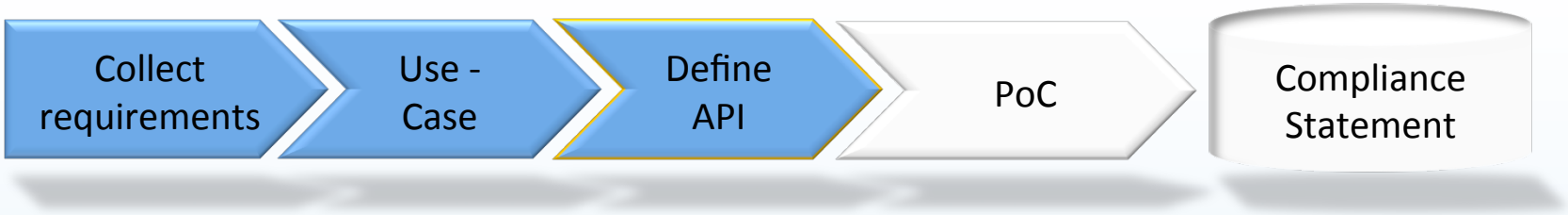
Introduction of GENIVI Speech Services

W3C Face to Face in Santa Clara, CA
March 17 – 18

Mario Thielert, Continental Automotive AG
Dominique Massonié, Elektrobit Automotive GmbH

Status of Speech in GENIVI



Description:

An application can assume a standard interface to implement a speech dialog as well as to output speech. GENIVI application cores and external apps can rely on standard interfaces towards speech stacks.

Scope:

Identify requirements towards an unified Interface for speech components in the system, GENIVI Speech APIs, Integration of speech recognizer & TTS engines, identification of standards for resources (like phonetic alphabets etc.) .

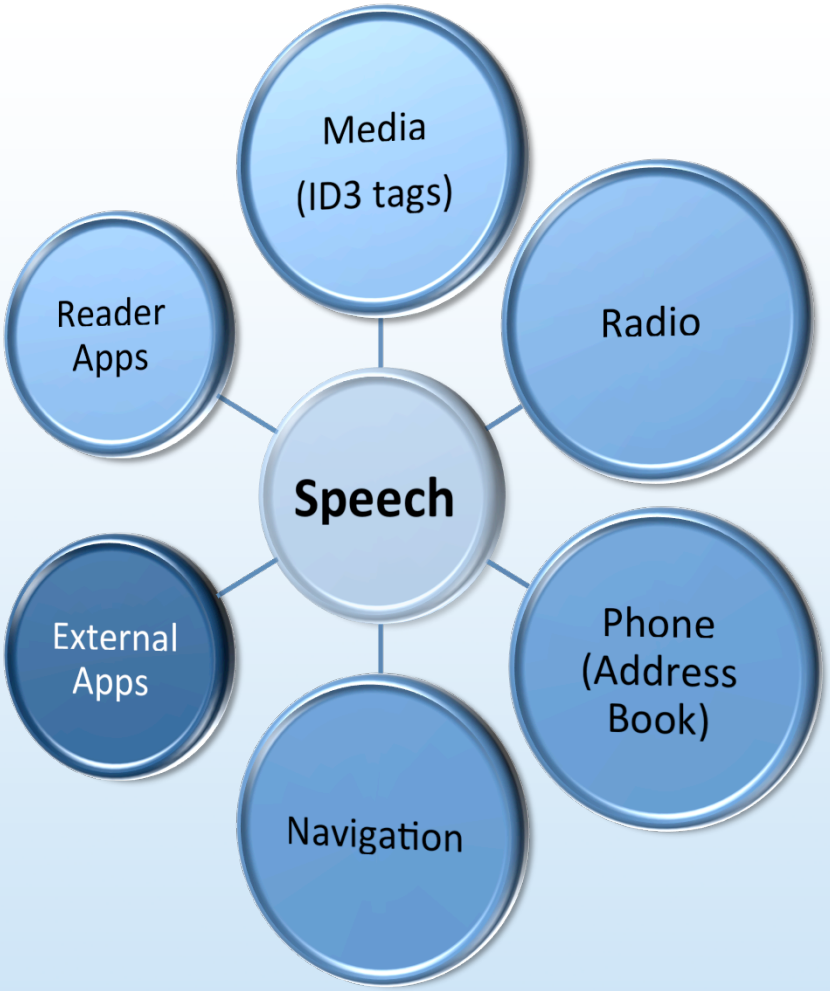
Responsibles:

David Kämpf (Subproject Lead), Continental Automotive

Mario Thielert, Continental Automotive

Dominique Massonié, Elektrobit

Basic Speech Architecture
Relations to other Areas



Application Cores may

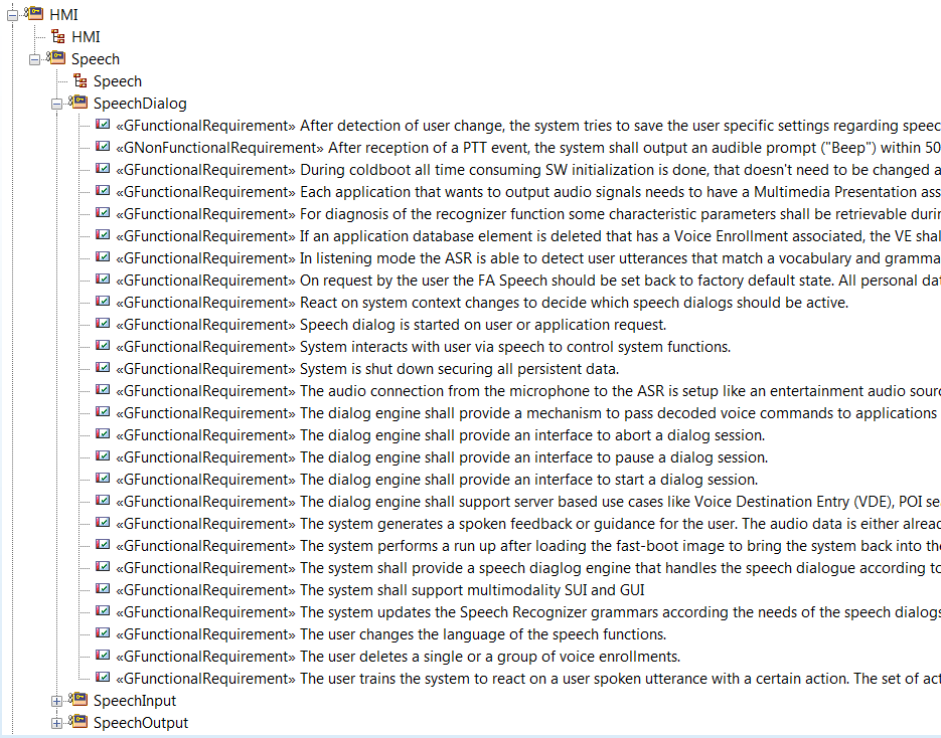
- provide data that will be included in dynamic grammars
- generate prompts that will be spoken by the TTS engine

External Apps may

- register app specific dialog/ content
- react on dialog steps
- generate prompts

Status of Speech @ HORIZON Requirements

- We have only collected requirements in the Speech Area that
  - …capture non differentiating aspects
  - …are not specific for a product segment (e.g. high end)
  - …capture KPIs only where usability is affected



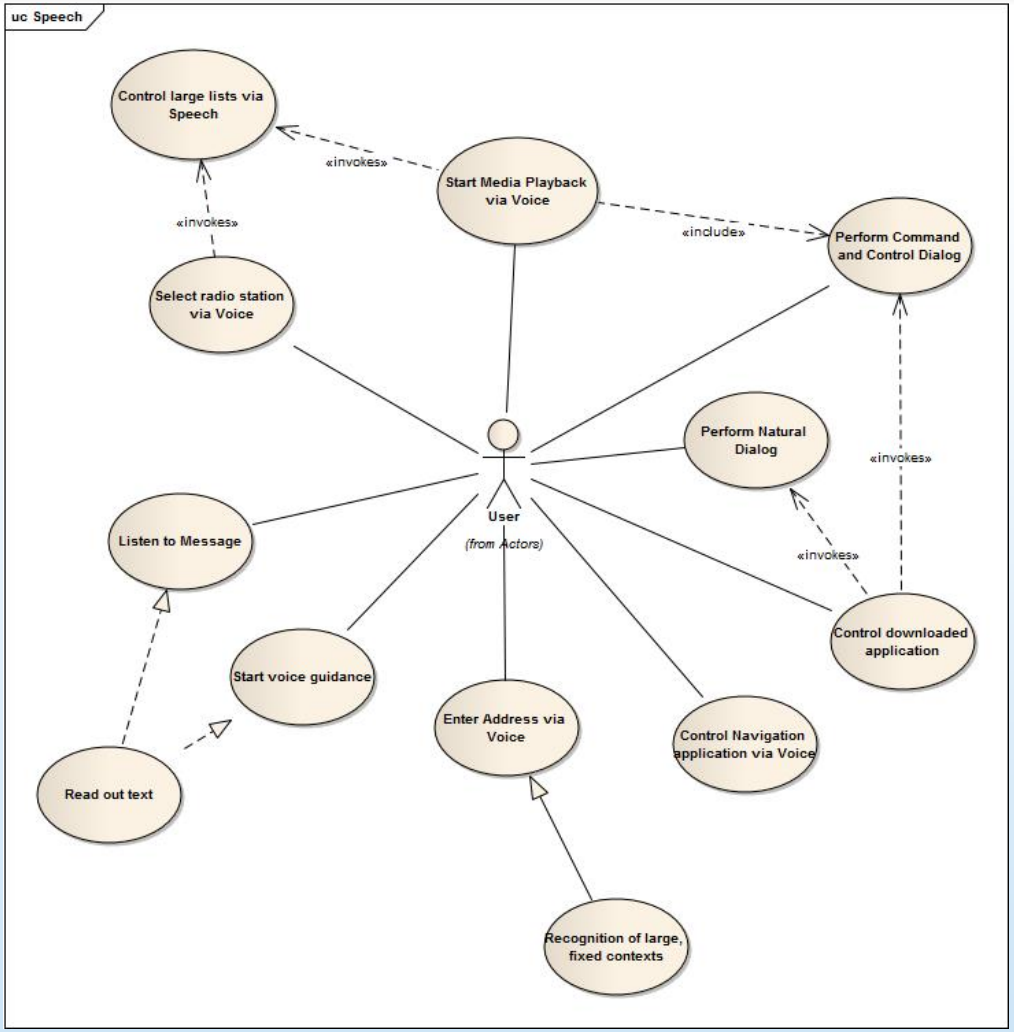
Additional information on the Speech requirements can be found in

- UML model
- Compliance Document

https://collab.genivi.org/wiki/display/genivi/Compliance+Team#ComplianceTeam-Compliance5.0DraftDocuments

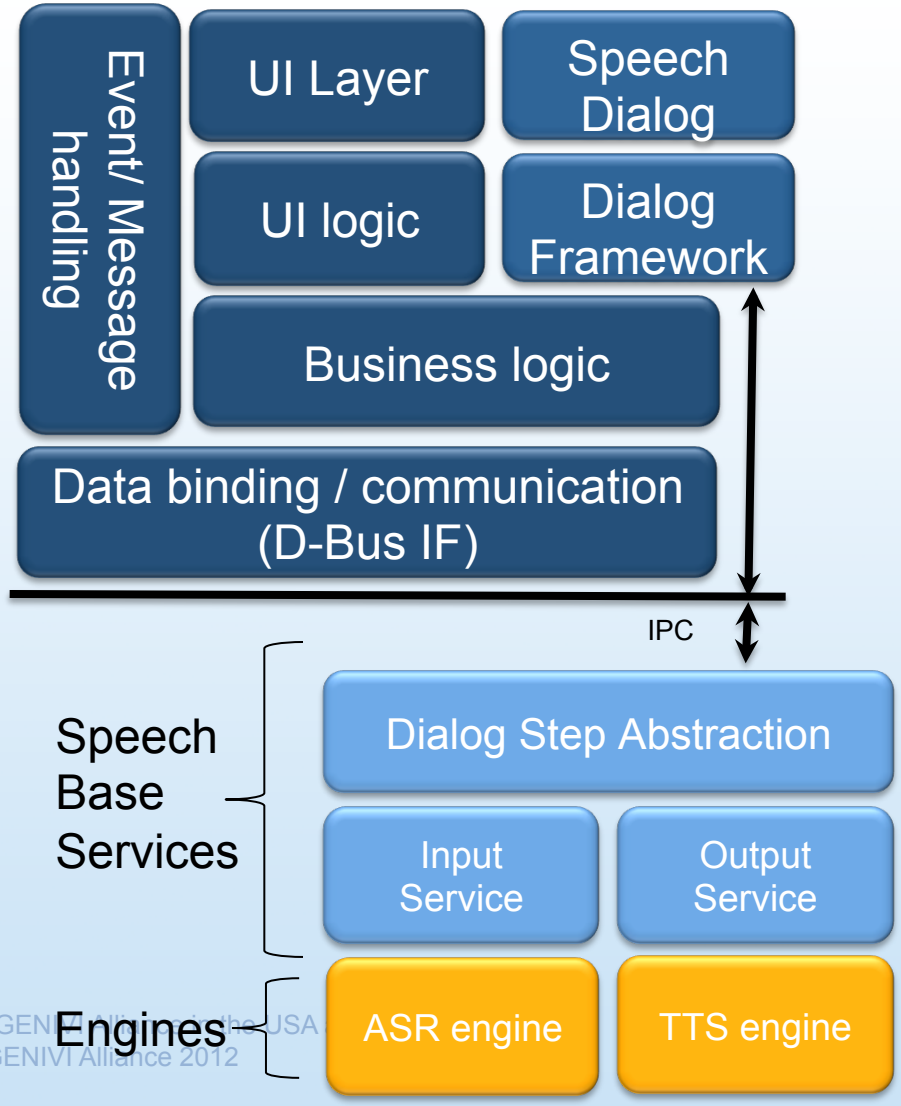
Status of Speech @ HORIZON Use Cases

- Speech is a P2 Placeholder component
- Use Cases are defined that cover:
  - Core App
    - Radio
    - Media
    - Navigation
    - Phone
  - Speech Dialog
  - NLU
  - Server based Reco.

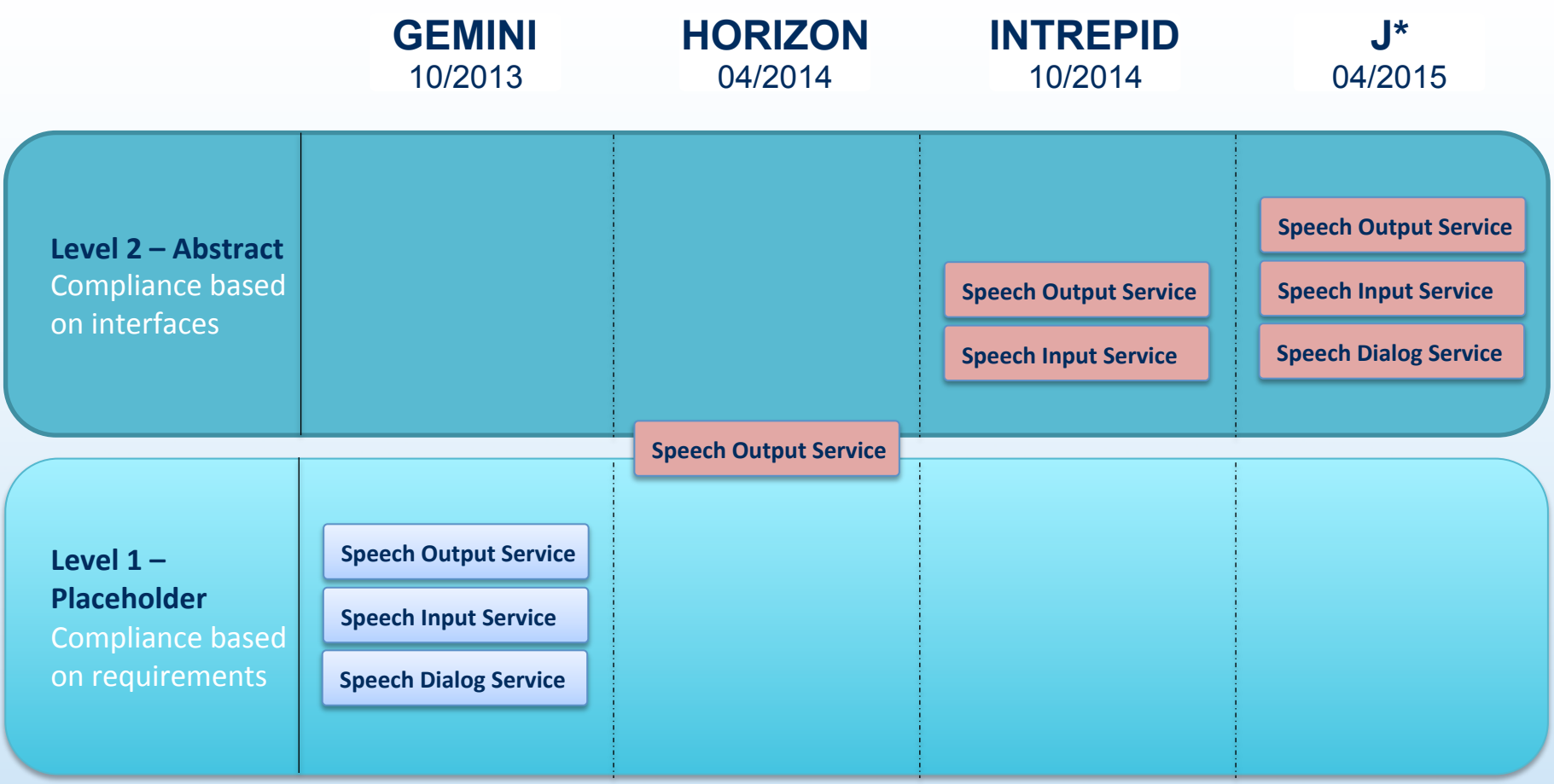


Basic Speech Architecture
Speech Software Stack

- **Speech Dialog**
  - Modeling the User Interaction
  - Handling of resources
  - Interact with GUI
  - Interact with Business Logic
- **Dialog Step Abstraction**
  - Modeling one dialog step
- **Speech Input Service**
  - Integration of the Speech Recognizer (ASR engine)
  - Resource handling
- **Speech Output Service**
  - Accessing TTS functionality
  - Integration of TTS engine



Speech Roadmap



EG HMI Subproject Speech Status Summary

- Status of Speech @ HORIZON (04/2014)
  - Speech Uses Cases defined (Output, Input, Dialog)
  - Speech Requirements defined (Output, Input, Dialog)
  - Basic Speech Architecture defined
  - Defined and agreed Speech Output Service API
- Next Steps
  - Proof of concept for Speech Output Service API
  - Define Speech Input Service and Speech Dialog Service APIs
  - Define Interfaces towards application cores
    - Dynamic data (e.g. media ID3 tags, phonebook contacts, station lists etc.)
    - Navigation address data

Why should you care about GENIVI Speech API ???

- **GENIVI API based on an API that is used in multiple customer products**
  - Covering the automotive use cases
  - Proven and mature API
- **Benefit of a GENIVI API for the „Speech Output Service“**
  - Session management to support multiple concurrent clients (prioritization, audio connection handling etc.)
  - Prompt preparation for low latency playback
- **Benefits of a GENIVI API for the „Dialog Step Abstraction“**
  - Reduces the effort to implement a dialog
  - Basis for App Development

Next Steps for W3C and GENIVI

- Converging W3C / GENIVI Speech APIs
  - Adding automotive capabilities to the W3C proposals
  - GENIVI could support out-of-the-box W3C standard
- Shared development effort / Joint Meetings

- Open Questions about Speech Standardization
  - ASR grammars (NLU, server based, word lists, …)
  - Phonetic Alphabets and Transcription mechanism
  - Leverage W3C Markup Languages (VoiceXML, SSML, SRGS…)

BACKUP

Speech Output API – Reader Interface

- Applications can read out text easily
- Connection Handling is taken care of by the TTSReader
- Applications have to provide:
  - text or multiple text chunks to be spoken
  - an application priority
  - a context ID that identifies the domain of the text being spoken (e.g. Navi, E-Mail etc)

```
class TTSReader {
      signals:
          void notifyConnectionStatus(const
              TTSAppPrompterConnectionStatus
              eConnState);
          void notifyPrompterStatus(const
              TTSAppPrompterStatus
              ePromptState);
          void notifyChunkMemorySize(const Int
              memSize);
      public slots:
          void openPrompter(const TTSAppPrio
          ePrio, Int ctxId);
          void closePrompter();
          void abortPrompter();
          void addTextChunk(const QString
          chunk);
          void addTextChunk(const
          QVector<QString> chunks);
          requestChunkMemorySize(void);
}
```