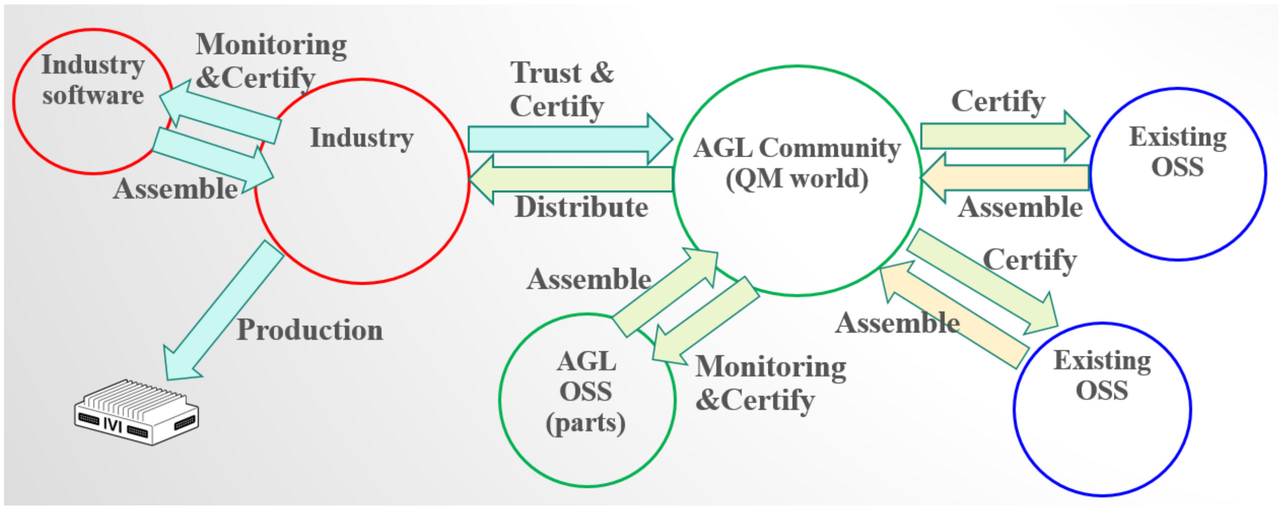
AGL Instrument Cluster Development Process Draft

This development process aim to create workflow from open source development to product development for linux based instrument cluster. This process aim to be able to certify that it has quality control. This process must embrace by open source community and industry.



Attention: This rule applies to Instrument cluster software stack only.

1. Roadmap and Feature Planning

Purpose

- To create the initial time-line for the release: the roadmap.
- To define/update the platform functional requirement.
- To identify the main technical goals for the new release.

Entry Criteria

- **E1-1** Previous release version of the distribution was released.

Tasks

- **T1-1 Create the roadmap and platform functional requirement specification.**

  A detailed description about the rules used to build the roadmap and **platform functional requirement** is in [xx]. These document need to be written in AGL confluence.

- **T1-2 Life-cycle calendar.**

  The following milestones should be determined at this stage. In order to complete each milestone, each exit criteria must be achieved and the deliverable released.

  M1. Roadmap and feature planning complete (section 1).

  M2. Architecture design complete (section 2).

  M3. Detail design complete (section 3).

  M3-1. Software interface freeze.Architecture Design Template

  All interface specification shall final freeze such as interface version, kernel version, compiler version, etc.. Same as current AGL M1 and M2.

  M3-2. Software internal design freeze.

  All software internal design shall freeze.

  M3-3. Demo software freeze.

All demo software (ex. demo ui) shall freeze. The demo software is developed outside this development process.

M4. Unit testing and creating unit test evidence is complete.

M5. Architectural testing and creating evidence is complete.

M6. Release.

M7. End of life.

10 or 5 years after release.

- **T1-3 Determine the new features.**

  On the AGL Instrument Cluster Expert Group the new features expected for this release are discussed and defined. As a result new version of the platform functional requirement specification will be released. This spec must be created before architecture development phase.

Verification

- **V1-1 Review the platform functional requirement**

  The platform functional requirement needs to be approved by at least half of the reviewers.

- **V1-2 Review the roadmap**.

  The roadmap needs to be approved by the Product Manager(Walt?) and the Instrument Cluster Expert Group Leader(Haraki-san?).

- **V1-3 Platform functional requirement and roadmap maintenance**.

  Small changes in the roadmap are not communicated, just executed. But if a delay threatens the release day, an announcement is expected, and if the delay is serious we need to communicate it widely.
  When the platform function is dropped from initial platform functional requirement, it must be approved by at least half of the SAT (System Architecture Team) core members. After that platform functional requirement specification must be update.

Exit Criteria

- **X1-1 The platform functional requirement specification and the roadmap calendar are fixed and froze.**

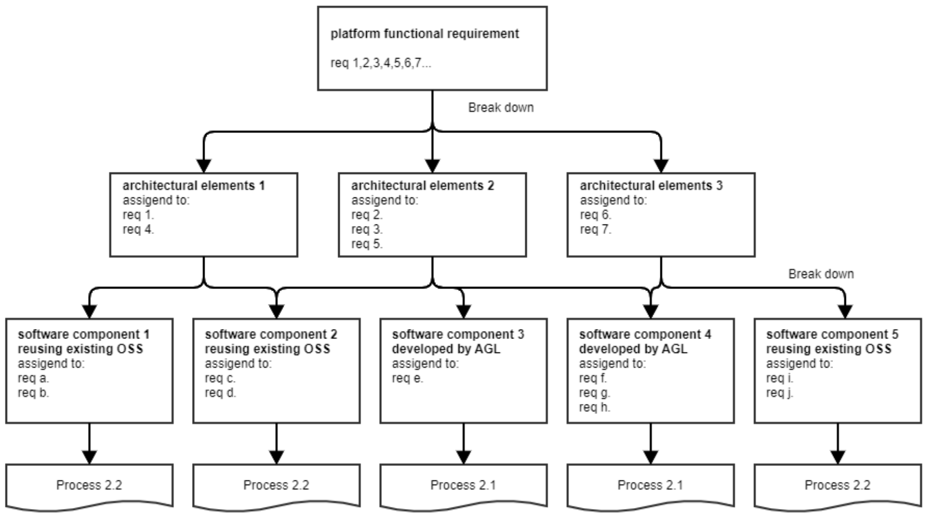
Deliverable

- **D1-1 The platform functional requirement specification is published in AGL confluence.**
- **D1-2 The roadmap calendar is published in AGL confluence.**
- **D1-3 The reason for the change in platform functional requirements is published in AGL Jira. These release information will be published in Confluence at end of development.**
- **D1-4 All review record is published in AGL Jira.**

2. Architecture Design

Purpose

- Identify which software requirements are to be allocated to which architectural elements of the software.
- Evaluate the software architectural design against defined criteria.



Entry Criteria

- **E2-1 Roadmap and Feature Planning phase was completed.**

Tasks

- **T2-1 Create/update software architecture block diagram**

  First step of the architecture design shall create and update software architecture block diagram such as **this**.
  Typically, adding/removing architecture blocks is only allowed by adding/removing functional requirements or refactoring.
  This block diagram need to be written or stored in AGL confluence.

- **T2-2 Create/update software component list and component block diagram**

  The software architecture block diagram shall break down to software component list and component block diagram.
  Element of software component list shall be linking to platform functional requirement for tracing.

  Each software component will be separated and distinguished into AGL development software and existing opensource software.
  The process for AGL development software is set out in Section 2.1. The process for reusing existing opensource software is set out in Section 2.2.

Verification

- **V2-1 Review the software architecture block diagram**
  - The created architecture must conform to the requirements. Shall check for "all requirement is including" and "non requirements component is excluding".
    - ex. afm-binder shall be excluding instrument cluster stack. agl-compositor should be modified based on instrument cluster requirement such as screen layout.
- **V2-2 Review the component list and component block diagram**
  - Should check for "This component need to develop? Difficult to reuse existing OSS?", "This new component is better than current component? Need to replace?" in the review.
    - ex. This new init process is better than systemd?

2. 1. Architecture Design for the AGL development software

Purpose

- Determine the role of that software component in the AGL instrument cluster software platform.
- Determine the criteria for AGL development software.
- The starting point of the AGL development software.

Entry Criteria

- **E2.1-1 The extraction of the relevant software components shall be completed.**

Tasks

- **T2.1-1 Create/update diagram and description of software component from use case point of view**
    - Use case diagram and/or Use case description should be showed in design documents to help understand requirements and to help have common understanding for requirements and,
    - Use case diagram and/or Use case description show necessity of software component to be developed.
- **T2.1-2 Create/update software component interface description**
    - This is important information for the component users, so it shall be included in design documents or described as markdown
    - Developer shall define and describe meaning, parameter and return value of each interface
    - If the order of calling interface function/method is important, we recommend that developer create a sequence diagram.
- **T2.1-3 Create/update software component activity diagram**
- **T2.1-4 Create/update software component state machine diagram and table**
    - If the software component to be designed has a state, state transition should be represented by state machine diagram and/or state machine table.
    - To prevent omission of consideration during design process, we recommend that developer checks his design using state machine table.
- **T2.1-5 Create/update deployment diagram**
    - Developer should clarify related and/or interfering other components or hardware devices to show necessity of these.
- **T2.1-6 Create/update software component test specification**
    - To Design test cases to meet the requirement coverage and
    - To describe these as specification document.

Strong recommendation is to use UML for writing diagrams which are mentioned above.

See Architecture Design Template for the AGL development software **for the document template.**

Verification

- **V2.1-1 Review the each diagram and description**
    - Two or more reviewers must approve this design.
    - This document must be including description for tasks from T2.1-1 to T2.1-6.
    - All review records must be written in AGL Jira.

- **V2.1-2 Review the interface description**
    - The method of providing an interface should use for standard method of the own architectural layer.
    - Interface spec shall describe by standard format of the method.
      ex. In case of C functional interface should be use doxygen. In case of REST interface should be use OpenAPI.
    - All review records must be written in AGL Jira.

- **V2.1-3 Review the test** .
    - The correct and abnormal behavior of all interfaces must be defined.
    - The performance requirements in the reference environment must be specified. ex. Interface A : response tome less than 1ms in R-Car M3.
    - The resource requirements in the reference environment must be specified with limitations. ex. Runtime memory usage less than 20MByte in R-Car M3 (at 2 Guest container).
    - All review records must be written in AGL Jira.

Exit Criteria

- **X2.1-1 Completed review and accepted the documents for software component.**

Deliverable

- **D2.1-1 The software component use case diagram and description, activity diagram and description, state machine diagram and description and interface description is published in AGL confluence.**
- **D2.1-2 All review record is published in AGL Jira.**

2.2. Architecture Design for the reusing existing opensource software

Purpose

- Determine the role of that software component in the AGL instrument cluster software platform.

Entry Criteria

- **E2.2-1** The extraction of the relevant software components shall be completed.

Tasks

- **T2.2-1 Create/update software component use case diagram and description**

  Create a use case document to show why the that OSS is needed. This document shall be including all of assigned platform functional requirement.

- **T2.2-2 Assess to the reusing existing opensource software**

OSS assess and certify based on the AGL Instrument Cluster Distribution Criteria.

- **T2.2-3 Determine a major version of the OSS**

  The major version of the OSS shall be selected with a maintenance lifecycle that meets the roadmap creating in section 1. When it OSS is not provided Long Term Support/Stable by upstream, should select downstream that is maintained by distribution.

  In case of downstream maintenance, AGL shall select and refer to downstream of one distribution (RedHat/CentOS or Debian or Ubuntu or OpenSUSE or other), not select Yocto code base.

Verification

- **V2.2-1 Review the use case diagram and description**
  - Two or more reviewers must approve this definition.
  - This document must be including description for tasks from T2.2-1.
  - All review records must be written in AGL Jira.
- **V2.2-2 Review the assessment result**
  - The selected OSS must meet the assessment criteria.
  - Risk information for the selected OSS must be provided. It should be updated during the detailed design phase.
  - All review records must be written in AGL Jira.
- **V2.2-3 Review the test** .
  - Selected OSS have test suite that fit the use case is available.
  - and/or the test cases is defined based on the defined use case.
  - The performance requirements in the reference environment must be specified.
    - ex. AES encryption processing : more than 10 MByte/sec in R-Car M3.
  - All review records must be written in AGL Jira.

Exit Criteria

- **X2.2-1 Completed review and accepted the documents for software component.**

Deliverable

- **D2.2-1 The software component use case diagram and description is published in AGL confluence.**
- **D2.2-2 Assessment documents published in AGL confluence.**
- **D2.2-4 All review record is published in AGL Jira.**

3. Detail Design and Code Development/Certification for existing OSS

3.1 Detail Design and Code Development

Purpose

- Develop software component which is specified in Architecture document.

Entry Criteria

- **E3.1-1** AGL development process section 2.1 has been completed.

Tasks

- **T3.1-1 Create source code**
  - That source code must be able to cross building
    - If that code include assembler code, must include generic implementation code for other architecture.
  - That source code must conform to the coding rules.
    - AGL must define coding rule. Coding rule is refer to systemd Coding Style.
      - https://systemd.io/CODING_STYLE/
  - External interface spec shall describe by standard format of the method in that source code.
    - ex. In case of C functional interface should be use doxygen.
  - Internal interface spec should describe by standard format of the method in that source code.
- **T3.1-2 Scan developed code using static analysis tool, findings from tools should be fixed.**
  - When static analysis tool detected "Must Fix" error, developer must fix this error.
  - When static analysis tool detected "Need Evidence" error, developer must create "no problem evidence".
    - Static analysis tools are subject to many miss detection. Developers need to show that it is miss detection.
- **T3.1-3 Check for coding rules by static analysis tools**
  - Code developed in AGL must conform to the AGL coding rules.
  - AGL coding rule is refer to systemd Coding Style.
    - https://systemd.io/CODING_STYLE/
- **T3.1-4 Create release note**

Verification

- **V3.1-1 Review the detail design and source code**
    - Two or more reviewers must approve this design.
    - Two or more reviewers must approve this source code.
    - All review records must be written in AGL Jira.
- **V3.1-2 Review the assessment result**
    - Asses to static analysis result and "no problem evidence".
        - That evidence must be written in AGL static analysis infrastructure.
    - All review records must be written in AGL Jira.
- **V3.1-3 Review the test** .
    - That code need to include test suite. It need to fit the use case.
        - It does not have to be in sync on a per-commit basis at all times.
    - All review records must be written in AGL Jira.

Exit Criteria

- **X3.1-1 Completed review and accepted the documents.**
- **X3.1-2 Completed code development.**
- **X3.1-2 Completed Yocto recipe development.**

Deliverable

- **D3.1-1 The software component use case diagram and description is published in AGL confluence.**
- **D3.1-2 Assessment documents published in AGL confluence.**
- **D3.1-3 All review record is published in AGL Jira.**

3.2 Certification and importing for existing OSS

Purpose

- Create, update or fix packages/recipes.

Entry Criteria

- **E3.2-1** AGL development process section 2.2 has been completed.

Tasks

- **T3.2-1 Assessment for the change logs (release note, commit log , etc.)**
    - In initial phase, should update existing OSS frequently. In late phase and maintenance phase, should not update existing OSS frequently excluding to security fix and critical bug fix.
    - Shall check to change log, because we need to judge "this change is need in this phase?".
    - Create or update to the document of assessment results.
        - Template: Template for Detailed criteria of the reusing existing OSS
- **T3.2-2 Source code check by static analysis tools**
    - Must check to source code using static analysis tool.
        - When static analysis tool detected "Must Fix" error, that OSS must not use.
        - When static analysis tool detected "Check by AGL" error, that OSS code must check and judge by AGL community before than adopting this OSS.
        - When static analysis tool detected "Check by User" error, developer shall create error report only. AGL community provide risk information for that OSS to user.
    - Checker rule as a follow:
        - https://confluence.automotivelinux.org/download/attachments/26968229/clang_code_checker.xlsx?api=v2
- **T3.2-3 Check for Software interface specification**
    - Check the source code and library software interface specifications provided by OSS.
    - The important point is to check compatibility from previous version. When current phase is after M3-1, must not change to incompatible version.
    - Create the document of check results or write in the release note.

Verification

- **V3.2-1 Review the assessment result**
    - Two or more reviewers must approve this assessment.
        - That evidence must be written in AGL static analysis infrastructure.
    - All review records must be written in AGL Jira.
- **V3.2-2 Review the test**
    - All review records must be written in AGL Jira.

Exit Criteria

- **X3.2-1 Completed review and accepted the documents.**
- **X3.2-2 Completed code development.**

- **X3.2-2 Completed Yocto recipe development.**

Deliverable

- **D3.2-1 The document of assessment results.**
- **D3.2-2 New or updated Yocto recipe**

4.Test

4.1 Test for AGL Development Software

- Refer to: Test criteria for the AGL development software

4.2 Test for Existing OSS

- Refer to: Sub document for the AGL Instrument Cluster Development Process