Support specific operational design domains (ODDs)

- ODDs defined in part using parameterised scenarios

- Target five user types
  - Students
  - Researchers
  - Engineers building PoCs
  - Engineers building products
  - Engineers building an Autoware component

- Utilise ROS software packaging and project compositing capabilities

- A “foundational Autoware” that provides the essential functionality necessary for autonomous driving with a focus on high quality software that proves the interfaces

- An “extended Autoware” that builds on the core and provides additional or newer algorithms, but with less quality assurance
Caveats

- Focus of this document is on the Autoware software
- A goal to aim for, not an intermediate target
- Focus on the “ideal” Autoware - no distinction between Autoware.AI and Autoware.Auto
Development approach

- **ODD-focused development**
  - Incrementally achieve additional ODDs
  - Starting simple and achieving increasingly-complex ODDs
  - Having an end-to-end stack in a simple ODD sooner is the priority
  - Each ODD contains a number of required scenarios and an environment definition
  - ODDS chosen by the AWF TSC

- **Scenario-driven development**
  - Specific manoeuvre the car must perform combined with the environment in which it is performed
  - Scenarios are parameterised (weather, time of day, traffic conditions, etc.) to enable combinatorial testing
  - Scenarios defined in a machine-readable format (e.g. OpenScenario) to enable automated testing

- **Demonstrations as use cases**
  - One or more demonstrations per ODD
  - Demonstrations used to drive application-level requirements
  - Demonstrations provided as part of Autoware for use by anyone - demonstrations show how to use Autoware "properly"
Target user types

- **Student**
  - Wants to use Autoware as-is
- **Autonomous driving researcher**
  - Wants to replace specific components with their own implementations, and use the most recent advances
- **Engineer developing a Proof-of-Concept/prototype**
  - Wants to build applications on top of Autoware, may want to use the recent advances for specific parts
- **Engineer developing a product**
  - Wants to build applications on top of Autoware, but stability and reliability is more important than being state-of-the-art
- **Engineer developing an Autoware component**
  - Wants to replace a specific component of Autoware with their own commercial implementation
- **(Autoware developer)**
  - (Wants to alter the core functionalities of Autoware to enable a new ODD/scenario)
How to build applications in ROS

- “Federated” model of software management
- Application made up of packages from many sources, including
  - Other developers
  - The application's creator
- Packages used by an application managed via one or more VCS files
- Application developed in the equivalent of a virtual environment
  - cf. Docker, Python venv, ...
Autoware

Autoware Core

- Provides a complete end-to-end software stack
- Focus on stability and quality over cutting-edge
- Contains the interface definitions that define “Autoware” as a framework
- Managed by the Autoware Foundation
- **Analogy:** Debian Stable release, Ubuntu Main repository

Autoware Universe

- Provides additional packages that can work as part of Autoware
- Lower (but not zero) quality requirement
- Must be compatible with Autoware Core
  - Not a parallel Autoware
  - Cannot be used independent of Core
- Packages can graduate to Core
- Managed by the Autoware Foundation, but more dependent on the community
- **Analogy:** Debian Testing release, Ubuntu Universe repository
Do not think:
- “Autoware Core is Autoware.Auto and Autoware Universe is Autoware.AI”
- “Autoware Core is just a renamed Autoware.Auto”
- etc.

Together, Core and Universe form a single Autoware
- In keeping with this, we must produce a fully-functional Core before there is any point in doing anything in Universe

Autoware Core and Autoware Universal are where we want to be
- Whether we get there from just Autoware.Auto, just Autoware.AI or a mix is not important to the definition of what they are

We need to decide a strategy for how to get to there
- When to formally create Autoware Core and Autoware Universe
- If and when to end support of Autoware.AI
- What to do with existing Autoware.AI code
- What to do with Autoware.AI between now and its end-of-life (if we end it)
Routes to Autoware Core and Universe: Option #1 of many

- **Autoware.AI**
  - Selective porting

- **Autoware.Auto**
  - Use as-is + updates

- **Autoware Universe**
  - Newly created repositories, etc. as needed

- **Autoware Core**
  - Use as-is + updates
Routes to Autoware Core and Universe: Option #2 of many

Autoware.AI

Selective porting

Autoware Universe

Newly created repositories, etc. as needed

Autoware.Core

Selective porting

Use as-is + updates

Autoware.Auto
Routes to Autoware Core and Universe: Option #? of many

Autoware.AI

Autoware.Auto

Please suggest

Autoware Universe

Autoware Core

Newly created repositories, etc. as needed
Obtaining and using Autoware

- **Autoware (Core or Universe)** can be installed from binary packages or source
  - Installing from source utilises VCS files
  - Either approach supports all uses except where the original Autoware source must be altered

- **Using stable Autoware as-is and build applications on top of it**
  - Create new packages that contain application functionality, and *depend on and use* Autoware nodes, resources, etc. from the Autoware packages

- **Using recent advances in Autoware**
  - Create new launch files that launch nodes and use resources from Autoware Universe in place of those from Autoware Core

- **Replace parts of Autoware with alternate implementations (research, commercial, etc.)**
  - Create new packages providing the alternate implementations
  - Create new launch files that launch nodes and use resources from these new implementations in place of those from Autoware Core or Universe
Development roadmap

● Cycles of development with specific focus
  ○ e.g. Specific ODD
  ○ e.g. Cleaning up technical debt, polishing documentation

● Current planned cycle goals:
  ○ Cycle 1 (early 2020): Autonomous Valet Parking
  ○ Cycle 2 (mid 2020): Architecture clean up, technical debt clean up, documentation clean up
  ○ Cycle 3 (late 2020-early 2021): Next ODD
  ○ Cycle 4: Technical debt cleanup