Application FW Update

Oct 19, 2016

Guru
Architect, GENIVI Alliance

Gunnar
SAT Lead, GENIVI Alliance

Content

- Application FW
- App Manager
- App Manager POC1 – Apertis
- Working Session

GENIVI®

Application FW

- Application framework consists of a set of software components
  - Which enables interaction of Apps with the underlying GENIVI platform
  - App Download and Install management
  - Defines broadly the App structure
  - App life cycle management
  - Data Management
  - Defines security model and access mechanism
  - Last User context handling

GENIVI®

Application FW

- GENIVI Reference Architecture works with two different types of Applications
  - Managed Apps
  - Native Applications
- Application can be a UI based or without a UI

GENIVI®

Application FW – Native Applications

- Developed under OEM supervision or otherwise vetted
- “Trusted Application” principle for user data separation
- Security : Follow principle of least possible privilege to reduce damage in case of exploit
- Applications by nature are part of System SW update

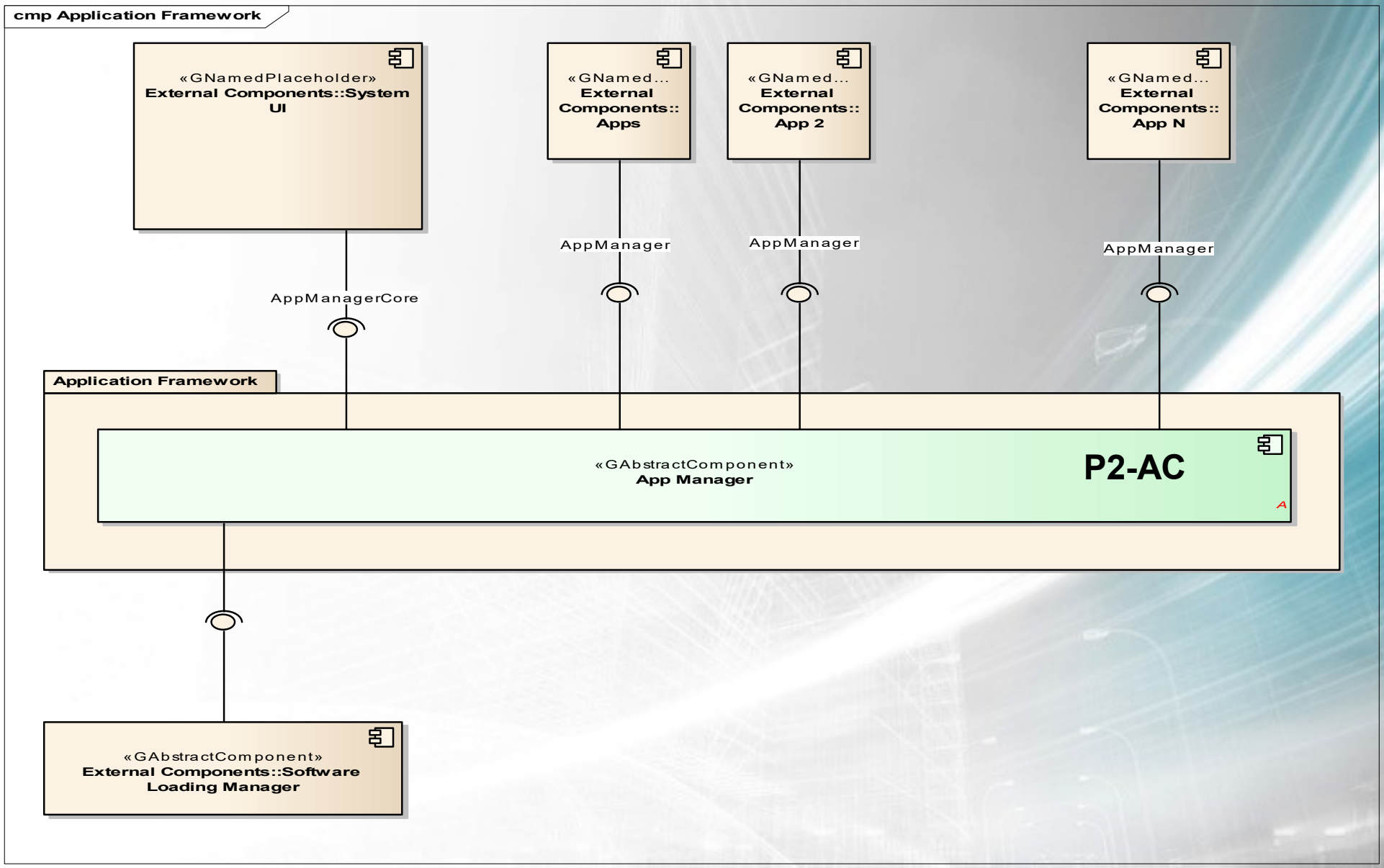
GENIVI®

Application FW – Managed Apps

- Downloadable apps
- Third party developed Apps
- Provide full sandboxing
- May reuse some already existing foreign frameworks
- Extended/alternate lifecycle strategy
- Extended/alternate user management strategy
- Extended/alternate persistence management strategy

GENIVI®

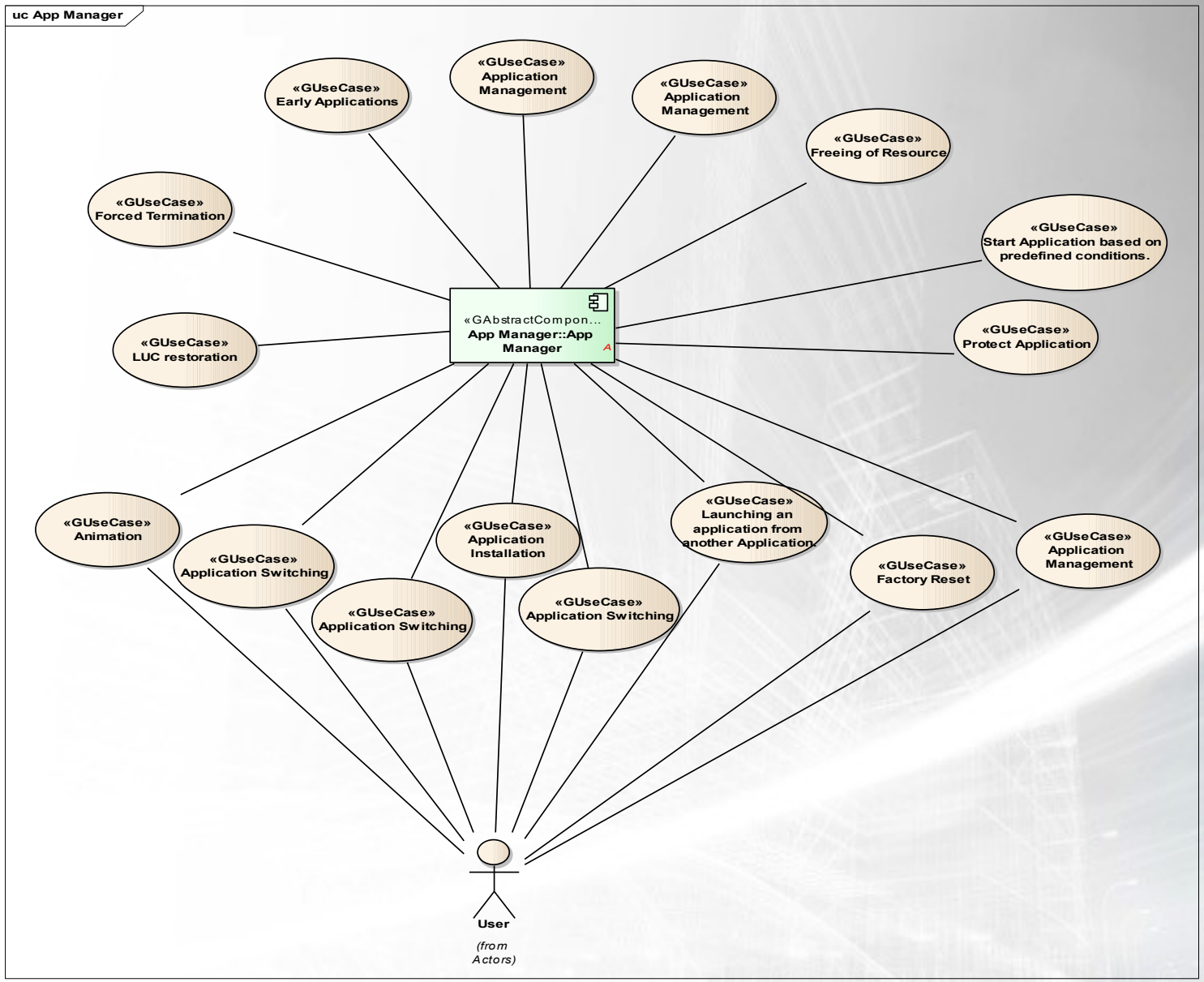
Application FW



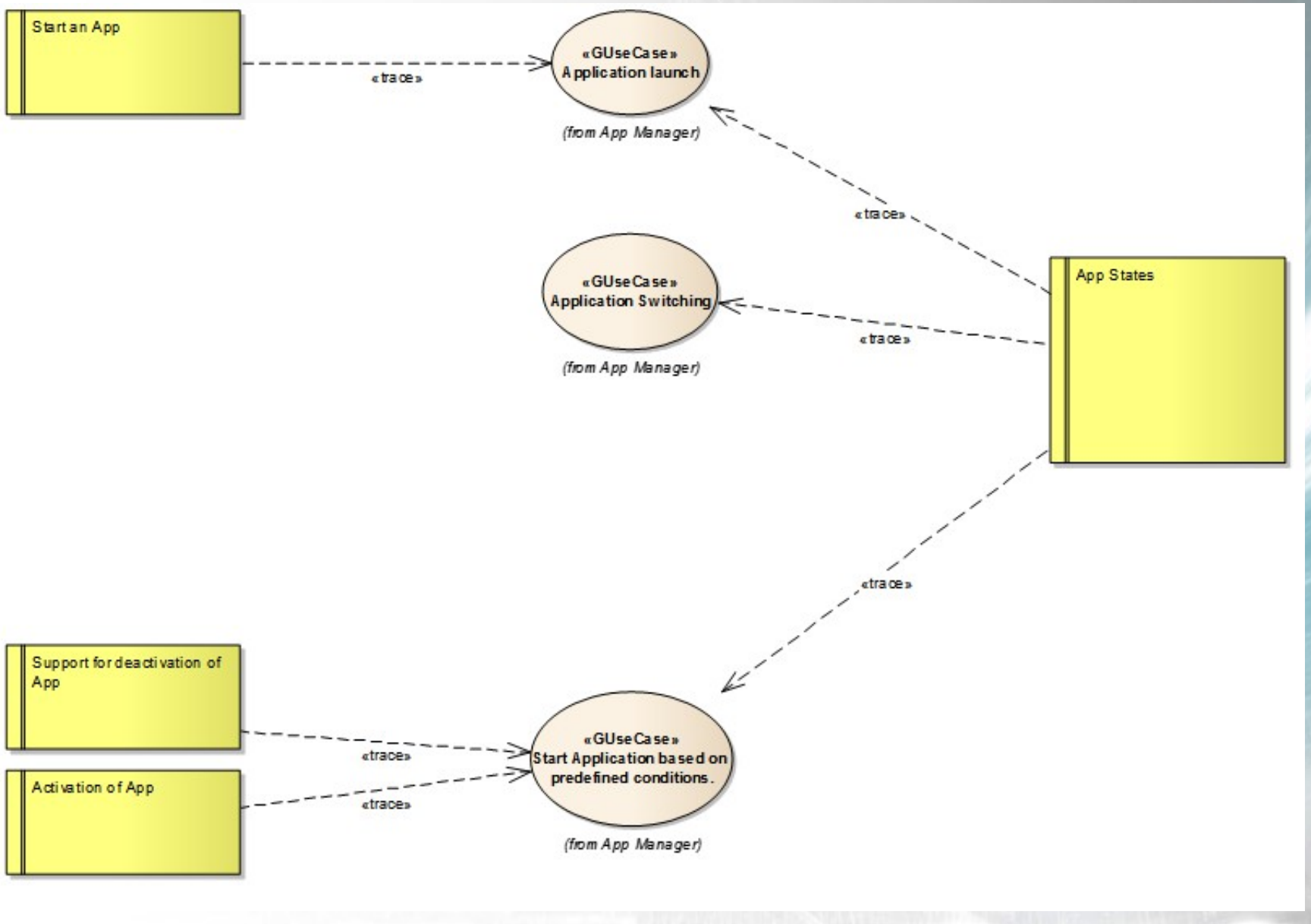
Application FW - App Manager Requirements

GENIVI®

Application FW - App Manager UseCases

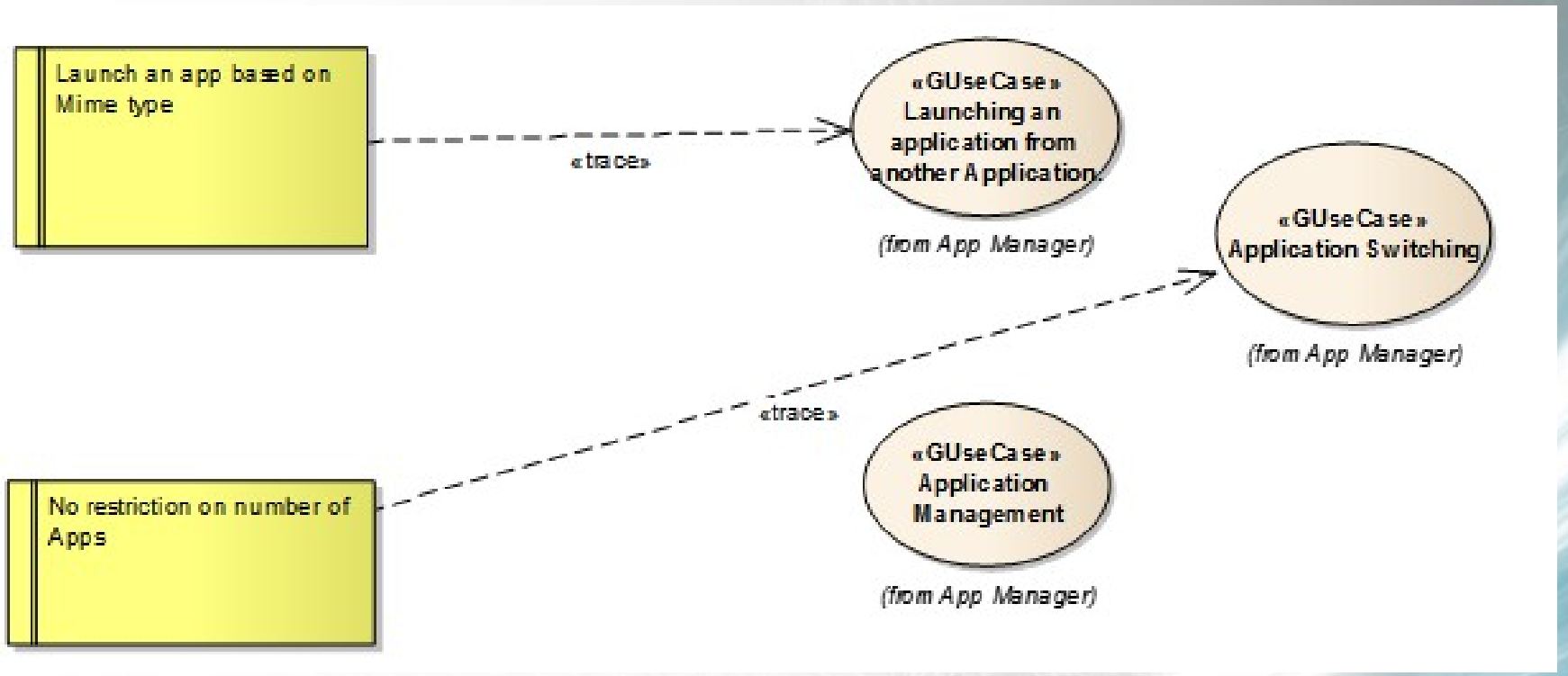


Application FW - App Manager Req – UseCases covered



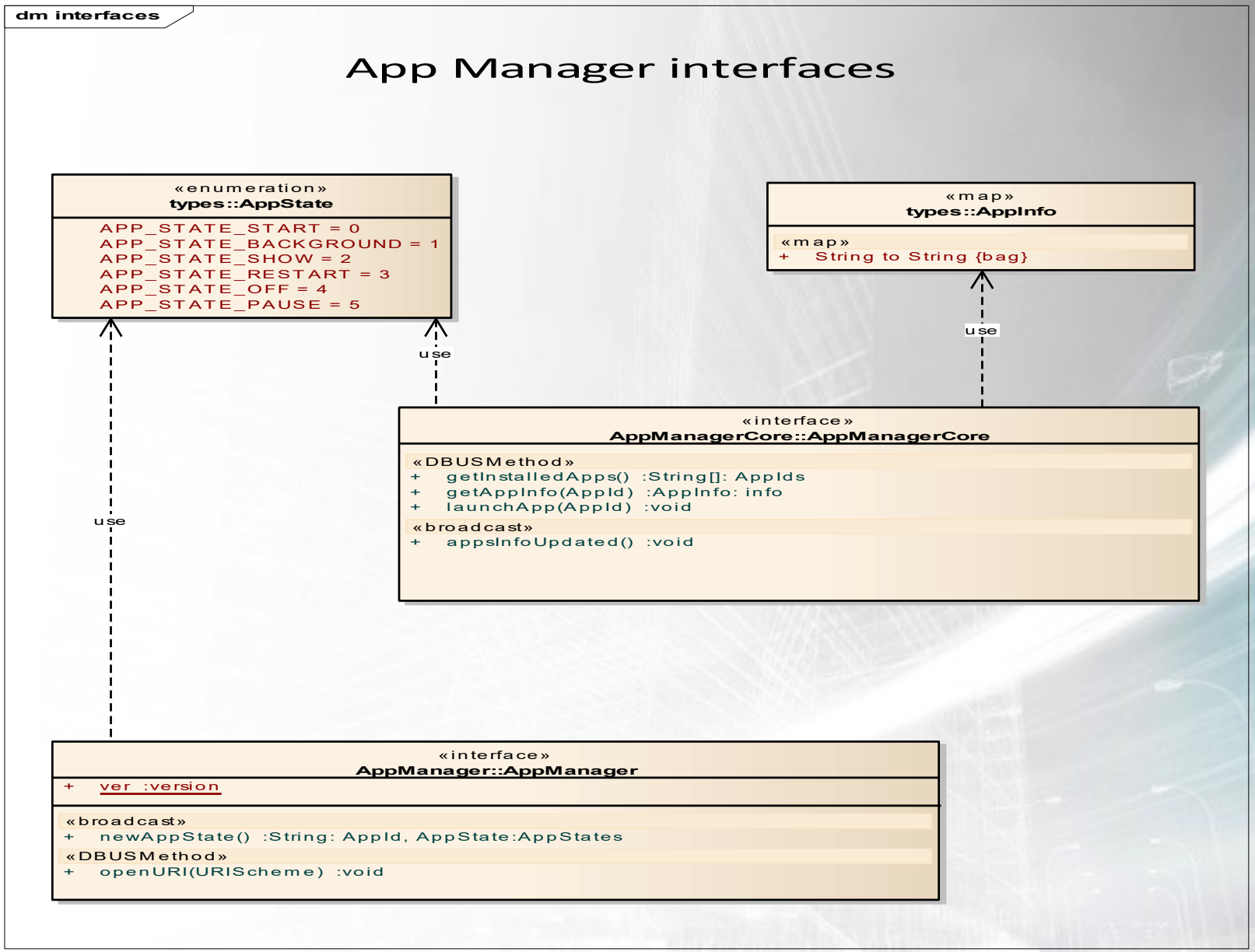
GENIVI®

Application FW - App Manager Req – UseCases covered

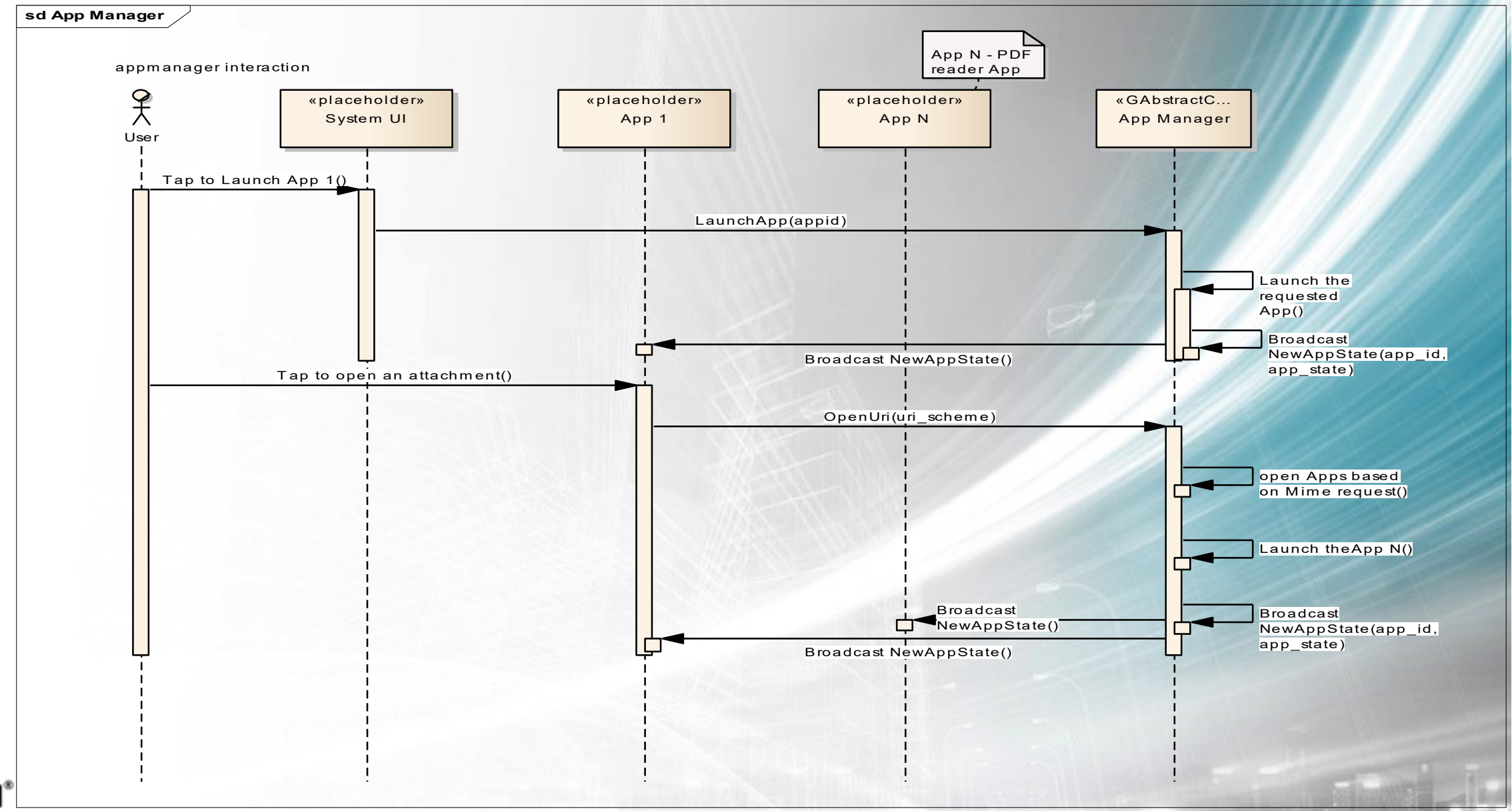


GENIVI®

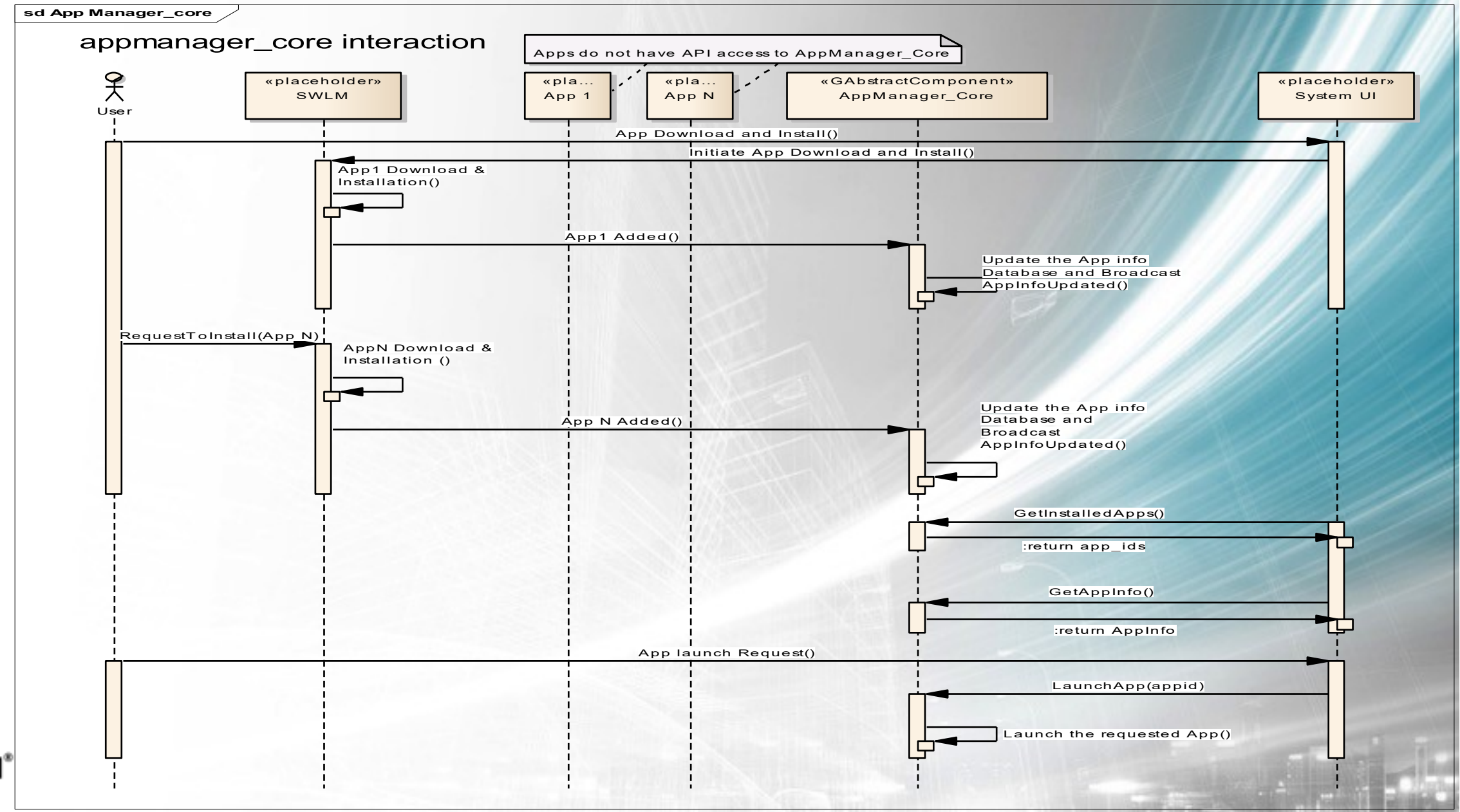
Application FW - App Manager Interfaces



Application FW - App Manager UseCases Realization



Application FW - App Manager UseCases Realization



Application FW – Franca files

https://github.com/GENIVI/app-framework/tree/master/appmana

Application FW - POC

POC – Apertis

Application FW - Working session

Collabora : Application Framework scope and requirements

https://collab.genivi.org/wiki/download/attachments/136937525/Application+Framework+scope+and+requirements

Open Discussion – Requirements and Concepts

GENIVI®

Application FW - Working session

What's in an App?

- There are two commonly-used definitions of an "app": either a user-facing launchable program (an *entry point*) such as would appear in launcher menus, or a user-installable package or bundle such as would appear in an app store.
- It can have one or multiple entry point
- App is expected to bundle with metadata aka metafile
- An App can be a
  - Native code (compiled from C or C++)
  - Run under an interpreter or JIT in a runtime environment (Java, Py, JS etc)
  - HTML5
- App FW is expected to support installation and execution of a/m Apps types

GENIVI®

Application FW - Working session

Data Management

- System / App FW needs to provide a mechanism to handle the App's private data
- FW may need to delete cached data of the Apps to free up memory
- It should be possible for the App FW to delete the App's private data for example as part of removal or a factory reset.

Application FW - Working session

Sandboxing and Security

- App processes should run in a sandbox which partially isolates them from the rest of the system.
- Any data with this access model is considered to be private data, whether it is in files directly written by the app, files written by platform libraries used by the app, or other data stored on behalf of the app by platform services (for example accessed via interprocess communication)
- Private data availability, confidentiality and integrity
- Per-user data would most commonly be kept outside apps' sandboxes and accessed via inter-process communication to a shared service
- User data availability, Confidentiality and integrity

GENIVI®

Application FW - Working session

Sandboxing and Security

- *App integrity*: a malicious or compromised app app must not be able to modify the executables and static data of other apps.
- *App confidentiality*: in general, a malicious or compromised app must not be able to list the other apps that are running on the system or the other apps that are installed, either by their bundle names, by their entry points
- *System integrity*: a malicious or compromised app app must not be able to violate the integrity of the system as a whole (for example by modifying the executables or static data of the system, or by altering the system's idea of what is a trusted app source
- *Resource limits*: A malicious, compromised or buggy app might use more than its fair share of system resources, including CPU cycles, RAM, storage (flash) or network bandwidth

GENIVI®

Application FW - Working session

App Permissions

Use cases

- Operations that cost money might be considered to be particularly sensitive e.g. a parent installing apps on behalf of a child is likely to want to prevent them. It is important that operations like "send SMS" and "make in-app purchases" must be declared in advance.

- Access to online accounts (such as social media) might be considered particularly susceptible to social engineering (since a user might not recognize when a request to fill in their social media account/password is or isn't legitimate), It is important that operations that operations involving these accounts must be declared in advance.

GENIVI®

Application FW - Working session

App Permissions

- App bundles must be able to specify permissions without which they will not work, given in bundle metadata

- The user might be asked whether to grant those permissions on installing that app bundle or on launching any entry point from that bundle, or the framework might automatically grant certain permissions based on approval from an app-store curator without user interaction

GENIVI®

Application FW - Working session

App Launching

- A launcher must be able to list all of the visible, available entry points in any installed bundle, together with enough metadata to display them in its menus
- The base set of metadata fields should be standardized, in the sense that they are described in a vendor-neutral document
- vendors will wish to introduce non-standardized metadata, either as a prototype for future standardization or to support vendor-specific additional requirements.

GENIVI®

Application FW - Working session

Document Launching

- An entry point must be able to identify the file types that it can receive. For example, a document viewer might register itself to receive Microsoft Word documents, Open Document Text files, and PDFs.
- If no handler is available for the selected file type, the app framework should arrange for a suitable fallback to be displayed. For example, it might show an error message, or it might launch its app store user interface with a search query for the handlers for that file type
- app-bundles should be able to define their own new file types. This can potentially alter the interpretation of existing file types

GENIVI®

Application FW - Working session

URI Launching

- Some app entry points will provide handlers for particular URI schemes such as https, mailto or skype
- When a URI is activated, the app framework must locate the entry points that are able to handle that URI and choose one for launching, much like file type handling.
- As with URI schemes, system vendors must be able to force a particular app to handle particular URIs. For example, a vendor might wish to make their built-in web browser handle all http and https URIs

GENIVI®

Application FW - Working session

Content Selection

- App might wish to interact with data stored in locations that are not naturally accessible to the app. e.g, an attachment to an email would be private data for the email app as run by the user whose email account is accessing it.
- The app framework should provide a way to ask the user to browse for a file to open for reading, similar in principle to the conventional "Open" dialog on desktop operating systems
  - If the user does so, the framework must make this file available to the app program for reading.
  - If the user cancels this prompt, the framework must indicate this to the requesting app, and must not grant it any additional access to any files.

GENIVI®

Application FW - Working session

Data Sharing

- One use-case for this is that a global search facility within the platform needs to discover a list of background services (entry points) within app bundles that can provide search results in response to user queries entered into some global search UI; for example, a Spotify client could use the search term to match artists or songs.

- Another use-case for data sharing is a menu for sharing content with other users, for example via social media, email or real-time communications, similar to the Android Sharing menu.

GENIVI®

Application FW - Working session

App Life cycle Management

- The possible states of a program in an app are as follows:
  - Not installed
  - Inactive (installed but not running)
  - Running
  - Paused

- The valid state transitions move linearly through that list in single steps:
  - Not installed → inactive: install app bundle
  - Inactive → running: start (launch), see this section
  - Running → paused: pause, see this section
  - Paused → running: unpause, see this section
  - Running → inactive: stop (kill, terminate), see this section
  - Inactive → not installed: remove app bundle

Application FW - Working session

Last Used Context

- The system must allow each app to store a *last-used context* that encodes its user-visible state during its most recent use.
- If an app does not have any particular state, a reasonable fallback implementation is that its last-used context is the same as normal app launching
- As noted in Life-cycle management, the app program should be given the opportunity to save its last-used context before it is paused or stopped.
- The app program may save its last-used context in response, but is not required to do so.
- The app framework should give the app an indication of whether it is expected to load its last-used context or not

GENIVI®

Application FW - Working session

App Download and Installation management

- App bundles can be downloaded and installed from App store or storage media
- It must be possible to have multiple downloads in parallel
- The system may have a limit on the maximum number of downloads that will proceed in parallel. If it does, additional downloads must be held in a queue and downloaded subsequently
- The progress of each pending download must be updated regularly.
- Upgrading Apps should be possible and checked periodically
- Should rollback mechanism be provided?
- The user must be able to remove an installed app.

Application FW - Working session

Thank you

Thank you!

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org

Contact us: help@genivi.org

GENIVI®