Embedding Chromium into AGL demo platform with WAM

Lorenzo Tilve

Web Engines Hackfest 2021

igalia

What is AGL demo platform?

- **Automotive Grade Linux (AGL)** is an initiative by the Linux Foundation oriented to provide an entire open source stack for the vehicle industry.
  - https://www.automotivelinux.org/
- Multiple software layers integrated into an **Unified Code Base (UCB)** for several supported embedded platforms, into different image versions.
- One of them is the Infotainment Vehicle Information (IVI) **demo platform**.
- This image provides an application framework, which was originally consumed by Qt applications.

Why integrating a Web Runtime?

- Provide a way to ship and run independent **web applications** in the platform seamlessly.
- Webapps should aim to have the same level of security granularity and performance as native applications
- Targeting to a full HTML5 UI:
  - Capability to reach a wider scope of developers
  - Use of standard web technologies and independency of frontend framework.
  - Ease of development, debugging, customization and interoperability with other services.

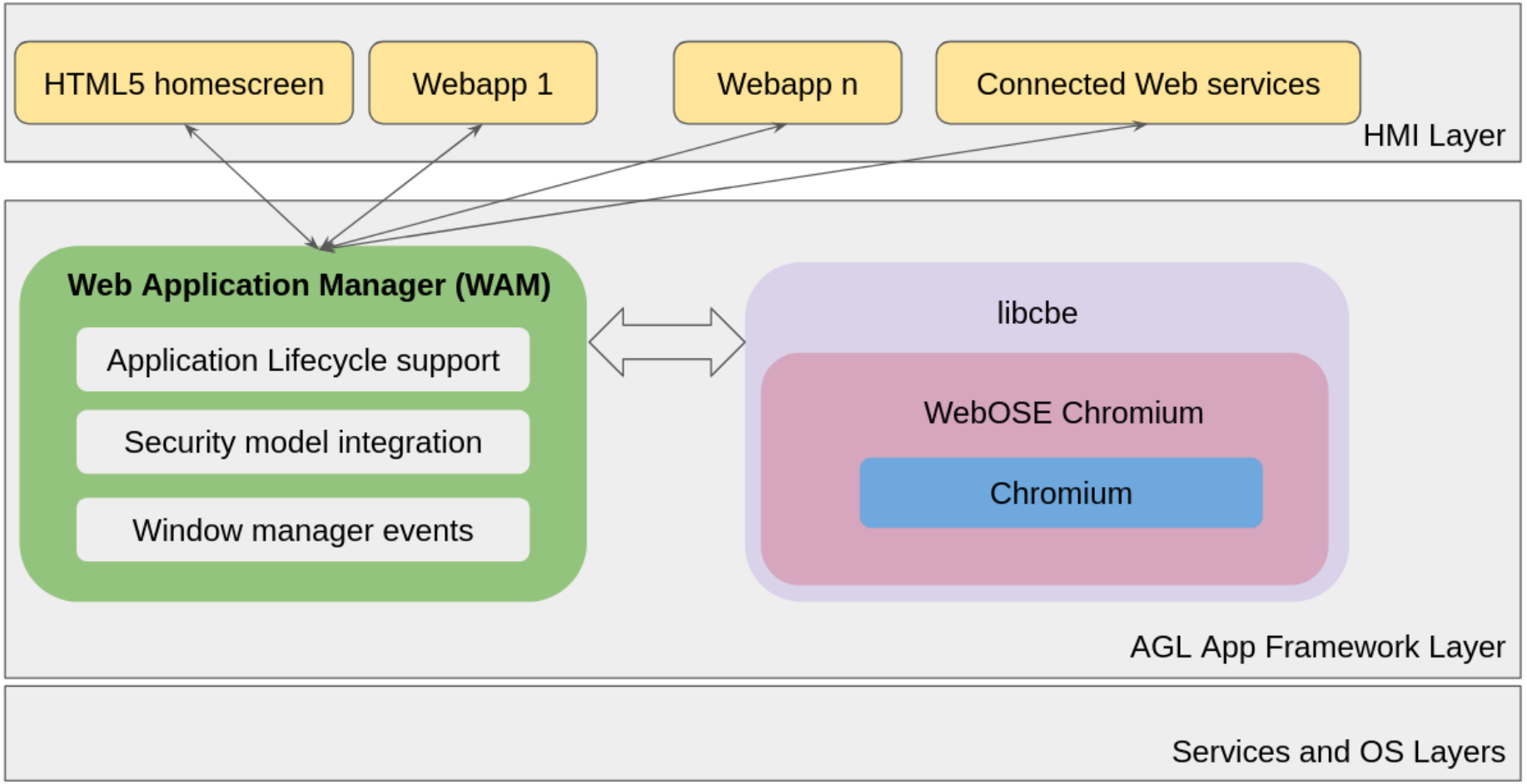
Using WAM to embed Chromium

- Most of these functionalities were already covered by the existing **Web Application Manager (WAM)** originally developed by **LG for WebOS**.
  - Launches independent webapps, adding additional security support.
  - Manages web applications life-cycle depending on visibility status, memory management.
  - Requires changes on the Chromium side to connect WAM.
- Allowed targeting for faster adaptation into the AGL demo platform:
  - Wraps native platform IPC and APIs for ease of use in JavaScript.
  - Already some existing Yocto recipes for WebOS.
  - Original support for initial Wayland implementation.

Current Web Runtime architecture

- The existing approach of the AGL web runtime, is based on the adaptation of WAM + the modified Chromium versions from WebOS by LG.
  - https://github.com/webosose/wam
  - https://github.com/webosose/chromium84
- Adapted for AGL and integrated into their build recipes hierarchy, into meta-agl-demo layer
  - https://gerrit.automotivelinux.org/gerrit/gitweb?p=AGL/meta-agl-demo.git;a=tree;f=recipes-wam

Current Web Runtime on AGL architecture



Current Web Runtime architecture

- Webapps on the Human Machine Interaction (HMI) layer communicate with the application framework with websockets.
  - All the connections with the different services provided in the internal layers, are wrapped by the AppFramework, and used equally by native or web applications.
- WAM interface to Chromium is done through libcbe.
- Initially integrated with Weston Wayland compositor, currently running on top of libweston based agl-compositor developed by Collabora.

Building/testing the demo platform

- Checking out the code, and building the entire demo platform with Yocto:

```
repo init -b master -u https://gerrit.automotivelinux.org/gerrit/AGL/AGL-repo
repo sync

source meta-agl/scripts/aglsetup.sh -f
    -m <m3ulcb | raspberrypi4| qemux86-64> -b build agl-devel agl-localdev
    agl-demo
bitbake agl-demo-platform-html5
```


Building/testing the demo platform

- For additional information on how to create and test webapps
  - https://www.youtube.com/watch?v=quM6DXOAVvM

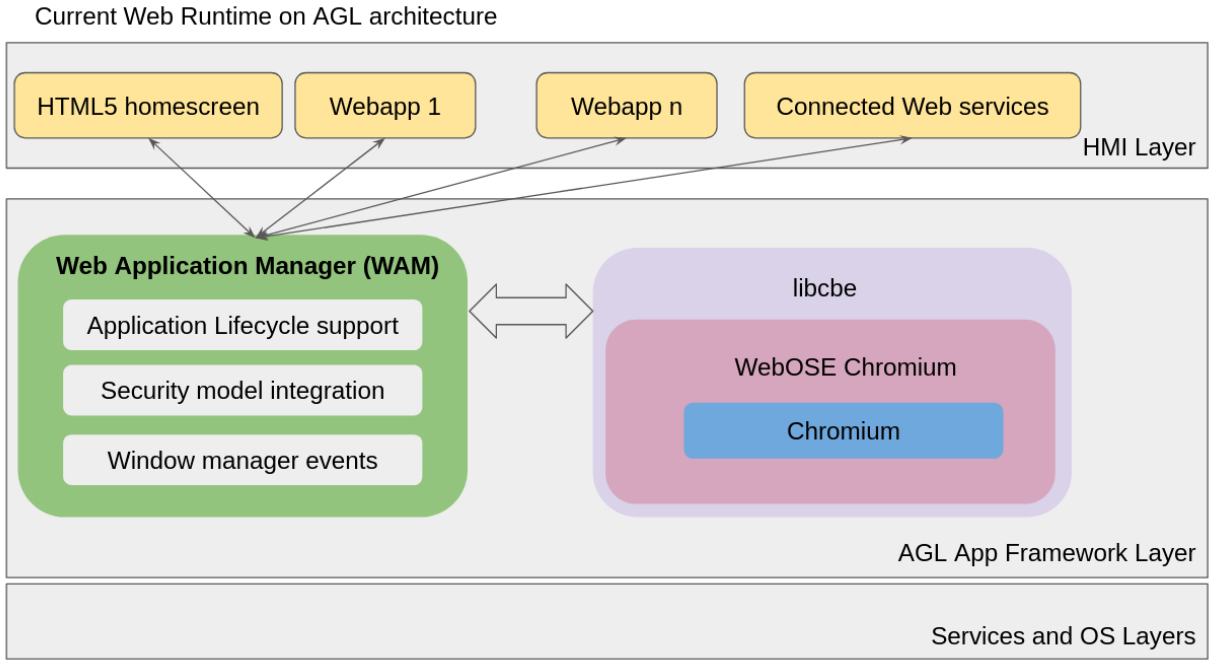
Syncing with upstream

- One of the current dependencies is on the LG WebOS release cycle, which upgrades between certain Chromium versions (68, 72, 79, 84) after all the platform-specific changes are validated.
- Several AGL specific modifications have to be backported between releases, as the integration with the compositor, WAM Qt dependencies removals, etc.
- It could make the process simpler to keep WAM but adapt it to use an alternative approach to embed Chromium.

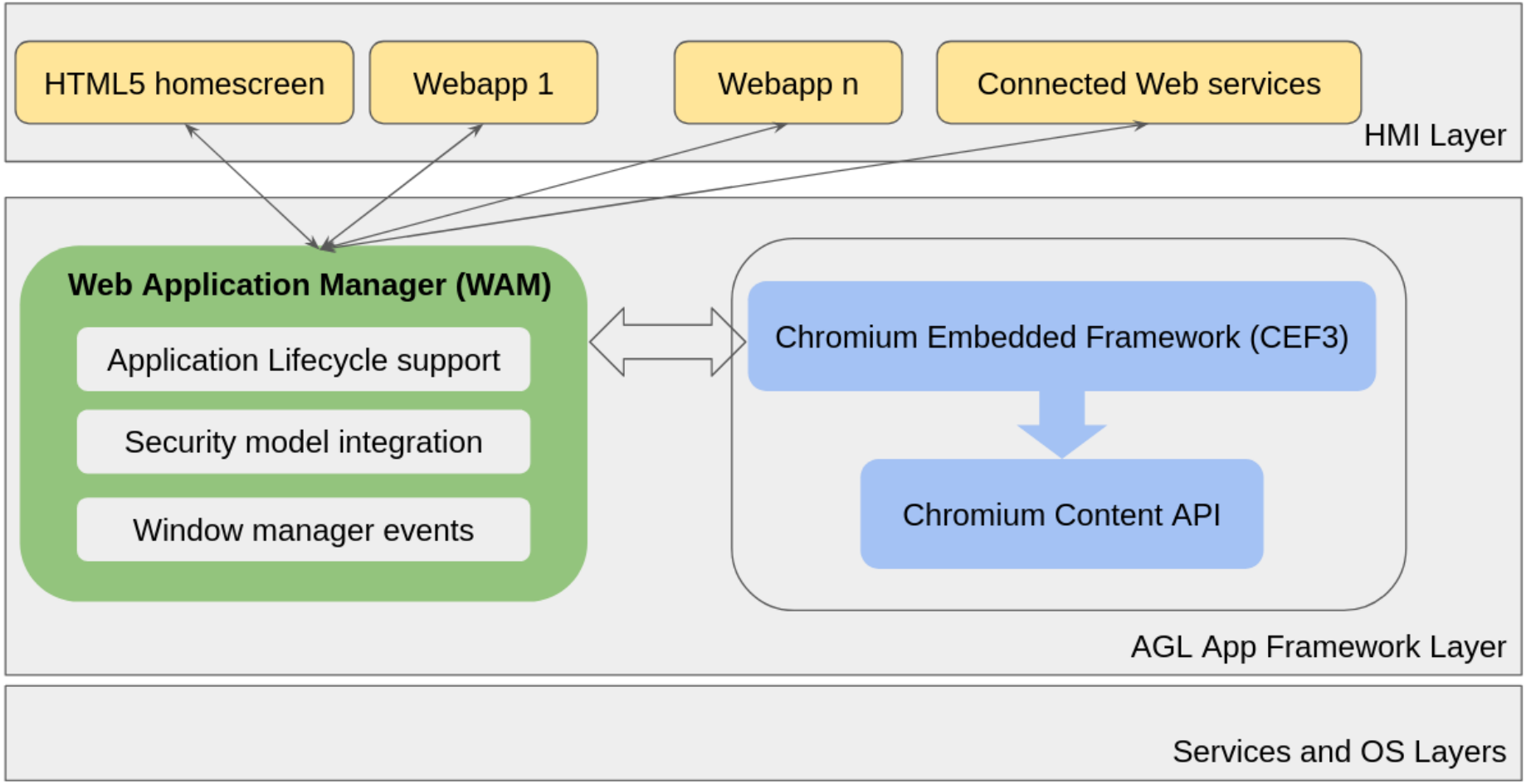
Why considering WAM on top CEF

- Chromium Embedded Framework (CEF)
  - https://github.com/chromiumembedded/cef
- Consolidated framework designed specifically for embedding Chromium and widely adopted by multiple upstream projects.
- Has been getting improvements on Ozone/Wayland support.
- Synced with fresh Chromium release channels.

Modify the mentioned architecture



Web Runtime on AGL architecture using CEF



Make WAM work on top CEF instead

- An interesting path would be to explore adapting WAM to use directly CEF instead of modified WebOS/Chromium.
- This could allow Web Application Manager to get a potentially wider upstream usage.
- Stable API between WAM and CEF.
- Smaller delta, as the majority of the work would be needed on WAM to adapt for its connection with CEF, without the need of backporting changes into Chromium.

Path for CEF integration

- Preparation:
  - Build the CEF library (aka libcef) using the AGL build system, adding bitbake recipe to build libcef for the different hardware architectures.
- Infrastructure:
  - Remove the dependency on WebOS from WAM and make it compiling and linking with libcef instead.
- Prototype:
  - Make PoC stage of WAM to integrate a minimal view using CEF.
- Completing the integration:
  - Iterate the prototype to support the pending functionalities.


igalia