided to assist developers in making their own data available through the platform.

### “Web” Architecture

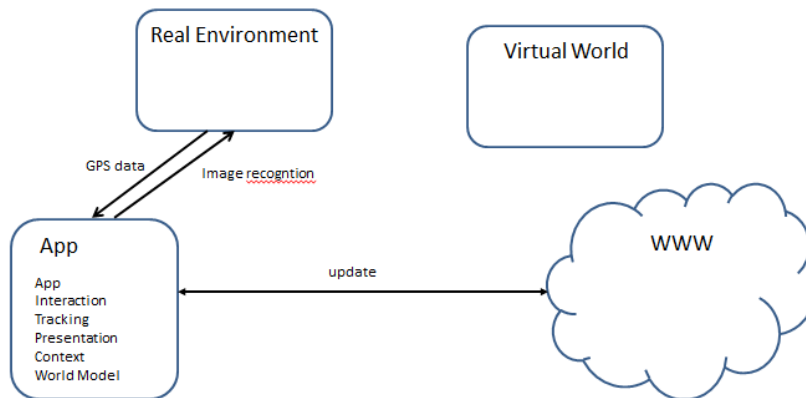
It is possible to eliminate the platform component. In this case the AR application becomes a genuine web browser with unmediated access to the World Wide Web. This “Web” architecture releases the developer from any dependency on a platform provider but burdens them with the task of making their content discoverable and controlling access to the world model.



“Web” architecture style: (e.g, LibreGeoSocial [9])

## “Standalone” Architecture

Wikitude API browser [8] incorporates every sub system into the browser application, including POI data and channel publishing information. This “Standalone” architecture has the advantage that the application is not dependent on a wide area network connection and the developer is free of any vendor platform restrictions on use. The downside is that the application has to be re-installed (updated) if new data becomes available and it is harder to integrate with 3<sup>rd</sup> party POI providers.



Standalone architecture (e.g. Wikitude API [8])

## Discussion

Playing around with the model can lead to some thoughts on new possibilities for AR browser designs. Tracking can be moved onto the Platform – for example where a server process recognizes an image the user has sent and uses it to locate the user’s exact position and orientation. The Application subsystem might also move into the Platform, where data from the phone’s sensors are combined with that of external sensors to control the workflow of the application to improve sensitivity to environmental conditions. It is also interesting to speculate how change in smartphone hardware and software capabilities may influence future architectures. For example the introduction of RFID/NFC technology [10] might enable parts of the tracking and interaction subsystems to migrate from the App into the Real Environment/Platform.

## Resources

- [1] Reicher, Thomas , 2004, A Framework for Dynamically Adaptable Augmented Reality Systems (PhD thesis), <http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2004/reicher.pdf>, Technische Universität München, Universitätsbibliothek
- [2] Asa MacWilliams Thomas Reicher Gudrun Klinker Bernd Brügge 2004 conf/mixer/2004 MIXER <http://SunSITE.Informatik.RWTH-Aachen.de/Publications/CEUR-WS//Vol-91/paperE4.pdf> db/conf/mixer/mixer2004.html#MacWilliamsRKB04
- [3] Thomas Reicher, Asa MacWilliams, Bernd Brügge, Gudrun Klinker: Results of a Study on Software Architectures for Augmented Reality Systems. ISMAR 2003: 274-275
- [4] A. MacWilliams  
**[A Decentralized Adaptive Architecture for Ubiquitous Augmented Reality Systems](#)**  
*Dissertation, Technische Universität München, June 2005.*
- [5] Layar Home Page <http://www.layar.com/>
- [6] Junaio Home Page: <http://www.junaio.com/>
- [7] Sekai Camera <http://sekaicamera.com/>
- [8] Wikitude Home Page <http://www.wikitude.org/en/>
- [9] LibreGeoSocial Home Page <http://libregeosocial.morfeo-project.org>
- [10] Richard MacManus, April 2010, "Bringing Sensor Networking To Smartphones"  
[http://www.readwriteweb.com/archives/dash7\\_bringing\\_sensor\\_networking\\_to\\_smartphones.php](http://www.readwriteweb.com/archives/dash7_bringing_sensor_networking_to_smartphones.php),  
accessed 28/01/2011