Table of Contents

F.1 Introduction

This document presents the architectural design of the system and contains designs and decisions that relate to the system described in the vision, requirements, and supplementary specification.

F.1.1 Definitions and abbreviations

See project dictionary.

F.2 Architectural Factors

This sections purports to record the architectural decisions made and the reasoning behind.

Technical Memo

F.2.1 Issue: Maintainability – Rapid reconfiguration of display layout

Solution Summary: Generated layout-configuration file in JSON format from GUI-tool.

Factors:

- Generated configuration-file avoids human errors in configuration.
- Configuration is verified by GUI-tool.
- Reduced time spent on configuration
- Human readable format with little overhead.
- Pre-made widgets allow for easy placement and configuration of variables.

Solution

Achieve reduced cost in effort and time used on configuration by developing a GUI tool that generates a configuration file that can be uploaded to the DIS. Using a widget-solution allows for experimentation with different presentations of various sensors, as well as making it easier to find configurations that fit together.

Motivation

We don't know which design and layouts will give positive results, and developing this tool and specifying a format contract beforehand reduces time spent debugging and makes it easier to quickly test concepts beforehand.

Unresolved Issues

- Contract for format has to specified.

Alternatives Considered

- Manual configuration of files. Takes a lot of time to write by hand, needs testing to see if they work.
- Using XML instead of JSON. Has larger overhead and more difficult to parse.
- Hard coding configuration in software. Gives little flexibility, might increase performance.

Technical Memo

F.2.2 Issue: Maintainability – Configuration of sensor calibration without compilation of code

Solution Summary: Using GUI-tool and reconfigurable pipe and filter pattern to make configurable transformation filters

Factors:

- Generated configuration-file allows verification before runtime.
- Generalized pipe and filter pattern allows pipeline transformations of data.
- Filter-pipeline allows for easy debugging of transformations.

Solution

Achieve reduced cost in effort and time used on configuration by developing a GUI tool that generates a configuration file that can be uploaded to DIS, it is also possible to do this in configuration files, but format might be complex due to mathematical notation. GUI tool could also provide visualization of transformations.

Motivation

Sensor data needs to transformed to meaningful real-world data so that it can be visualized. This fits very well with the pipe and filter pattern and data can be transformed with minimal latency.

Unresolved Issues

- Contract for configuration format has to be specified.
- Is the benefit of making a GUI-tool great enough to be worth the extra effort?
- Will the overhead from running potentially long and concurrent pipelines result in unacceptable latency

(ie. do we have enough computing resources to allow for this type of reconfigurability)?

Alternatives Considered

- Manual configuration of files. See "Solution"
- Hard coding configuration in software. Gives little flexibility, might increase throughput and decrease latency.

Technical Memo

F.2.3 Issue: Maintainability – Rapid reconfiguration of sensor profiles

Solution Summary: Using abstraction and a layered architecture to defer network package specification to higher levels of software and allow for configuration files specifying package format.

Factors:

- Layer pattern allows compartmentalizing functionality and deferred configuration.
- Increases modularity and reusability.
- Increases overhead related to network computing.
- Increases system complexity
- Allows for easily using multiple package formats within the same setup.

Solution

Develop a format specification for network packets decomposition and composition to allow high level configuration of low-level network systems. Using this approach makes it possible to easily adapt the system to other vehicles or if the package format should change. It also makes it easier to support different network standards as well.

Motivation

Making it easier to port the system to future vehicles and use other network standards.

Unresolved Issues

- Format for configuration file.

Alternatives Considered

- Hard coding configuration in software. Gives little flexibility, might increase throughput and decrease latency.

Technical Memo

F.2.4 Issue: Performance – Processing and displaying data with minimal latency

Solution Summary: Lowering the level of abstraction in time-critical code and reducing the number of layers where layer pattern is applied. Keeping length of pipelines to a minimum where pipe and filter pattern is applied.

Factors:

- Layer pattern increases overhead and lowers performance.
- Long pipelines increases overhead.
- High number of pipelines increases overhead.
- Tailored filters can reduce overhead by replacing multiple filters with a single one.
- Direct access to lower layers for certain functionality can increase performance where functionality is being used directly.
- Optimizing mathematical functions can greatly increase speed as both multiplication and division typically requires considerably more compute time than addition and subtraction.

Solution

Start early with performance testing and use automated performance testing code commits. Performance also needs to be tested on actual hardware as soon as possible. Reducing the level of abstraction

will also result in increased performance, and complex calculations need to be optimized for the computation environment.

Motivation

Keep the system useful by providing data in real-time.

Unresolved Issues

none

Alternatives Considered

- Performance bottlenecks can be overcome by using more powerful hardware. - Splitting calculations between computing nodes can also result in lower latency.