Creating an Independent AGL Code Base

Revised: April 16, 2015

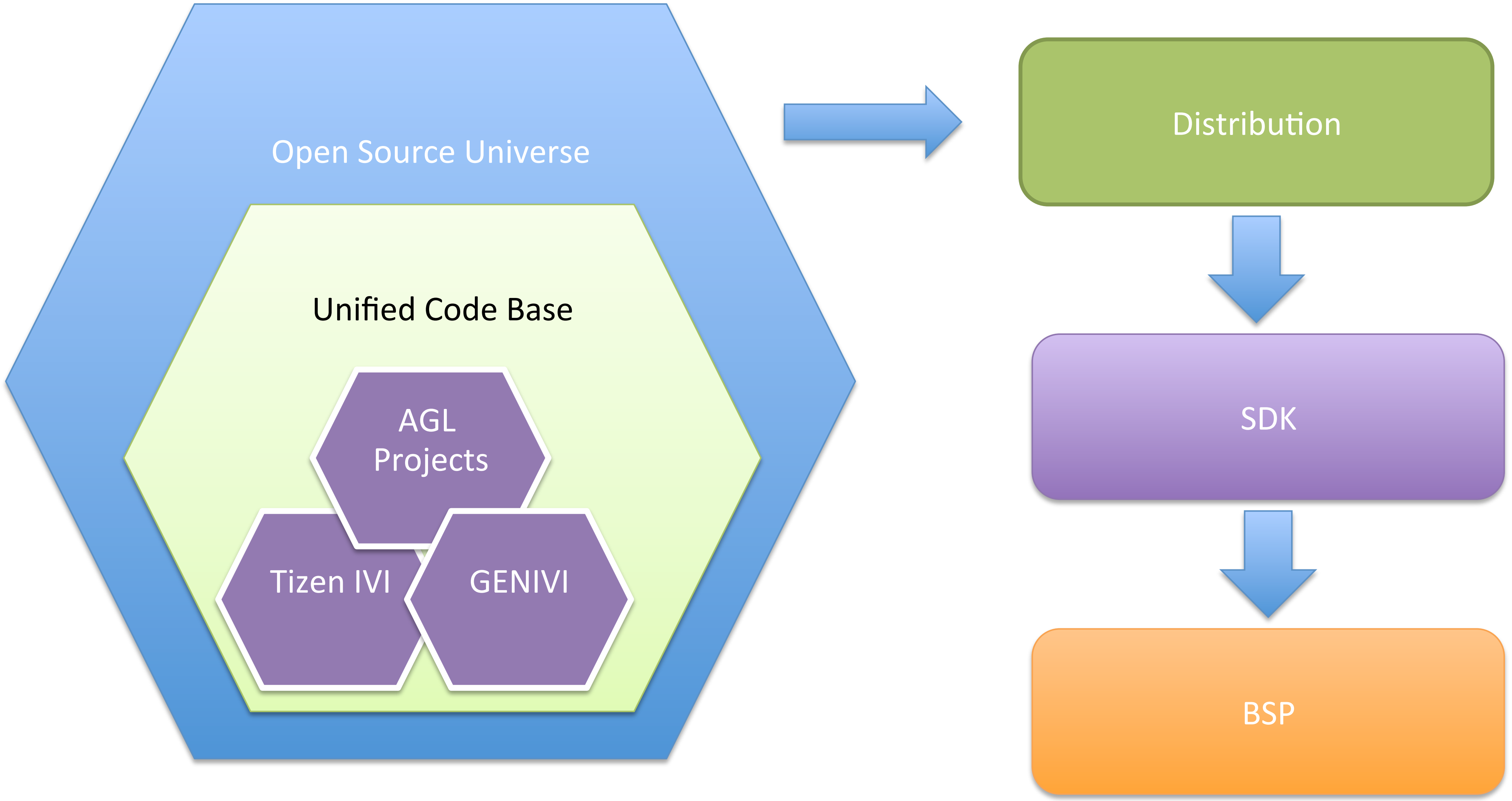
Purpose

- Murata-san from Toyota proposed that AGL host and maintain our own distribution
  - AGL is “code first” – having our own distro strengthens that claim
  - Avoid Tizen and GENIVI fragmentation by merging their code into AGL distro when appropriate

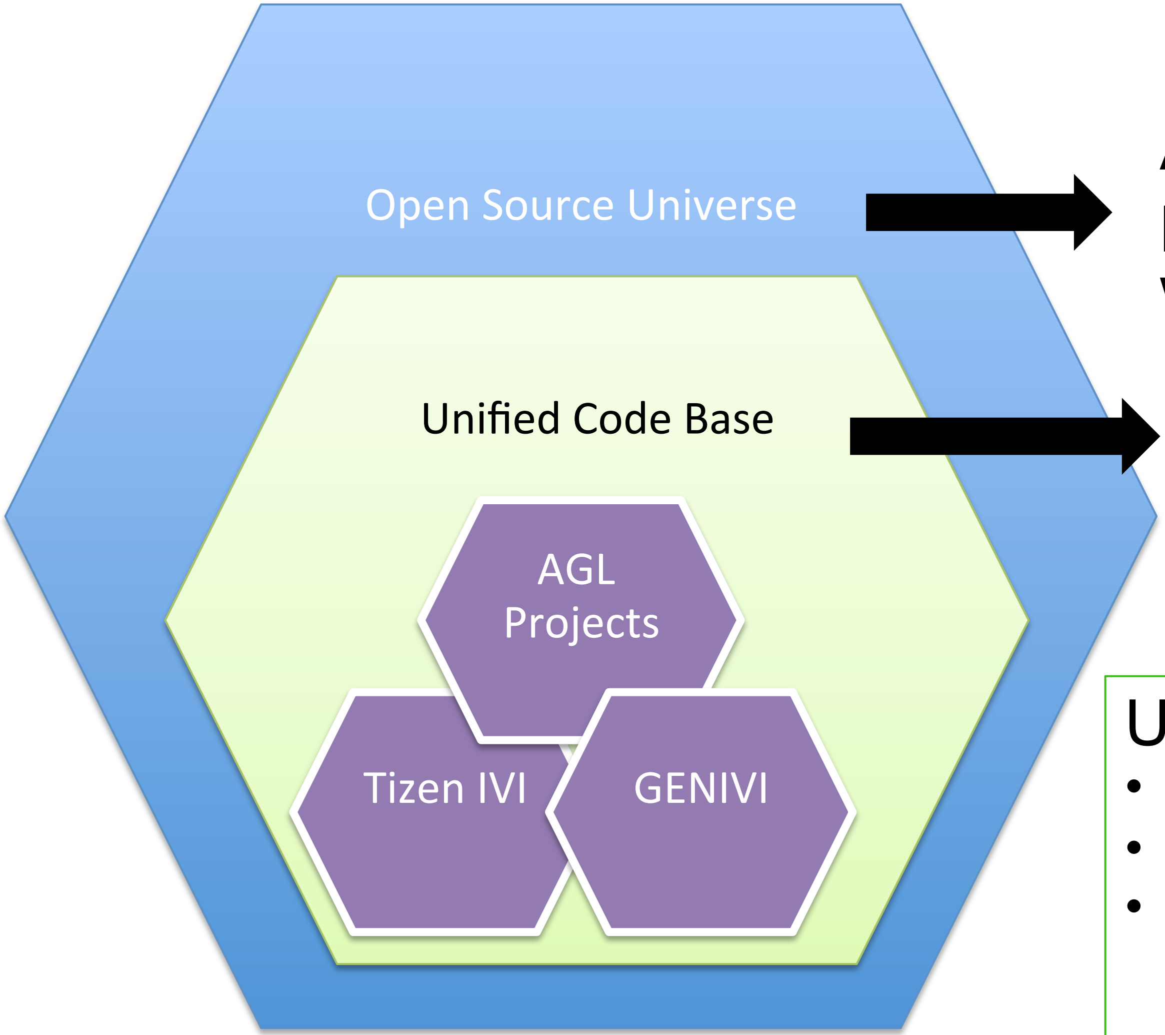
Status

- Action item from AMM to complete definitions of Unified Code Base etc.
  - Collect feedback until 24 April then make proposal for acceptance by SAT and SC
  - Walt to meet with Munakata-san to clarify his input.
- Weekly meetings kicked off. Light attendance.
- Created work-flow slide for getting to the initial distribution.

AGL Work Flow



AGL Work Flow



All of the open source code projects available in the wild!

Subset of open source projects appropriate for automotive

Unified Code Base

- Does not specify package versions
- Meets license requirements
- May have packages that duplicate functionality

AGL Work Flow

Distribution

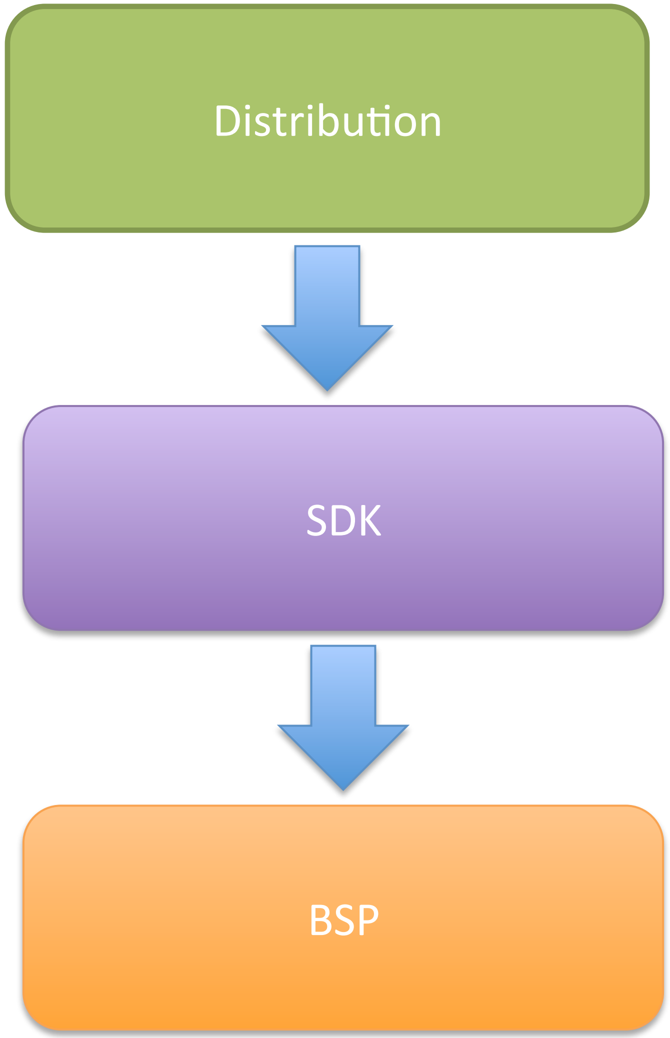
- Specific packages and versions that are built together
- Can be more than a single distribution for AGL

SDK

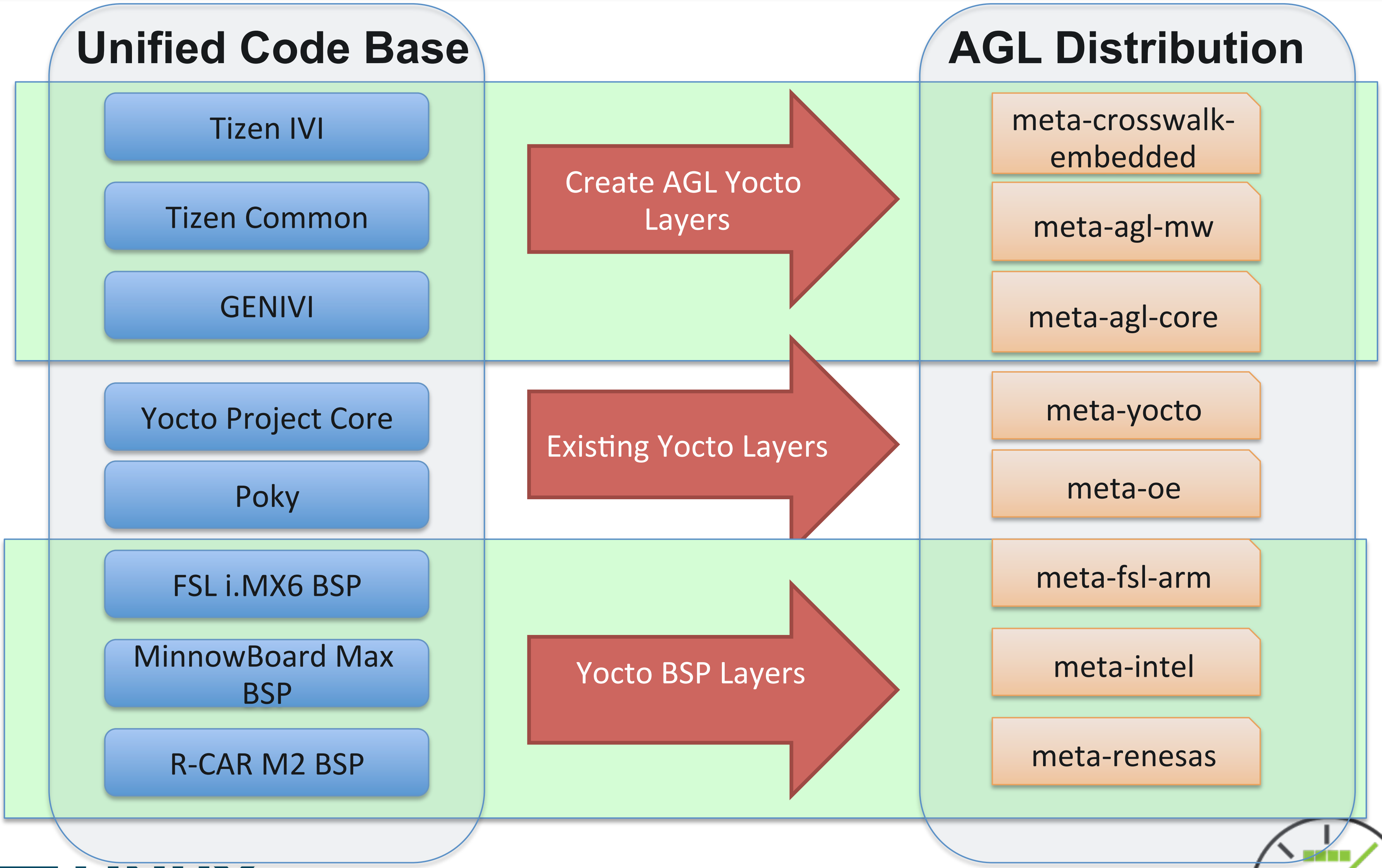
- App framework specified
- Provides common application development environment for product development.
- Multiple BSPs may be bound to an SDK

BSP

- Provided by SOC vendor for an SDK
- Different BSPs may be required for the same board to support multiple SDKs



Getting Started



Unified Code Base

- Definition
  - A collection of open source software that is appropriate for automotive projects
  - Does not specify package versions
  - All packages meet license requirements
  - Allows the possibility of creating different SDKs that support distinct app frameworks and BSPs
  - May have packages that duplicate functionality (e.g., GENIVI Audio Manager and Pulse Audio)
- Process
  - Appoint a group of system architects from all stakeholders (OEM, Tier 1, System integrators, and Silicon Vendors) to evaluate possibilities for the UCB.
    - Invite outside participation but do not make it a limiting factor. (GENIVI, Tizen, others)
    - Non-AGL OEMs

Unified Code Base

- Open Issues/ Questions
  - What form does the UCB take (e.g., web site, database, spreadsheet)
  - Is the Tizen IVI package list the best starting point?
  - How do packages get added to or removed from the UCB?
  - In practice what will the UCB be used for?
  - Need a better name that better reflects what the Unified Code Base Represents

Unified Code Base (Japan input)

Unified Code Base (UCB) is

- a list of packages without determining versions.
- Including only packages that may be used for AGL.
- Comment: UCB is not limited to AGL packages. I think what is intended is to say that AGL is limited to packages in the UCB. The UCB may be used by other organizations as well.
- SAT: Agreed
- ideally including only upstream codes, but only if the upstream codes cannot be used properly, AGL will fork and maintain by AGL. (AGL should discuss whether being included in the upstream is necessary or not)
- Comment: Since the UCB is version independent, requiring code to be upstream is not important for the UCB. This becomes a consideration at later stages in the work flow.
- SAT: Discuss with Munakata-san
- **How to collect UCB packages:**
- Analyzing Worlds major distributions to find our a) commonalities, b) uniqueness
- Analysis further on uniqueness:
- There are various packages that realize same functions, (e.g. security framework, GUI, Sound (Puls audio or ALSA)). AGL may chose one to include in UCB or Not making a selection is an option as well. If we are choosing packages we will do so base on the criteria such as 1) community's activity level, 2) licenses, 3) functions, 3) vulnerability, and other dependencies. This discussion should be done by SAT.
- Compare with Spec. If anything is missing from USB to satisfy the spec, AGL need to develop on our own effort.
- Compare Spec with the real product demands. What's missing need to be reflected to the Spec.

Distribution

- Definition
  - A collection of open source software packages with versions specified
  - Source code and tools with prebuilt binaries for specific processor targets and QEMU
  - User space and kernel built with tools that are built for specific processor targets
  - Allows the possibility of creating different SDKs that support distinct app frameworks and BSPs
  - May have packages that duplicate functionality (e.g., GENIVI Audio Manager and Pulse Audio)
- Process
  - TBD

Distribution

- Open Issues/Questions
  - In principle multiple distributions are possible from the UCB. In practice, the goal of this effort is to come to a common IVI distribution that eliminates the need for Tizen, GENIVI, and AGL to maintain separate code bases.
  - Is this Yocto/Poky with a set of recipes?
  - Who owns and manages the distribution(s)?
  - Release cadence? Yocto is twice per year. Do we need to keep up with every Yocto release?
  - Does the distribution include patches or is it unpatched from upstream sources (not sure this is a reasonable constraint)?

Distribution (Japan Input)

- **Definition of Distribution:**
- Fix version from UCB to build.
- Narrow down package further from UCB (by SAT).
- Build one AGL Reference Distribution.
- There is a possibility of deriving into several different distros from AGL Distribution.
- Comment: Should this read different SDKs?
- SAT: Discuss with Munakata-san
- Ideally, Windows System protocol should be narrowed down at this stage.
- Distribution needs to be actually built and run. It will have to run OK even after applying all packages selected by SAT.
- To confirm the above mentioned point, it is necessary to go through AGL Test Suite (subsequent development).
- Distribution needs to be a form that we can "distribute" .
- Comment: What is meant by "distribute"?
- SAT: Discuss with Munakata-san
- Several years of maintenance is necessary.
- Comment: For every release? Ubuntu has long term support for every fourth release (every two years).
- SAT: Discuss with Munakata-san
- Whether it should be Product ready or not should be discussed.
- Comment: We should describe this in terms of the expected quality level of the distribution and whether the target hardware is close to a production system or not. The expectation for a desktop or server distribution is that it will work with any hardware that is commonly available. Since embedded hardware is very application specific and not commonly available this objective may be hard to meet.
- SAT: Discuss with Munakata-san
- AGL distribution need to be built with Yocto

SDK

- Definition
  - App framework specified
  - Provides common application development environment for product development.
  - Multiple BSPs may be bound to an SDK

- Process
  - TBD

SDK

- Open Issues/Questions
  - Is this generated as a Yocto Application Developer Toolkit ?
  - Who owns and manages the SDK(s)?
  - Release cadence (once or twice per year)
  - Does the SDK include patches or is it unpatched from upstream sources (not sure this is a reasonable constraint)?

SDK (Japan Input)

- **SDK:**
- Distribution after further narrowing down fur the purpose to increase the App/MW that can run commonly. (E.g. Enable Company A, Company B, and Company C to share the same application when a NAVI is developed for AGL.)
- If handled by AGL, every company can share the same application so it will be ideal.
- Initially, keeping GENIVI compatible API (tentatively), and use it as a discussion testbed for AGL.
- SDK also needs to be build with Yocto

BSP

- Definition
  - Provided by SOC vendor for an SDK
  - Different BSPs may be required for the same board to support multiple SDKs
  - Includes board and processor specific patches to the kernel, device, drivers, codecs, etc.
- Process
  - TBD

BSP

- Open Issues/Questions
  - ?

BSP (Japan Input)

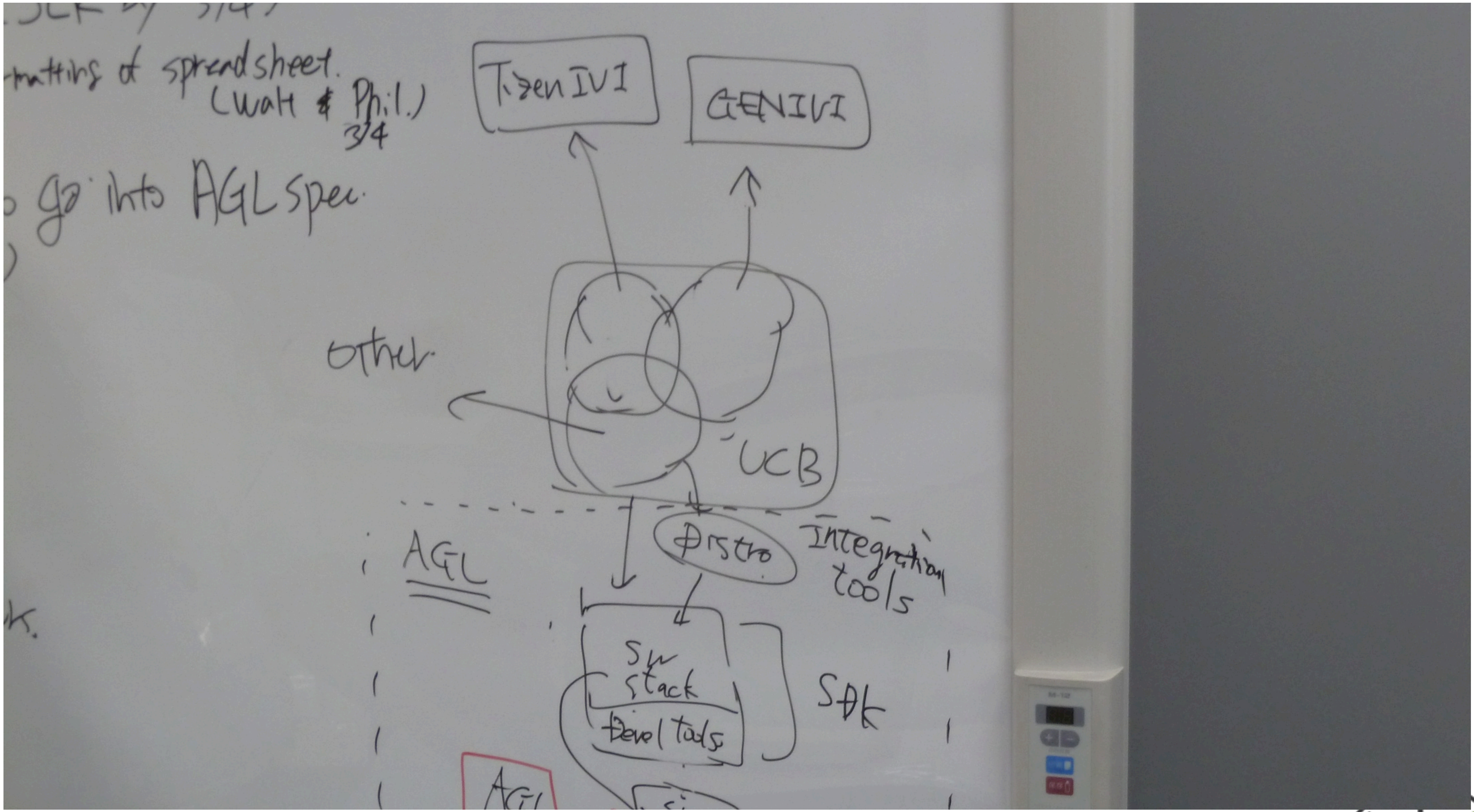
- **BSP:**
- Each chip vendor will develop BSP based on SDK mentioned above.
- BSP ported to each company’s board.
- Build with Yocto
- ======================

BACKUP

Notes from AMM Meeting

- Some issues discussed:
- What needs to be in an automotive code base that makes it compelling for people to use?-
- Unified Code Base (UCB) should look at the existing pieces available in open source and proprietary code and evaluate the gaps between what is out there and what needs to be added to get to a real system available for use.
- Distinction between SDK, distribution, and UCB was discussed. UCB includes the possibility of creating different SDKs that support distinct app frameworks and SDKs.
- During the SAT meeting and the later meeting concerning applications it was agreed that different products may use a variety of application frameworks and that the UCB should not preclude those from being used.
-
- The following action plan was agreed to:
- Write definitions for what each of the products are (unified code base, distribution, sdk, and BSP) in the work flow and get agreement on them
  - Meet in two weeks
  - Matt and Marek to craft proposed starting point
  - Nori to work with Japanese companies to craft a proposal
  - Invitation list shall be limited – Matt asked that this activity be withheld from mail lists/
  - Determine the component list to be included in the UCB
- Appoint a group of system architects from all stakeholders (OEM, Tier 1, System integrators, and Silicon Vendors) to evaluate possibilities for the UCB.
  - Invite outside participation but do not make it a limiting factor. (GENIVI, Tizen, others)
  - Non-AGL OEMs

Notes from AMM meeting



Definitions from Japanese companies

- =======
- **Definition of UCB (=Unified Code Base) :**
- Unified Code Base (UCB) is
- a list of packages without determining versions.
- Including only packages that may be used for AGL.
- ideally including only upstream codes, but only if the upstream codes cannot be used properly, AGL will fork and maintain by AGL. (AGL should discuss whether being included in the upstream is necessary or not)
-
- **How to collect UCB packages:**
- Analyzing Worlds major distributions to find our a) commonalities, b) uniqueness
- Analysis further on uniqueness:
- There are various packages that realize same functions, (e.g. security framework, GUI, Sound (Puls audio or ALSA)). AGL may chose one to include in UCB or Not making a selection is an option as well. If we are choosing packages we will do so base on the criteria such as 1) community's activity level, 2) licenses, 3) functions, 3) vulnerability, and other dependencies. This discussion should be done by SAT.
- Compare with Spec. If anything is missing from USB to satisfy the spec, AGL need to develop on our own effort.
- Compare Spec with the real product demands. What's missing need to be reflected to the Spec.

- **Definition of Distribution:**
- Fix version from UCB to build.
- Narrow down package further from UCB (by SAT).
- Build one AGL Reference Distribution.
- There is a possibility of deriving into several different distros from AGL Distribution.
- Ideally, Windows System protocol should be narrowed down at this stage.
- Distribution needs to be actually built and run. It will have to run OK even after applying all packages selected by SAT.

Definitions from Japanese companies

- To confirm the above mentioned point, it is necessary to go through AGL Test Suite (subsequent development).
- Distribution needs to be a form that we can “distribute” .
- Several years of maintenance is necessary.
- Whether it should be Product ready or not should be discussed.
- AGL distribution need to be built with Yocto

- **SDK:**
- Distribution after further narrowing down fur the purpose to increase the App/MW that can run commonly. (E.g. Enable Company A, Company B, and Company C to share the same application when a NAVI is developed for AGL.)
- If handled by AGL, every company can share the same application so it will be ideal.
- Initially, keeping GENEVI compatible API (tentatively), and use it as a discussion testbed for AGL.
- SDK also needs to be build with Yocto

- ------------------------Above mentioned: Developed at AGL------------------------
- ------------------------Below mentioned: Each developed at SoC------------------------

- **BSP:**
- Each chip vendor will develop BSP based on SDK mentioned above.
- BSP ported to each company’s board.
- Build with Yocto
- =====================

Current Assets

- LF has three servers dedicated to AGL builds
  - Currently set up for OBS builds (1 master and two workers)
  - Master OBS/frontend server is shared with DOORS NG
  - Repurpose OBS worker nodes to run Jenkins for AGL CI
- LF has an AGL Gerrit server in place
- GitHub used on RVI project

Staffing Requirements

- Initial Setup (4-5 FTE)
  – Develop infrastructure requirements
  – Create initial build and test environment
- Once project is up and running (10+ FTE)
  – Program Manager
  – System Administrator (1-2)
  – Upstream maintainers (x2)
  – Kernel developers (x2 for each HW platform)
  – Test Engineers (x2 for each HW platform)
  – Release Engineer
- Staffing level is designed to maintain a current release and not long-term support for releases
- A number of AGL member companies have this expertise in-house
  – Ideally member companies agree to contribute FTEs with expertise
  – Project team will reach out to member companies to identify specific areas of expertise

Thank You