AGL Reference Compositor: 2019 plans

AGL System Architecture Team

Daniel Stone
Graphics Lead, Collabora
daniels@collabora.com

COLLABORA

AGL F2F Tokyo: March 2019 Compositor discussion recap

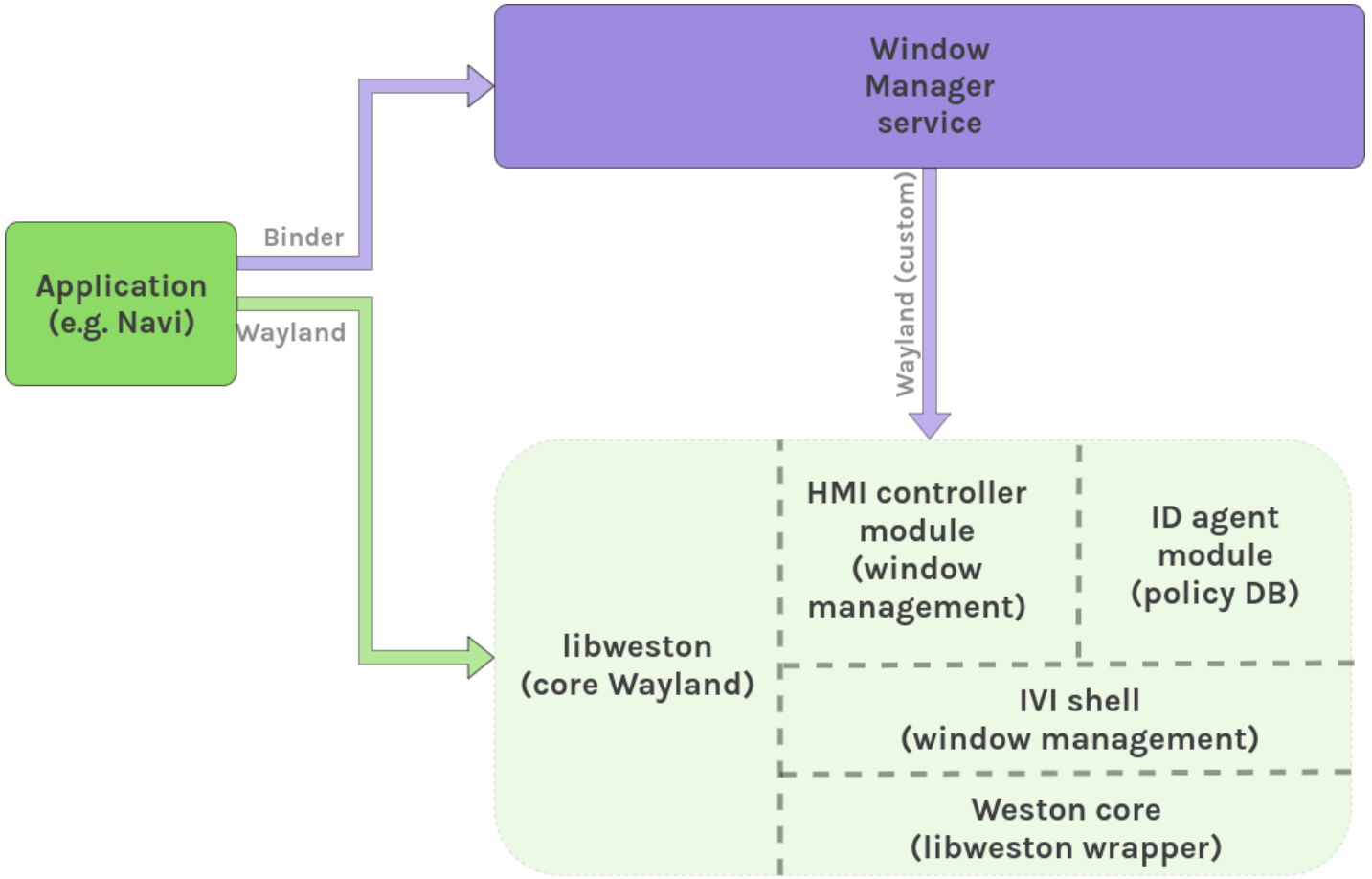
Background

- AGL defines high-level APIs: Window Manager, Home Screen
  - heavily integrated with App Framework
- Applications rely on Wayland for graphics & input
  - 'Wayland' not very well defined (what extensions?)
  - provided in AGL by Weston (community reference server)
  - lower-level API, not directly integrated with App FW

Definitions

- Window management: policy and placement to show HMI, application, notifications (etc) on screen
- WindowManager/WM & HomeScreen/HS: AGL defined Binder APIs
- Wayland: core Wayland protocol and common extensions
- Compositor: Wayland display server, hosting clients, displaying output, forwarding input
- Weston: community-maintained reference compositor
- IVI shell: Weston module allowing external WM

Current Window Manager design



Pain points in current design

- Complex architecture: many components
  - Weston IVI shell, GENIVI Wayland IVI extension, AGL WM
  - functional changes may need several API and protocol extensions (C, Wayland, Binder)
- Difficulty of change: multiple unsynchronised communities
  - Weston upstream, GENIVI extension, AGL WM/HS

COLLABORA

AGL window management
Proposed new architecture

Motivations for change

- Simplify architecture
  - multi-process & multi-protocol design does not improve reliability
- OEM flexibility for HMI customisation
  - make it easier for OEMs to change window management policy, create differentiated UI

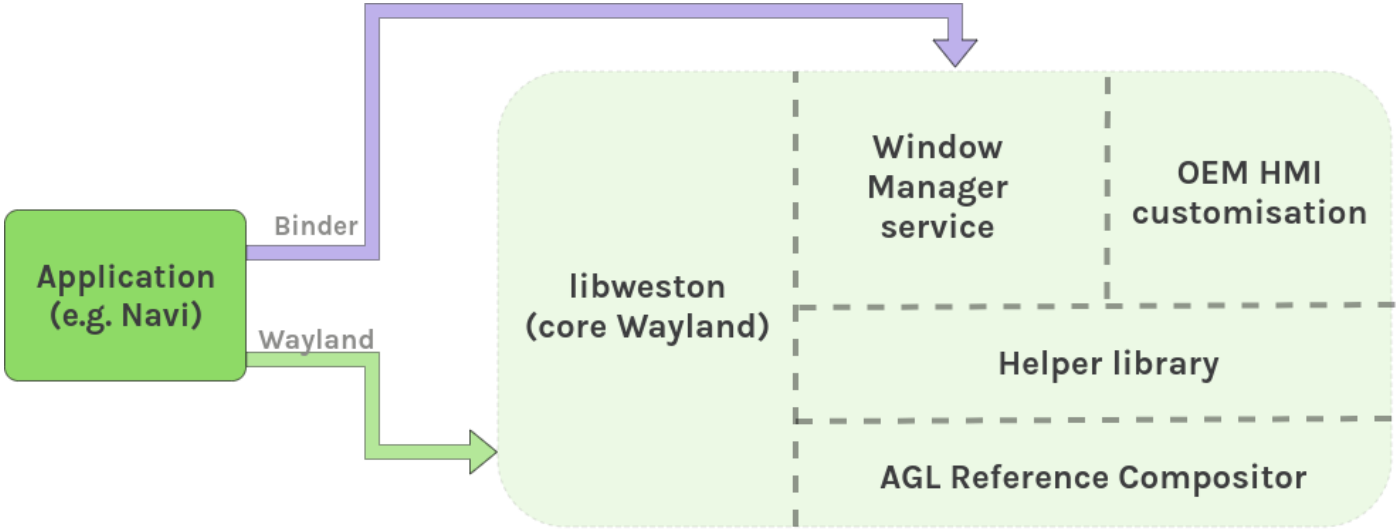
Assumptions for new design

- Differentiate at correct level for AGL
  - when creating new/specialised components: does this add value?
- Reliability and performance are critical
- Clearly define interfaces: what can each component expect of other components?

New design basics: high level

- Build reference compositor framework based on libweston
- Provide helper API alongside libweston implementing AGL APIs and integration
- Provide full reference compositor/WM for demo usecases
- Provide clear points of UI/WM differentiation and customisation for OEMs
- Allow OEMs to replace entire stack with own implementation if they implement the same APIs

Proposed Window Manager design



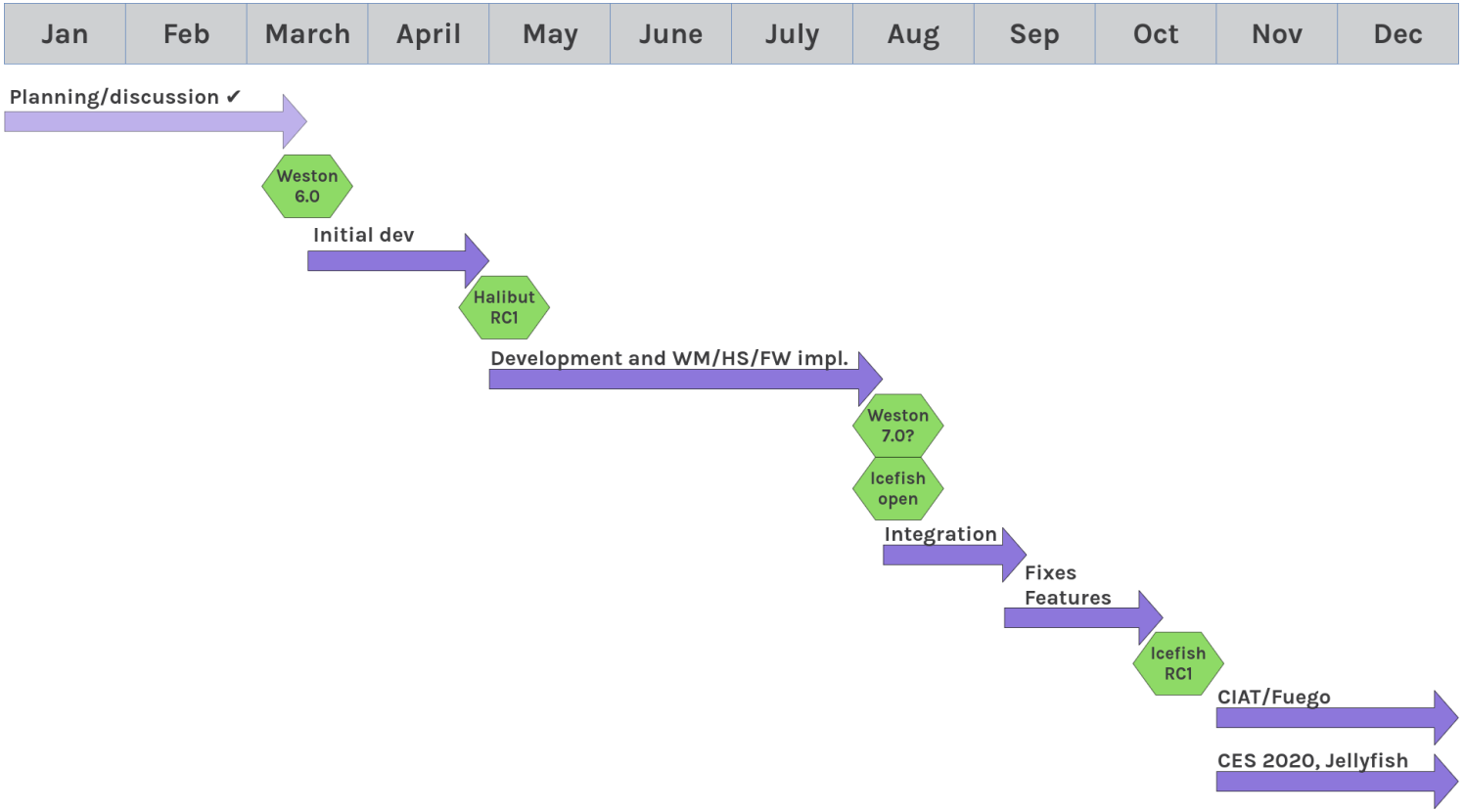
New design basics: technical detail

- Eliminate multi-process design: window management in same process as compositor
- Provide same AGL WM/HS Binder APIs to clients
- Integrate with AGL App Framework main loop
- Enable automotive functionality: CAN input (buttons)
- Deep support for logging, tracing, capture
- Clear path for multi-process apps (e.g. Navi UI & map) to control full placement and presentation (layers)
- Testable through CIAT/Fuego

COLLABORA

Current and future work plans

Work schedule



Currently ongoing work

- Clearly capture and document design, requirements
- Define external APIs: what must an AGL compositor implement for portable clients?
- Create publishable documentation and work plan

Next steps: Halibut

- Upgrade Weston version for Halibut Yocto build
- Allow for integrated Wayland and AGL App Framework main loops
- Continue bring up of reference compositor based on libweston
- Port AGL window manager service to libweston base
- Ensure basic HMI and clients work as is

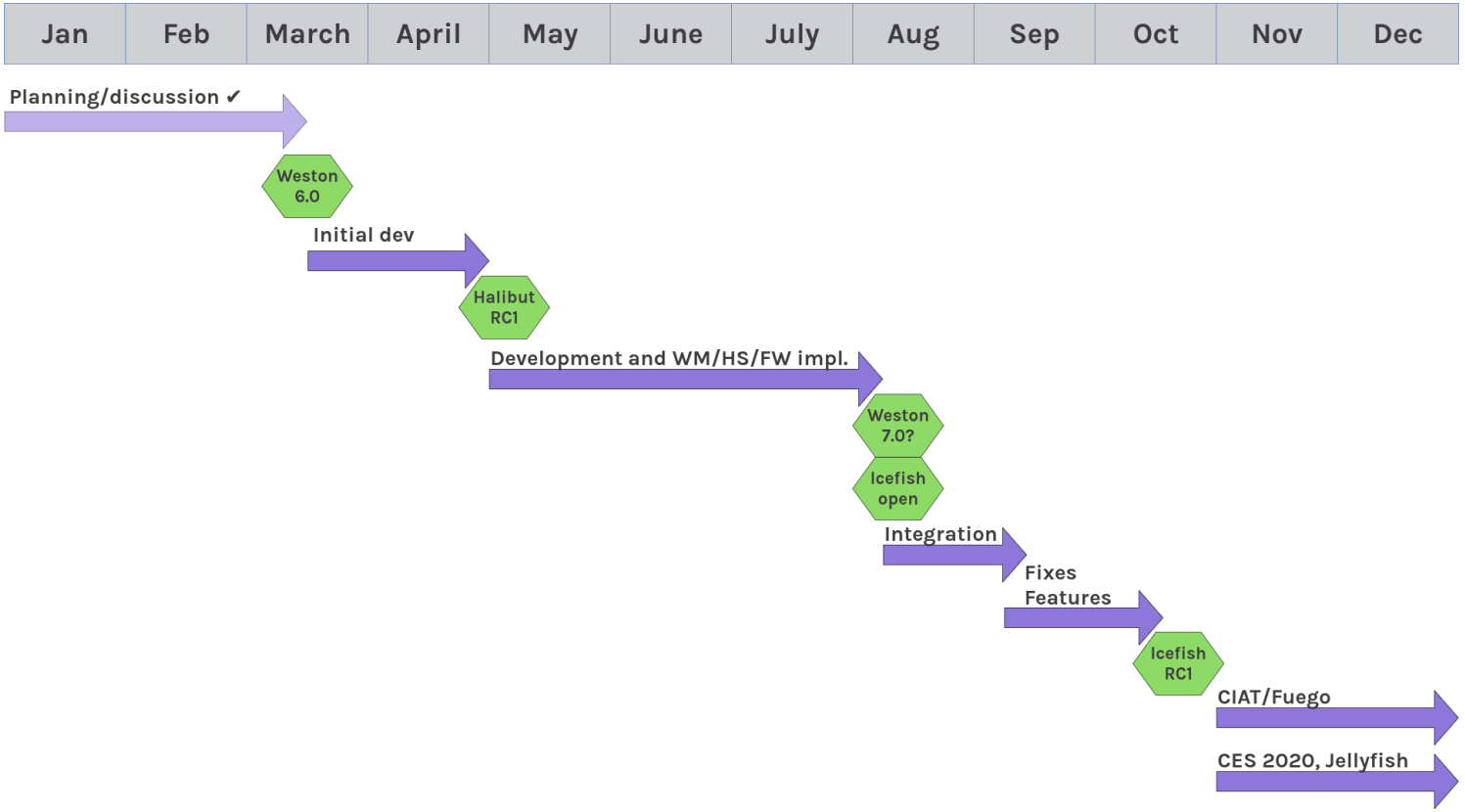
Next steps: Icefish

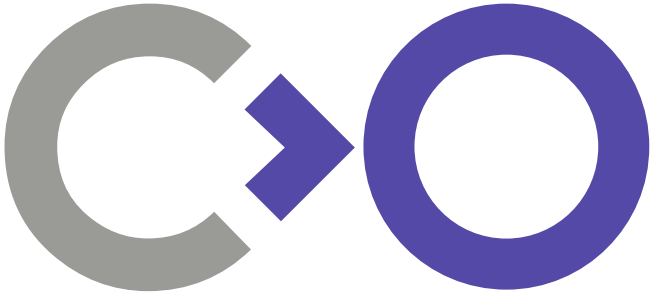
- Continue development of proof of concept
- Reach feature parity with current solution on reference platforms
- Ensure reference compositor works in development environments (QEMU), tested in Fuego
- Make startup reliable: investigate options for better home screen loading
- Integrate new Weston version after upstream improvement
- Integrate reference compositor to replace Weston if able

Next steps: beyond

- Capture requirements from CES 2020 demo
- Participate in requirements development for Jellyfish
- Push improvements to upstream Wayland community
- Ensure OEM HMI customisation points are clearly documented, provide examples
- Helper library API versioning and change process

Work schedule





Thank you!