Execution Management

Execution Management is the functional cluster within the Adaptive Platform Foundation that is responsible for platform initialization and the startup and shutdown of Adaptive Applications.

Execution Management, in common with other Applications is assumed to be a process executed on a *POSIX* compliant operating system.

Execution Management Responsibilities

Execution Management is responsible for:

- Initiating execution of the processes in all the Functional Clusters Adaptive AUTOSAR Services,
- Initiating execution of the processes in user-level Applications.
- The launching order to ensure proper startup of the AUTOSAR Adaptive Platform.
- Initialization / configuration of the OS to enable it to perform the necessary run-time scheduling and resource management

Execution Management is **not** responsible for run-time scheduling of Processes

Manifests

Manifests can be considered as configuration files [Written in ARXML] that are determined during design and can be transformed to platform specific format during development or deployment or during runtime [by UCM].

Types of manifests (deals with EM):

- Machine Manifest: The contents of a Machine Manifest include the configuration of Machine properties and features (resources, safety, security, etc.)
- Execution Manifest: Process properties (startup parameters, resource group assignment, scheduling priorities etc.).

Applications

Applications are developed to resolve a set of coherent functional requirements. An Application consists of executable software units, additional execution related items (e.g. data or parameter files). An Application can be implemented in one or more Executables.

Each executable has its own Manifest file and its arguments and configuration.

Process is considered as a running instance of an executable , mainly POSIX deals with processes during scheduling or resource allocation etc..

Types of Applications:

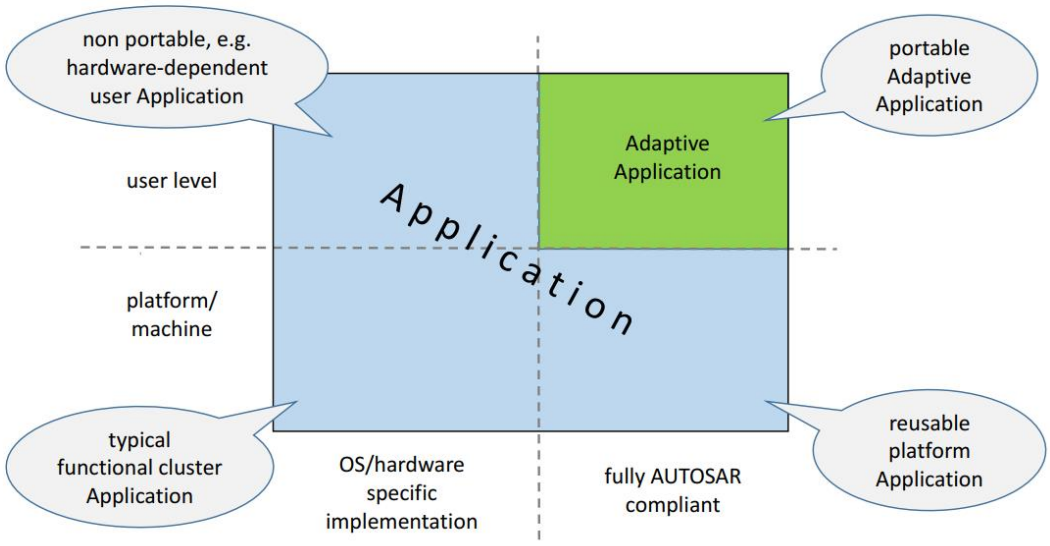

Figure 7.1: Types of Applications

Although we have different types of processes, but the EM treats both USER LEVEL or MACHINE with the same APIs as the POSIX deals with processes not applications.

Adaptive Application

An Adaptive Application is a specific type of Application. The implementation of an Adaptive Application fully complies with the AUTOSAR specification, i.e. it is restricted to the use of APIs standardized by AUTOSAR and needs to follow specific coding guidelines to allow reallocation between different implementations of the AUTOSAR Adaptive Platform.
Adaptive Applications are always located above the middleware. To allow portability and reuse, user level Applications should be Adaptive Applications whenever technically possible.

Executables

An Executable is a software unit which is part of an Application. It has exactly one entry point (main function) [SWS_OSI_01300]. An Application can be implemented in one or more Executables.

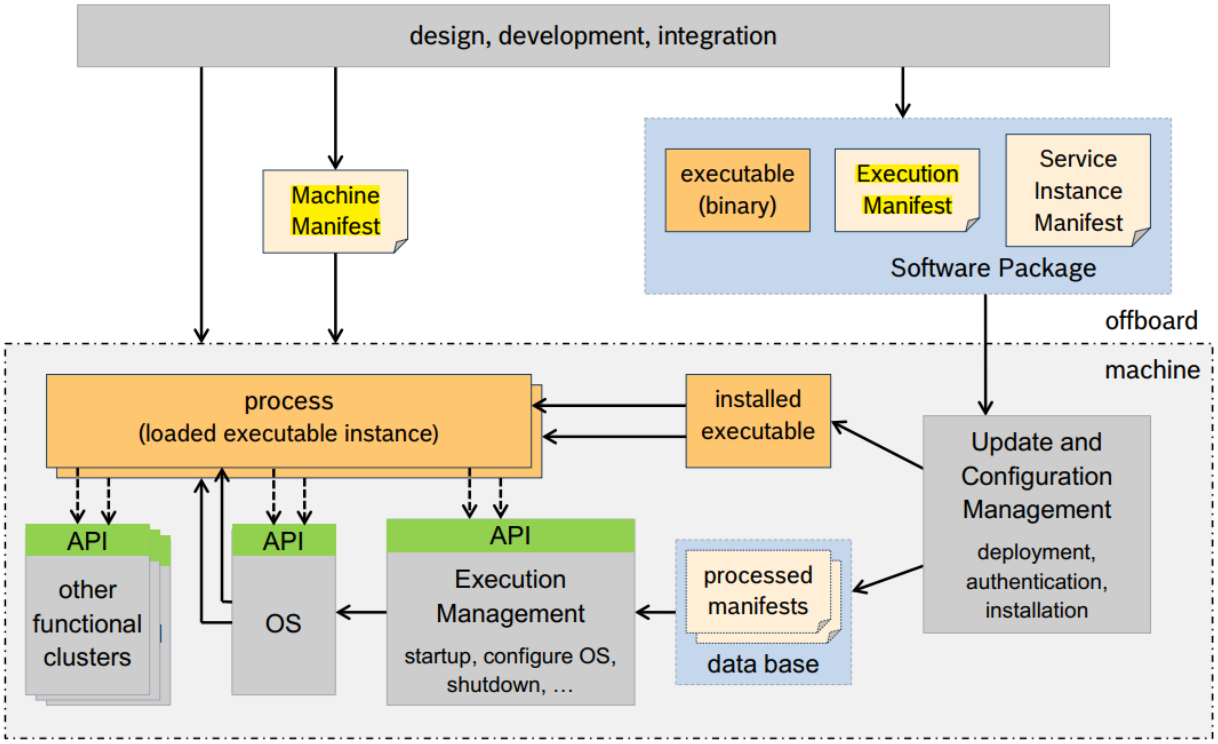

Figure 7.2: Executable Lifecycle from deployment to execution

Brief on Machine, Process , Execution States

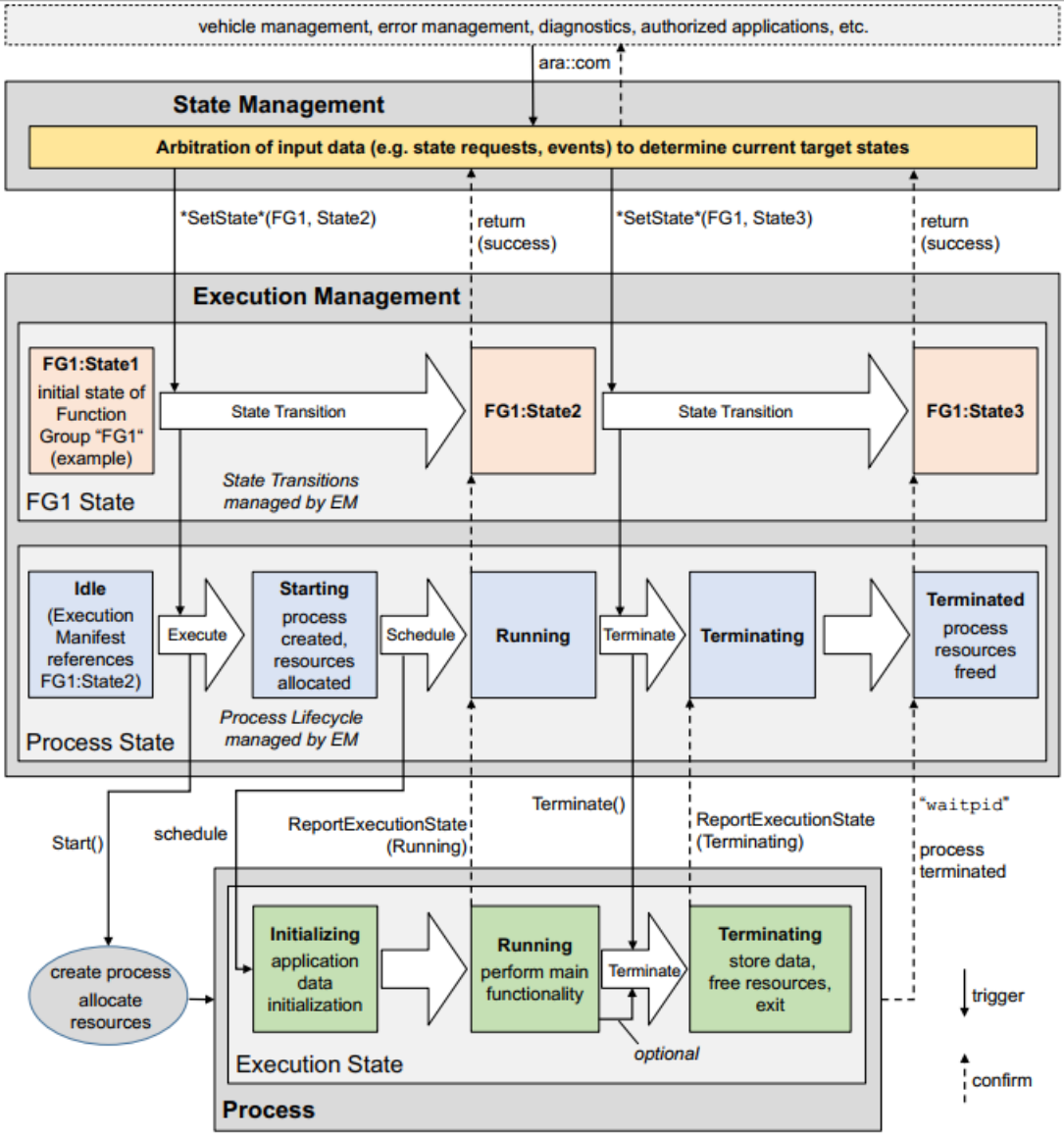

Figure 7.9: Interaction between states

Application Recovery Actions

The Platform Health Management Monitors Processes and could trigger a Recovery Action in case any Process behaves not within the specified parameters.
The Recovery Actions are defined by the integrator based on the software architecture requirements for the Platform Health Management and configured in the Execution Manifest.

The EM can any time request from the PHM the process states information.

The PHM can request several RECOVERY actions from the EM:

- Process Restart: A notification must show the state of restarting [Process Restart Failed, Process Restart Successful]
- Process Enter Safe State

Deterministic Execution

In general process are deterministic or self-terminating.

Types of Deterministic Processes:

- Time Determinism: The output of the calculation is always produced before a given deadline (a point in time).
- Data Determinism: Given the same input and internal state, the calculation always produces the same output.
- Full Determinism: Combination of Time and Data Determinism as defined above.

Resource Limitation

Resource Configuration

The Resources that are controlled by this module: RAM, CPU Time

In general, we need to distinguish between two resource demand values:
• Minimum resources, which need to be guaranteed so the process can reach its Running state and perform its basic functionality.
• Maximum resources, which might be temporarily needed and shall not be exceeded at any time, otherwise an error can be assumed.

The Application developer is responsible for the determination of memory usage so developer must configure the Manifest of the executable.

Resource Monitoring

For the entire system, the monitoring part of this activity is fulfilled by the Operating System. For details on CPU time monitoring.
The monitoring capabilities depend on the used Operating System. Depending on system requirements and safety goals, an appropriate Operating System has to be chosen and configured accordingly, in combination with other monitoring mechanisms (e.g. for execution deadlines) which are provided by Platform Health Management.

Fault Tolerance

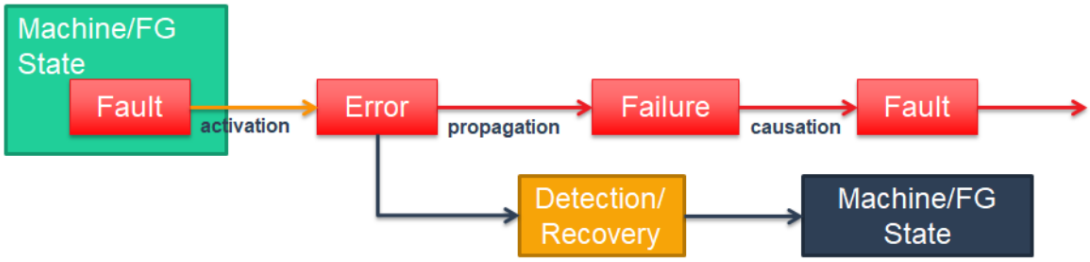

Figure 7.20: General Fault Tolerance scheme.