MICROEJ MODULE MANAGER
(MMM)
SPECIFICATION

Version 2.0-C

DISCLAIMER & CONFIDENTIALITY

MICROEJ®

MMM, MicroEJ Module Manager

- Modern electronic device design involves many parts and teams to collaborate to finally obtain a product to be sold on its market.

- MicroEJ encourages modular design which involves various stake holders: hardware engineers, UX designers, graphic designers, drivers/BSP engineers, software engineers, etc.

- Modular design is a design technique that emphasizes separating the functionality of an application into independent, interchangeable modules. Each module contains everything necessary to execute only one aspect of the desired functionality.

- In order to have team members collaborate internally within their team and with other teams, MicroEJ SDK provides a powerful modular design concept, with smart module dependencies, controlled by the MMM.

- The MMM frees engineers from the difficult task of computing module dependencies. Engineers specify the bare minimum description of the module requirements.

MICROEJ®
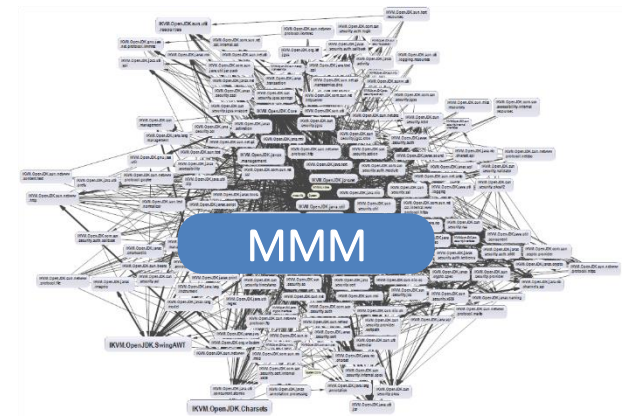
MODULE REPOSITORY

Modules are shared among developers using a Repository

- A repository can be

  - a filesystem organized using directories

  - a database on a server

  - a composite of multiple repositories

Team
Repository 1

Team
Repository 2

Team
Repository 3

MICROEJ
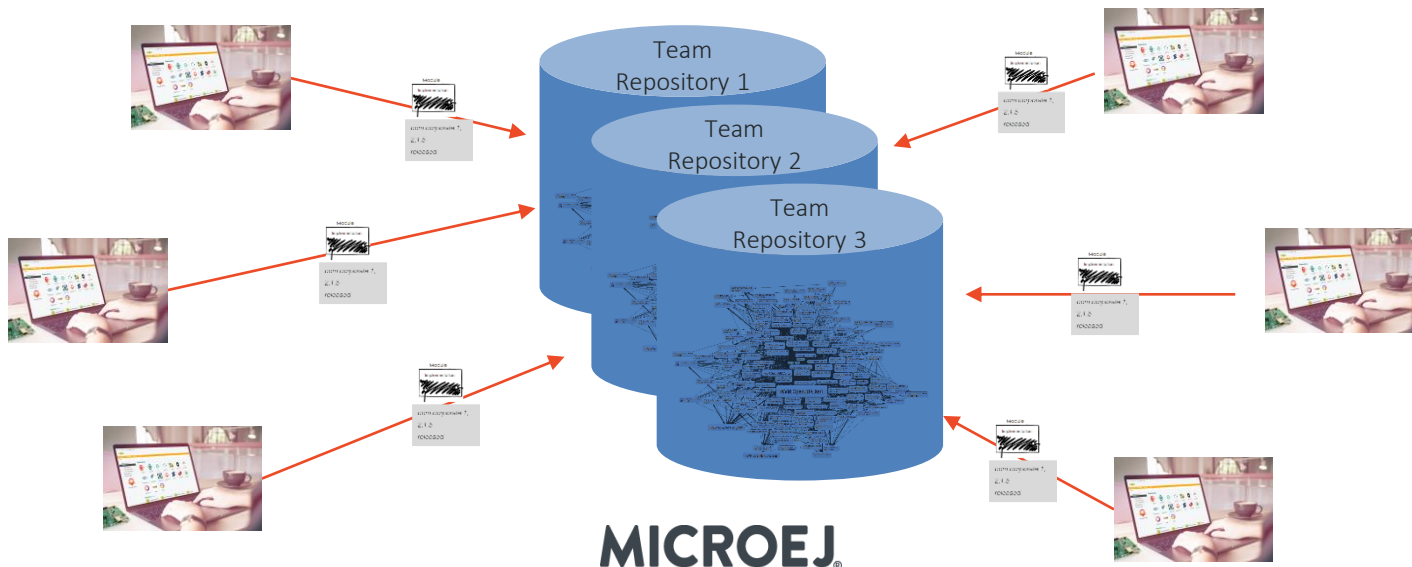
MODULE DEFINITION

- A module is a set of files that contains some information (code, text files, binary files) to be stored in a repository and to be delivered to a work process when requested.

- A module is uniquely identified by:

  - An organization name (a sequence of dot-separated strings)
    - `[a-zA-Z0-9\-_]+ ( \. [a-zA-Z0-9\-_]+ )*`
  - A name (a string with no dot)
    - `[a-zA-Z0-9\-_]+`
  - A version (3 dot-separated numbers)
    - `M.m.p (\-RCYYYYMMDDHHmm)?`
    - `[0-9]+ \. [0-9]+ \. [0-9]+`

- A module has the following information:

  - A list of dependencies
  - A build status: `release` vs `snapshot`
  - A nature to qualify the content

MICROEJ®

MODULE LIFE CYCLE

- Once delivered, the module is called a *release*

  - It has a unique version with the only 3 numbers (`M.m.p`)

  - A new module can only be delivered by incrementing its version

- While not delivered, the module is called a *snapshot*

  - Its version is suffixed with **–RCxxx** pattern

  - A new module can be delivered by incrementing its **–RCxxx** version suffix

- MMM follows Semantic Versioning v2 - https://semver.org/

  - `M.m.p < M.m.p+1-RCxxx < M.m.p+1-RCxxx+1 < M.m.p+1`

timeline

Module
Implementation

com.corp.module
2.1.**4**
release

Module
Implementation

com.corp.module
2.1.**5-**
RC201803141216
snapshot

. . .

. . .

Module
Implementation

com.corp.module
2.1.**5-RC201805060850**
snapshot

. . .

. . .

Module
Implementation

com.corp.module
2.1.**5**
release

MICROEJ
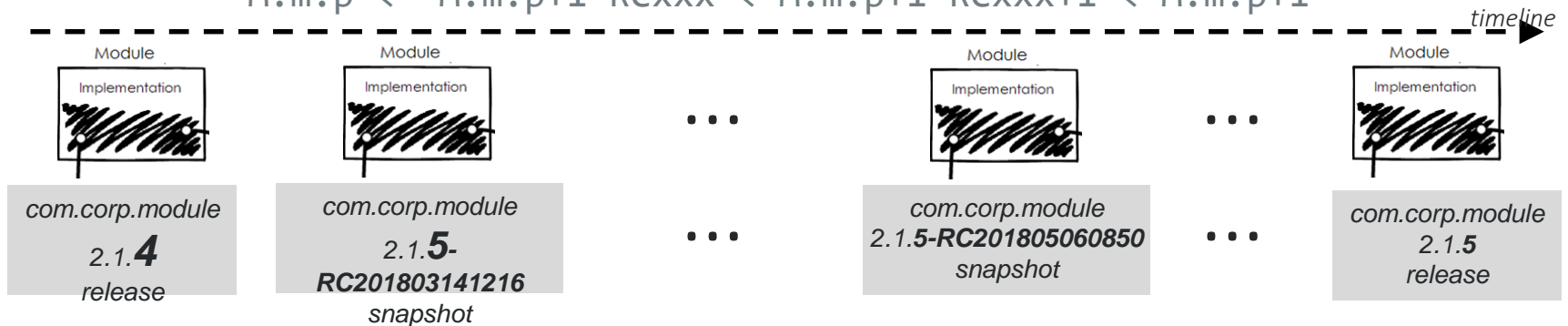
MODULE DEPENDENCIES (1/3)

- A module dependency is a link from one module to another module

- A module declares an ordered list of zero or more dependencies

- The dependencies graph is a direct acyclic graph based on the module name and organisation but not module version number
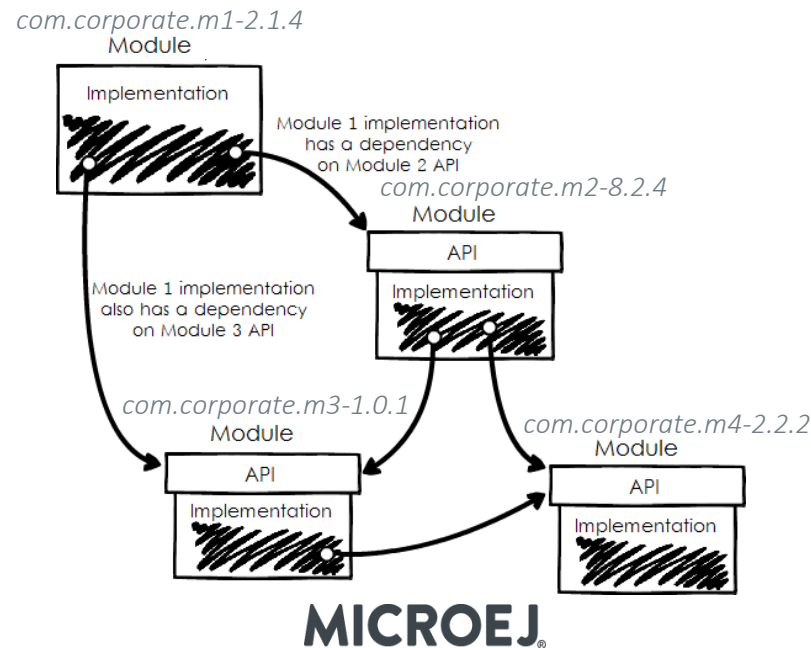
MODULE DEPENDENCIES (2/3)

- A dependency is described by

    - An organization: same format as module organization

    - A name: same format as module name

    - A version: `major.minor.patch` (same format as module version)

    - A matching rule: one of `compatible` (default), `equivalent`, `greaterOrEqual`, `perfect`

    - A visibility: one of `public` (default) or `private`

MICROEJ

MODULE DEPENDENCIES (3/3)

- Dependency Visibility

    - A dependency declared `public` is transitively resolved by upper modules

    - A dependency declared `private` is only used by the module itself, typically for:

        - Bundling the content into the module

        - Testing the module

- Dependency Matching Rule

    - The matching rule indicates the possibility for this module to be replaced by a higher version without breaking its specified behavior

    - Matching rules are used for transitive dependencies resolution or for module update action

    - The following table describes the available matching rules:

Name	Range Notation	Semantic
compatible	[M.m.p-RC, (M+1).0.0-RC[	Default if not set. Equal or up to next major version
equivalent	[M.m.p-RC, M.(m+1).0-RC [	Equal or up to next minor version
greaterOrEqual	[M.m.p-RC, ∞[	Equal or greater versions
perfect	[M.m.p-RC, M.m.(p+1)-RC[	Exact match (strong dependency)

MICROEJ®

MMM RESOLUTION SPECIFICATION

- For each dependency,

    - If the version is `M.m.p`, the fetched module is the **release** module, or the **most recent available snapshot** when there is not a released version yet.

    - If the version is `M.m.p-RCYYYYMMDDHHmm`, the fetched module is this **exact** version

    - if the dependency is `public` the resolved module dependencies are resolved recursively

- If multiple versions of a module have been fetched,

    - The final resolved version is the **highest** version, provided:

        - It matches the rule declared for each dependency

        - It is **equivalent** to the direct dependency declared version (if any)

    - Otherwise, the resolution fails.

- Modules are ordered using a depth-first search topological sort algorithm

    - Use of the dependencies list order of each module.

MICROEJ®

MMM RESOLUTION EXAMPLES (1/4)

Dependency Graph	Repository Content	MMM Resolution (Ordered Result)
A |- B 1.0.0	B 1.0.0	B 1.0.0
A |- B 1.0.0	B 1.0.0 B 1.0.1	B 1.0.0
A |- B 1.0.0	B 1.0.0 B 1.1.0	B 1.0.0
A |- B 1.0.0	B 1.0.0-RC201805090841 B 1.0.0-RC201805091055 B 1.1.0	B 1.0.0-RC201805091055
A |- B 1.0.0	B 1.0.0-RC201805090841 B 1.0.0	B 1.0.0
A |- B 1.0.0 |- C 1.0.0	B 1.0.0 C 1.0.0	B 1.0.0 C 1.0.0
A |- B 1.0.0 |- C 1.0.0 |- C 1.1.0	B 1.0.0 C 1.0.0 C 1.1.0	B 1.0.0 C 1.1.0
A |- B 1.0.0 |- C 1.0.1 |- C 1.0.0	B 1.0.0 C 1.0.0 C 1.0.1	B 1.0.0 C 1.0.1

A
|- B 1.0.0
   |- C 1.0.0
|- D 1.0.0
   |- C 1.1.0
	B 1.0.0
	C 1.0.0
	C 1.1.0
	D 1.0.0
		B 1.0.0
		D 1.0.0
		C 1.1.0

Matching Rules
- C = Compatible
- E = Equivalent
- G = GreaterOrEqual
- P = Perfect

MMM Resolution Examples (2/4)

Dependency Graph	Repository Content	MMM Resolution (Ordered Result)
A |- B 1.0.0 (P)	B 1.0.0 B 1.0.1	B 1.0.0
A |- B 1.0.0 (E)	B 1.0.0 B 1.1.0	B 1.0.0
A |- B 1.0.0 (G)	B 1.0.0 B 2.0.0	B 1.0.0
A |- B 1.0.0 |- C 1.0.0 (G) |- C 2.0.0	B 1.0.0 C 1.0.0 C 2.0.0	B 1.0.0 C 2.0.0

A
|- B 1.0.0
    |- C 1.0.0 (G)
|- D 1.0.0
    |- C 2.0.0
	B 1.0.0
	C 1.0.0
	C 2.0.0
	D 1.0.0
		B 1.0.0
		D 1.0.0
		C 2.0.0

Matching Rules
- C = Compatible
- E = Equivalent
- G = GreaterOrEqual
- P = Perfect

MICROEJ.

MMM Resolution Examples (3/4)

Dependency Graph	Repository Content	MMM Resolution (Ordered Result)
A |- B 1.0.0	∅	ERROR B 1.0.0 not found
A |- B 1.0.0 |- C 1.0.0	B 1.0.0 C 1.1.0	ERROR C 1.0.0 not found
A |- B 1.0.0	B 1.0.1	ERROR B 1.0.0 not found
A |- B 1.0.0	B 1.1.0	ERROR B 1.0.0 not found
A |- B 1.0.0 |- C 1.1.0 |- C 1.0.0	B 1.0.0 C 1.1.0	ERROR C 1.0.0 not found
A |- B 1.0.0 |- C 1.1.0 |- C 1.0.0	B 1.0.0 C 1.0.0 C 1.1.0	ERROR The higher version required by B➜C is **not equivalent** to the version declared by A➜C as a direct dependency)
A |- B 1.0.0 |- C 1.0.0 |- C 2.0.0	B 1.0.0 C 1.0.0 C 2.0.0	ERROR The higher version required by B➜C is **not equivalent** to the version declared by A➜C as a direct dependency)

Matching Rules
- C = Compatible
- E = Equivalent
- G = GreaterOrEqual
- P = Perfect

MICROEJ®

MMM RESOLUTION EXAMPLES (4/4)

Dependency Graph	Repository Content	MMM Resolution (Ordered Result)

A
|- B 1.0.0
    |- C 1.0.0
|- D 1.0.0
    |- C 2.0.0
	B 1.0.0
	C 1.0.0
	C 2.0.0
	D 1.0.0
		ERROR
		The higher version required by D→C is **not compatible** to the version declared by B→C)

A
|- B 1.0.0
    |- C 1.0.0 (P)
|- D 1.0.0
    |- C 1.0.1
	B 1.0.0
	C 1.0.0
	C 1.0.1
		ERROR
		The higher version required by D→C is **not perfect** to the version declared by B→C)

A
|- B 1.0.0
    |- C 1.0.0 (E)
|- D 1.0.0
    |- C 1.1.0
	B 1.0.0
	C 1.0.0
	C 1.1.0
		ERROR
		The higher version required by D→C is **not equivalent** to the version declared by B→C)

Matching Rules
- C = Compatible
- E = Equivalent
- G = GreaterOrEqual
- P = Perfect

MICROEJ

UPDATE Action

- The module dependencies list is modified as following:

    - For each dependency, search in the repository for the latest available version still valid for the matching rule and update to this version

    (Dependencies marked with matching rule `perfect` are never updated)

- UPDATE can be applied

    - Manually, from the IDE or command line (ANT Task)

    - Automatically, before resolution, each time a module is built

MICROEJ®

UPDATE ACTION EXAMPLES

Dependency Graph (Before)	Repository Content	Dependency Graph (After)
A |- B 1.0.0	B 1.0.0	A |- B 1.0.0
A |- B 1.0.0	B 1.0.0 B 1.1.0	A |- B 1.1.0
A |- B 1.0.0	B 1.0.0 B 1.1.0 B 2.0.0	A |- B 1.1.0
A |- B 1.0.0	B 1.0.0 B 1.1.0-RC201805090841	A |- B 1.1.0
A |- B 1.0.0 (G)	B 1.0.0 B 1.1.0 B 2.0.0	A |- B 2.0.0
A |- B 1.0.0 (E)	B 1.0.0 B 1.0.1 B 1.1.0 B 2.0.0	A |- B 1.0.1
A |- B 1.0.0 (P)	B 1.0.0 B 1.0.1 B 1.1.0 B 2.0.0	A |- B 1.0.0

MICROEJ®

MMM Ivy / Easyant Mapping

- Enable MicroEJ MMM specification

    - Add `ej:version` on Ivy module header

    - `<ivy-module version="2.0" xmlns:ej="https://developer.microej.com" ej:version="2.0.0" >`

- Specify a dependency matching rule

    - Add `ej:match` on dependency line

    - `<dependency org="xxx" name="xxx" rev="3.0.0" ej:match="perfect"/>`

- Enable Automatic Update before resolution

    - Add the following Easyant plugin dependency

    - `<ea:plugin org="com.is2t.easyant.plugins" module="ivy-update" revision="1.+" />`

COMPATIBILITY MODE

- MMM is compatible with any legacy built MicroEJ modules.

    - When a legacy module is resolved from a MMM module, each of its dependencies are automatically converted as following:

        - The version is the version that was used when the module was built (Ivy **rev** field)

        - The matching rule is based on the declared revision range (Ivy **revConstraint** field), with the following conversions:

Legacy Module revConstraint	MMM Matching Rule
`[M.m.p, M+1.u.v[`	`compatible`
`[M.m.p, M.m+1.u[`	`equivalent`
`[M.m.p, M.m.p+1[`	`perfect`
`[M.m.p,)`	`greater`
`+`	`greater`
`M.+`	`compatible`
`M.m.+`	`equivalent`
Any other unrecognized pattern	`compatible`

MICROEJ®

INTERNAL UNDERSTANDING NOTES

- Whatever the target repository content, release dependencies are always resolved with the same versions (snapshots are automatically updated to the most recent version)

- Uploading modules to a server does not change resolution result (for release modules), but update action result

- Matching Rules specification is Eclipse Feature matching rule

  - https://help.eclipse.org/photon/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fmisc%2Ffeature_manifest.html

MICROEJ

WHY

- Simplify usage

- Deterministic result
  - Ivy ranges notation was used for 3 different reasons:
    - Allow to fetch snasphots modules while it is not yet released
    - Allow the fetch of multiple versions before conflict resolution (otherwise `latest-compatible` resolution triggers an error)
    - Get the latest available version in the repository

- Semantic designed for
  - cache only resolution (offline)
  - repositories connection with high request latency (VPN: ~60ms per request)

- Extract semantic out of tools
  - Prepare a GUI view
  - Prepare to use other dependency managers than Ivy in the future (or other IDEs)
  - Documentation

MICROEJ

CHANGELOG

- **2.0-C** - 2019-06-27

  - Fixed regular expression (escaped '**.**' and '-' reserved characters)

- **2.0-B** - 2019-03-15

  - Added a Changelog

  - Fixed missing '-' character in organization string pattern

- **2.0-A** - 2018-10-02

  - Initial Revision

MICROEJ®