TELESPAZIO VEGA
DEUTSCHLAND
A Finmeccanica / Thales Company

UMF – A Productive SMP2 Modelling and Development Tool Chain

P. Ellsiepen, P. Fritzen (Telespazio VEGA)

V, Reggestad, T. Walsh (ESA/ESOC)

SESP 2012, ESTEC, Nordwijk
25-27 September 2012

OUTLINE

- Introduction
  - SMP2 and E-TM-40-07 SMP
  - Simulator Software Development Environment
- UMF Concepts and Features
  - Efficiency and Productivity
  - Dependency Management and Deployment
- Case Study: The BepiColombo Simulator
- Summary

Introduction

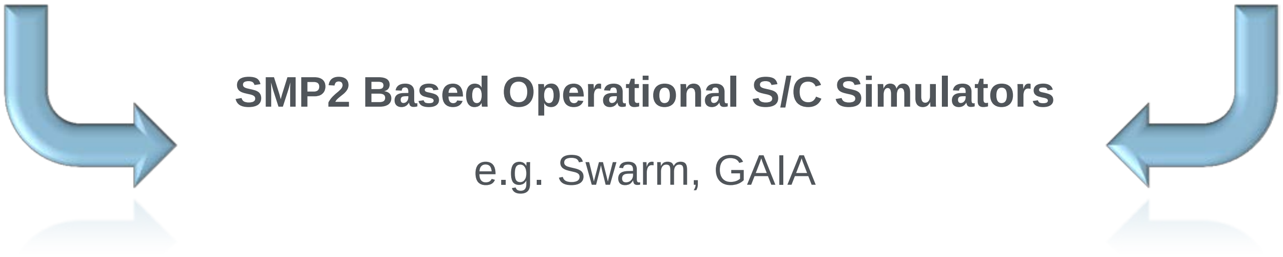
SMP2 and
E-TM-40-07 SMP

SMP STANDARDISATION HISTORY

- SMP = Simulation Modelling Platform (was: Simulation Model Portability)
- 1999: SMP1 / SMI
  - Focus on portability only (operating system, simulation environments)
  - C API
- 2005: SMP2 1.2
  - Focus on model development & integration, inter-model communication
  - C++ API, other languages possible (e.g. Java), XML meta-data (SMDL)
  - Various simulation environments (e.g. SIMSAT, Basiles, EuroSim, …)
- 2008: E-TM-40-07 SMP Draft C
  - “SMP2 plus lessons learned“: improvements from practical experience
  - No implementations yet

SMP2 MODEL DEVELOPMENT @ ESOC

- SMP2 Model Development Tools
  - 2006: SIMSAT 4 MIE
  - 2008: EGOS-MF v1
  - 2010: UMF v1
  - 2010: SMP-CS (ESTEC)
- SMP2 Model Libraries & Patterns
  - 2006: Generic Models (GENM)
  - 2008: Spacecraft Simulator Reference Architecture (REFA)



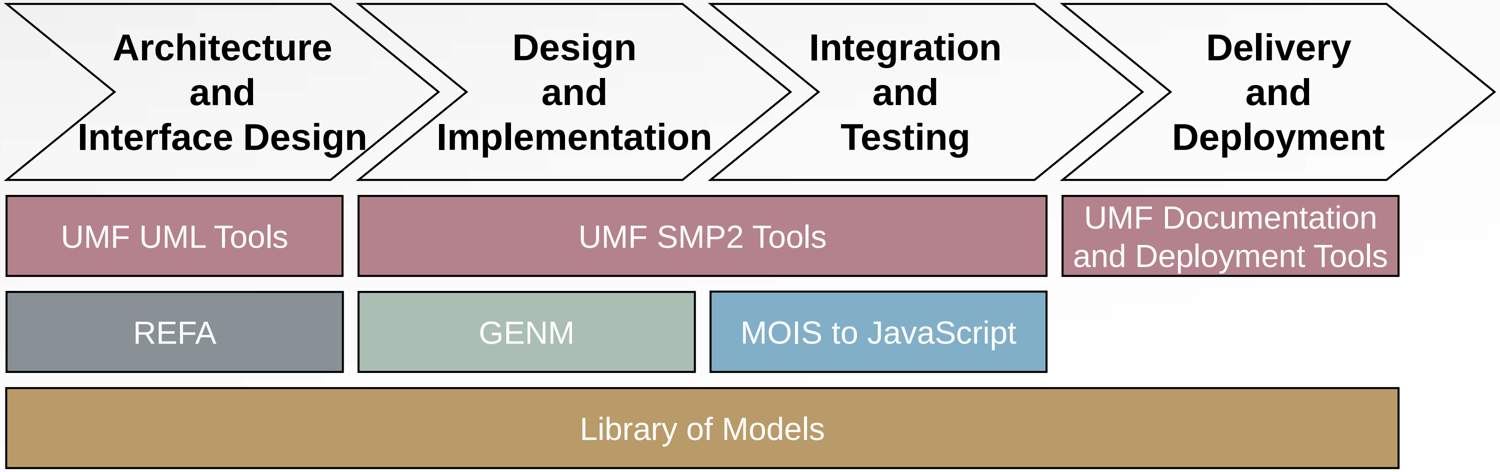
SMP2 Based Operational S/C Simulators

e.g. Swarm, GAIA

- **What's missing? Next steps?**
  - A **consolidated** development environment
    - with focus on **usability** and developer **productivity**
      - and management of **dependencies**, **deployment** and **reuse**

Introduction

SimSDE – Simulator Software Development Environment



Architecture and Interface Design

Design and Implementation

Integration and Testing

Delivery and Deployment

UMF UML Tools

UMF SMP2 Tools

UMF Documentation and Deployment Tools

REFA

GENM

MOIS to JavaScript

Library of Models

SIMSDE OBJECTIVES AND PRODUCTS

- Provide a validated and productive Modelling Framework for the needs of Simulus and the Operational Simulators
  - **Universal Modelling Framework (UMF), v2**

UMF

- Provide a definition and a first implementation of a Library of Models concept
  - **Library of Models (LoM), v1**

LoM

- Evolve SMP2 based Simulus models (Generic Models, REFA, FDS-DIF)
  - **Generic Models (GENM), version 5**
  - **Spacecraft Simulator Reference Architecture (REFA), v2**

GENM

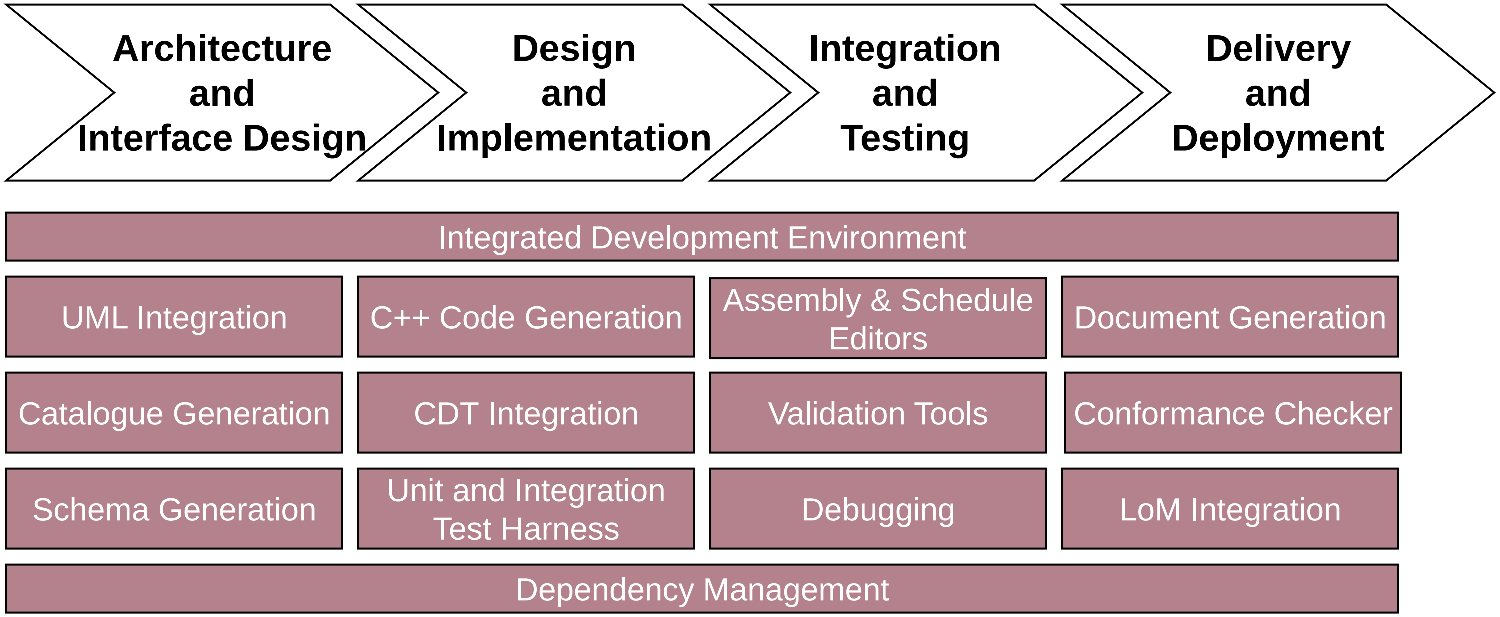
REFA

- Provide support tools for simulator testing
  - **MOIS to JavaScript Converter (M2J), v1**

M2J

UNIVERSAL MODELLING FRAMEWORK

- UMF v2 main features
  - Independent of SMP2 simulation runtime environment
  - “Best of all worlds“ joining parts from EGOS-MF, UMF v1, and SMP-CS
  - Eclipse based IDE for SMP2 based simulator developments
  - Command-line tools for various usage scenarios and batch processing



Architecture and Interface Design → **Design and Implementation** → **Integration and Testing** → **Delivery and Deployment**

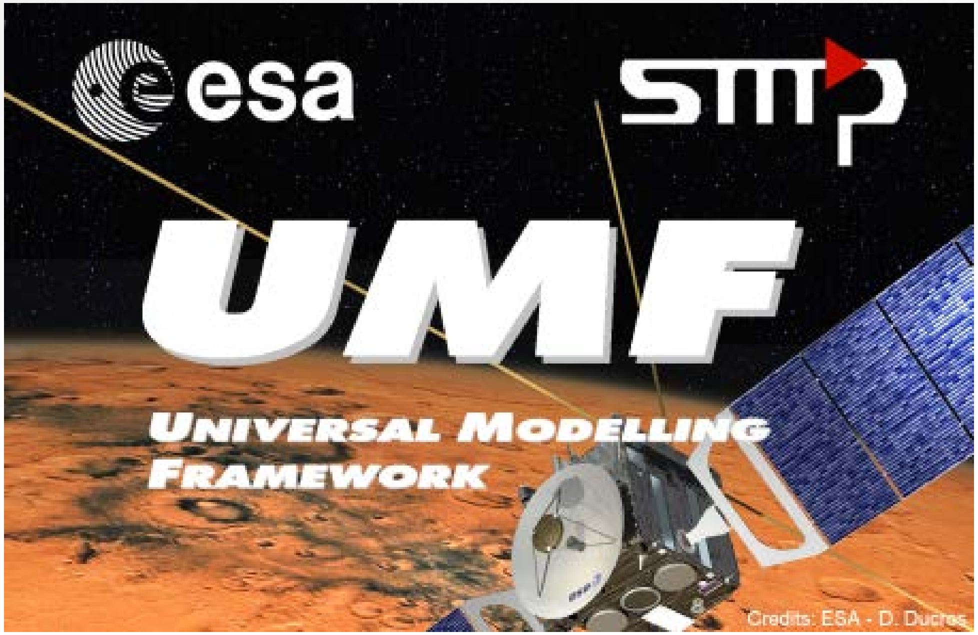
Integrated Development Environment

UML Integration	C++ Code Generation	Assembly & Schedule Editors	Document Generation
Catalogue Generation	CDT Integration	Validation Tools	Conformance Checker
Schema Generation	Unit and Integration Test Harness	Debugging	LoM Integration

Dependency Management

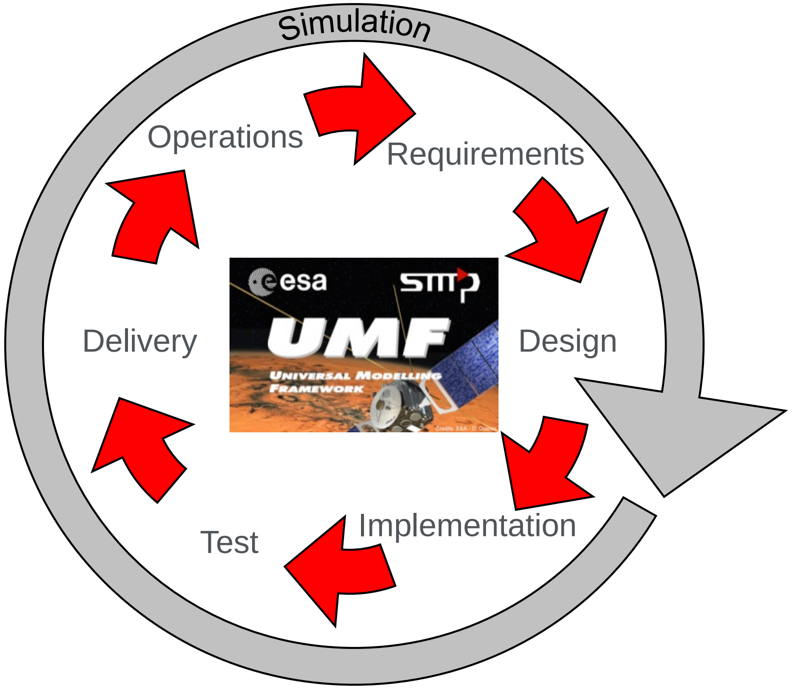
UMF Concepts and Features

Efficiency and Productivity



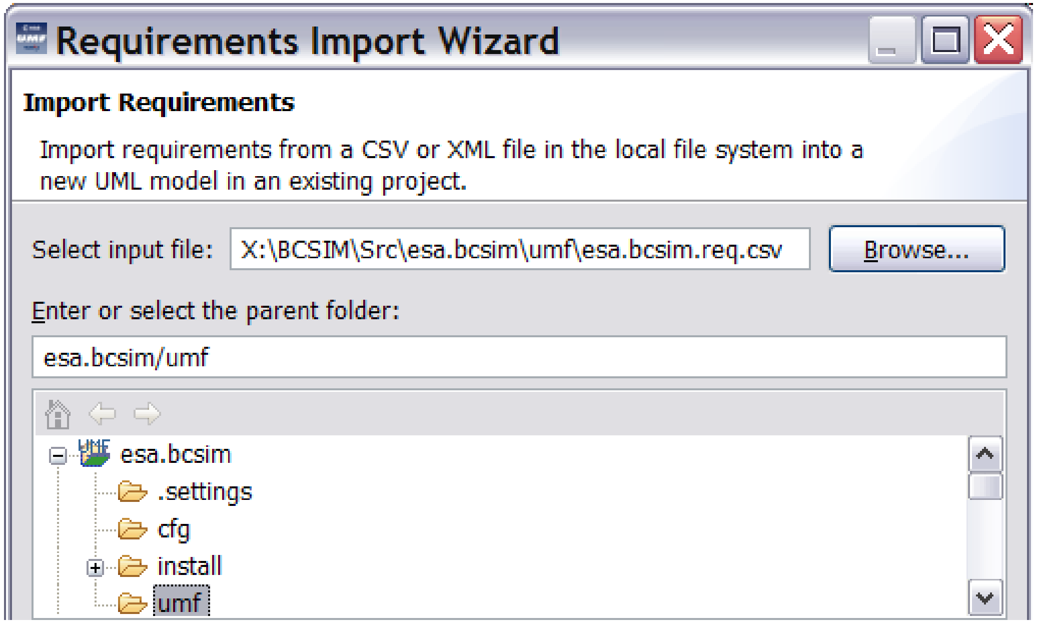
SIMULATION LIFECYCLE

- “Efficient and smooth” approach to SMP2 simulation development
- Support for all phases of the simulation lifecycle



SIMULATION LIFECYCLE: REQUIREMENTS

- Requirements are specified by the customer (generic and specific)
- Simulator development team imports requirements into design tool
- Simulator design can be started



Requirements Import Wizard

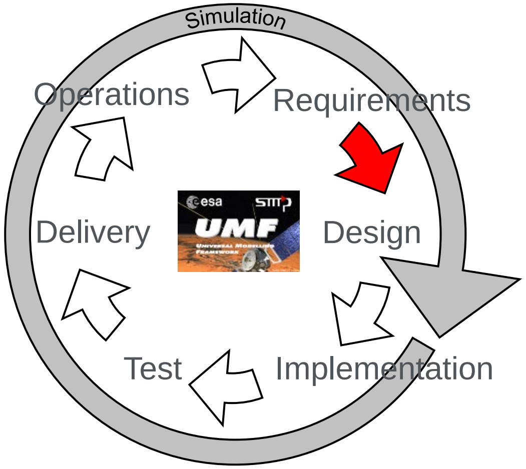
Import Requirements

Import requirements from a CSV or XML file in the local file system into a new UML model in an existing project.

Select input file: X:\BCSIM\Src\esa.bcsim\umf\esa.bcsim.req.csv

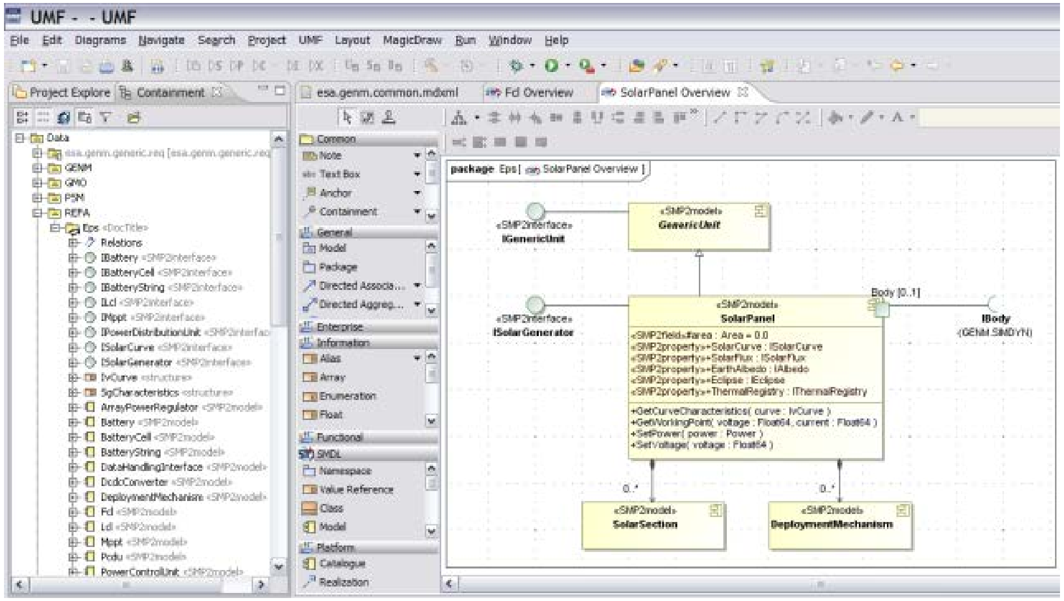
Enter or select the parent folder:

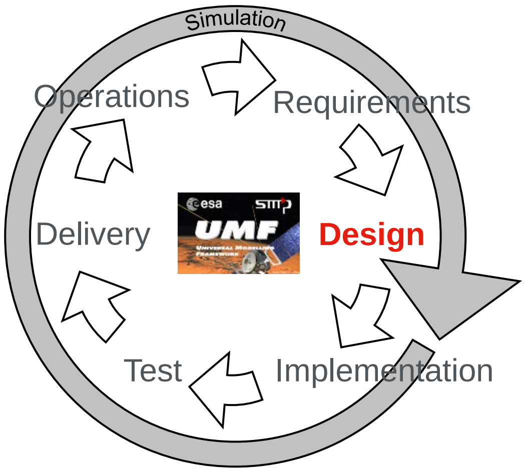
esa.bcsim/umf



SIMULATION LIFECYCLE: ARCHITECTURE & INTERFACE DESIGN

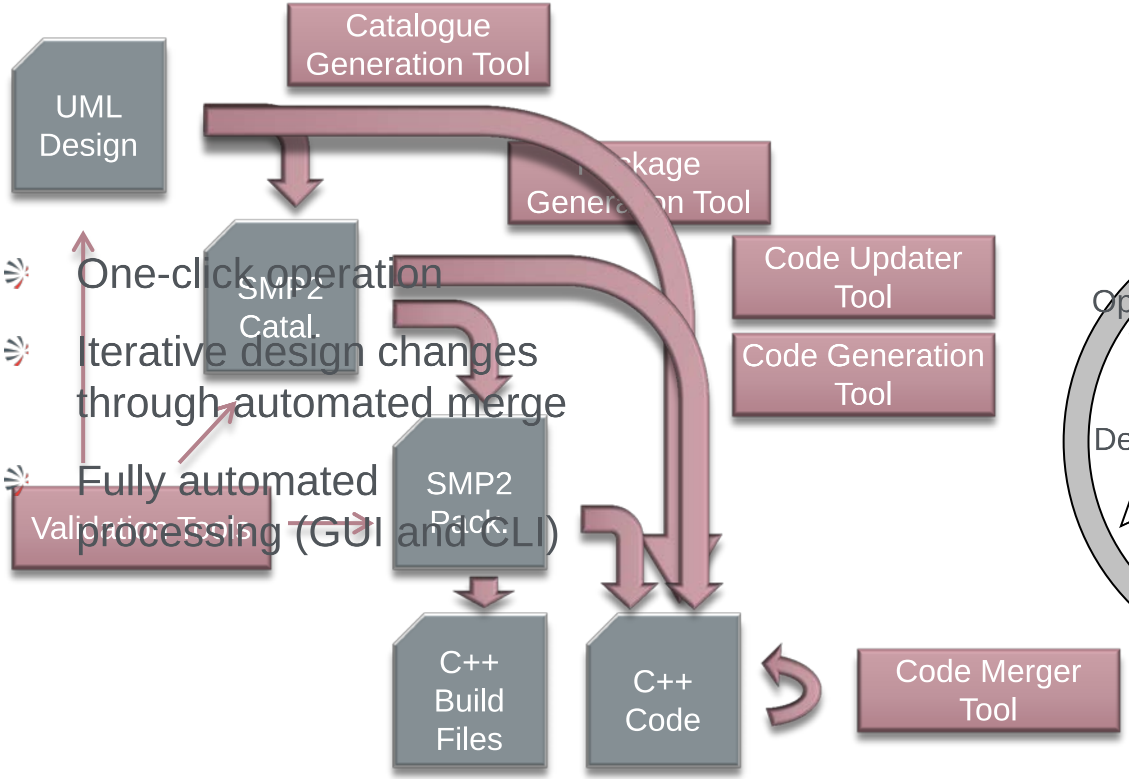
- UMF provides an integrated UML tool for SMP2 modelling
- Customised configuration to focus user interface on SMP2 concepts
- Existing UML/SMP2 models are referenced via appropriate dependencies


UMF - - UMF
File Edit Diagrams Navigate Search Project UMF Layout MagicDraw Run Window Help
Project Explore
Containment
Data
esa.genm.generic.req [esa.genm.generic.req
GENM
GMO
PSM
REFA
Eps «DocTitle»
Relations
IBattery «SMP2interface»
IBatteryCell «SMP2interface»
IBatteryString «SMP2interface»
ILcl «SMP2interface»
IMppt «SMP2interface»
IPowerDistributionUnit «SMP2interfac
ISolarCurve «SMP2interface»
ISolarGenerator «SMP2interface»
IvCurve «structure»
SgCharacteristics «structure»
ArrayPowerRegulator «SMP2model»
Battery «SMP2model»
BatteryCell «SMP2model»
BatteryString «SMP2model»
DataHandlingInterface «SMP2model»
DcdcConverter «SMP2model»
DeploymentMechanism «SMP2model»
Fcl «SMP2model»
Lcl «SMP2model»
Mppt «SMP2model»
Pcdu «SMP2model»
PowerControlUnit «SMP2model»
esa.genm.common.mdxml
Fcl Overview
SolarPanel Overview
Common
Note
Text Box
Anchor
Containment
General
Model
Package
Directed Associa...
Directed Aggreg...
Enterprise
Information
Alias
Array
Enumeration
Float
Functional
SMDL
Namespace
Value Reference
Class
Model
Platform
Catalogue
Realization
package Eps [SolarPanel Overview]
«SMP2interface»
IGenericUnit
«SMP2model»
GenericUnit
«SMP2interface»
ISolarGenerator
«SMP2model»
SolarPanel
«SMP2field»#area : Area = 0.0
«SMP2property»+SolarCurve : ISolarCurve
«SMP2property»+SolarFlux : ISolarFlux
«SMP2property»+EarthAlbedo : IAlbedo
«SMP2property»+Eclipse : IEclipse
«SMP2property»+ThermalRegistry : IThermalRegistry
+GetCurveCharacteristics(curve : IvCurve)
+GetWorkingPoint(voltage : Float64, current : Float64)
+SetPower(power : Power)
+SetVoltage(voltage : Float64)
Body [0..1]
IBody
(GENM.SIMDYN)
0..*
0..*
«SMP2model»
SolarSection
«SMP2model»
DeploymentMechanism

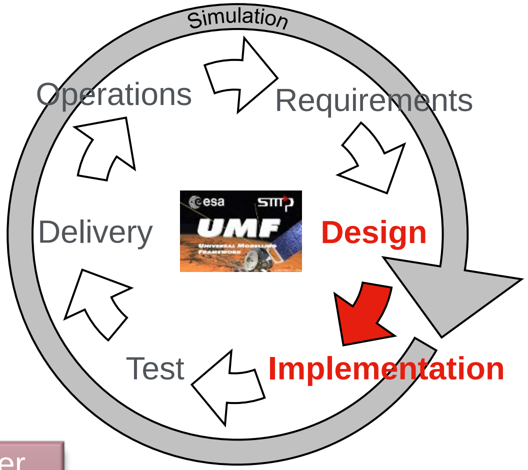


SIMULATION LIFECYCLE: DESIGN TO IMPLEMENTATION

- UMF provides a seamless path from UML to an executable simulator
- Tools can either be run independently or combined to support different sources of SMP2 models while providing a high level of usability

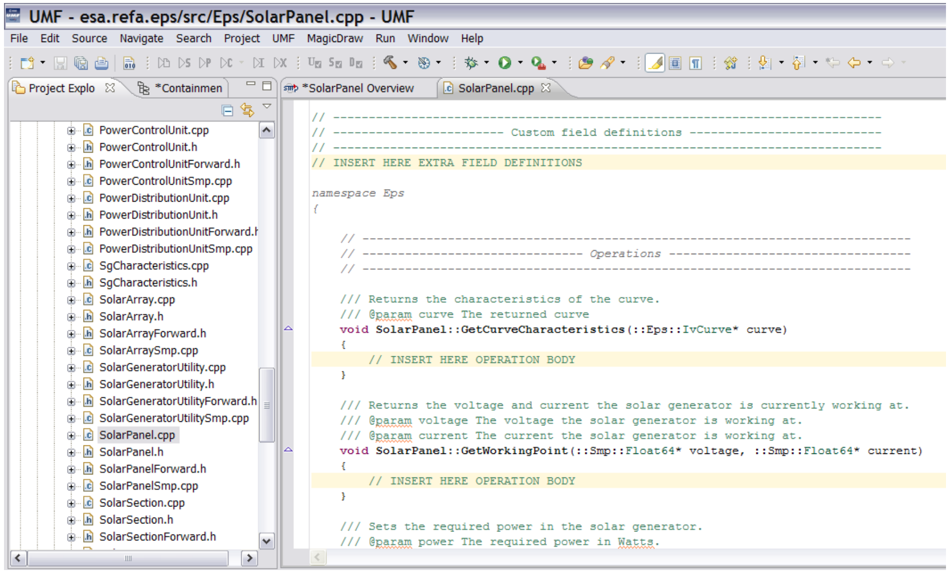


- One-click operation
- Iterative design changes through automated merge
- Fully automated processing (GUI and CLI)

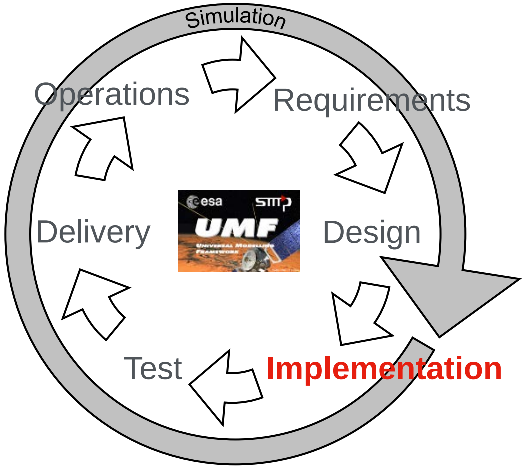


SIMULATION LIFECYCLE: IMPLEMENTATION

- UMF provides and generates/updates a CMake based build system
- UMF is fully integrated with Eclipse C++ Development Tools (CDT)
- Various development related tasks are supported via make targets
- Fully configurable and customizable to support different user/project needs

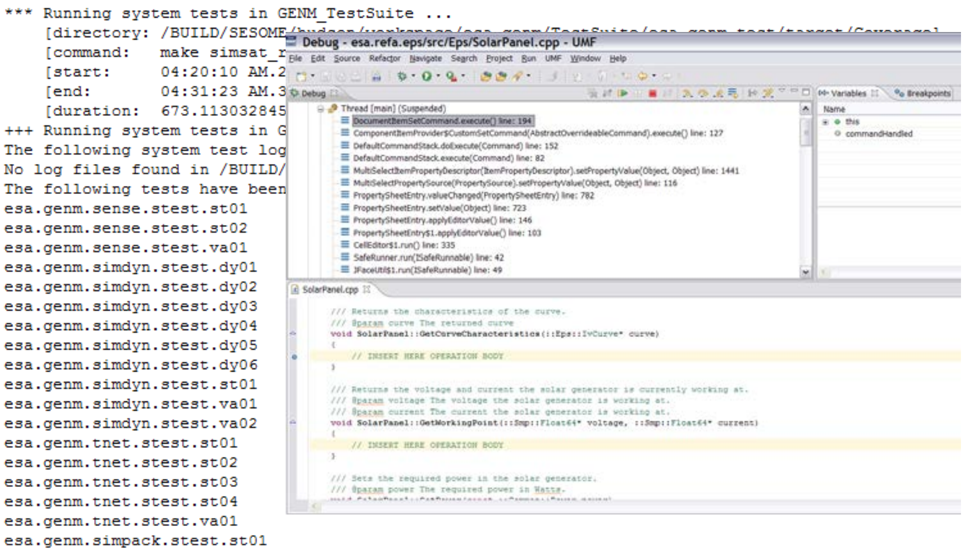


```
// ---------------------------------------------------------------------------
// ---------------------- Custom field definitions --------------------------
// ---------------------------------------------------------------------------
// INSERT HERE EXTRA FIELD DEFINITIONS

namespace Eps
{

    // ---------------------------------------------------------------------------
    // ------------------------------ Operations ---------------------------------
    // ---------------------------------------------------------------------------

    /// Returns the characteristics of the curve.
    /// @param curve The returned curve
    void SolarPanel::GetCurveCharacteristics(::Eps::IvCurve* curve)
    {
        // INSERT HERE OPERATION BODY
    }

    /// Returns the voltage and current the solar generator is currently working at.
    /// @param voltage The voltage the solar generator is working at.
    /// @param current The current the solar generator is working at.
    void SolarPanel::GetWorkingPoint(::Smp::Float64* voltage, ::Smp::Float64* current)
    {
        // INSERT HERE OPERATION BODY
    }

    /// Sets the required power in the solar generator.
    /// @param power The required power in Watts.
```



SIMULATION LIFECYCLE: TESTING AND DEBUGGING

- UMF provides a **Unit and Integration Test Harness** for SMP2 models
  - Stand-alone SMP2 runtime integrated with CppUnit test framework
- UMF provides an integrated **Debugger Facility** based on Eclipse CDT
  - Debugging of SMP2 models directly in UMF or in the target environment

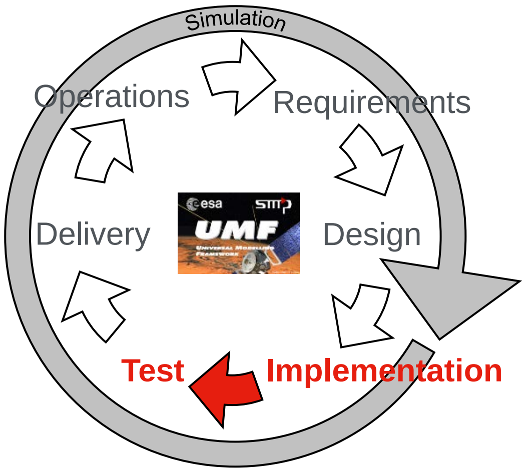


```
*** Running system tests in GENM_TestSuite ...
    [directory: /BUILD/SESOME
    [command:   make simsat_r
    [start:     04:20:10 AM.2
    [end:       04:31:23 AM.3
    [duration:  673.113032845
+++ Running system tests in G
The following system test log
No log files found in /BUILD/
The following tests have been
esa.genm.sense.stest.st01
esa.genm.sense.stest.st02
esa.genm.sense.stest.va01
esa.genm.simdyn.stest.dy01
esa.genm.simdyn.stest.dy02
esa.genm.simdyn.stest.dy03
esa.genm.simdyn.stest.dy04
esa.genm.simdyn.stest.dy05
esa.genm.simdyn.stest.dy06
esa.genm.simdyn.stest.st01
esa.genm.simdyn.stest.va01
esa.genm.simdyn.stest.va02
esa.genm.tnet.stest.st01
esa.genm.tnet.stest.st02
esa.genm.tnet.stest.st03
esa.genm.tnet.stest.st04
esa.genm.tnet.stest.va01
esa.genm.simpack.stest.st01
```

Debug - esa.refa.eps/src/Eps/SolarPanel.cpp - UMF
File Edit Source Refactor Navigate Search Project Run UMF Window Help
Debug
Variables
Breakpoints
Name
this
commandHandled
SolarPanel.cpp

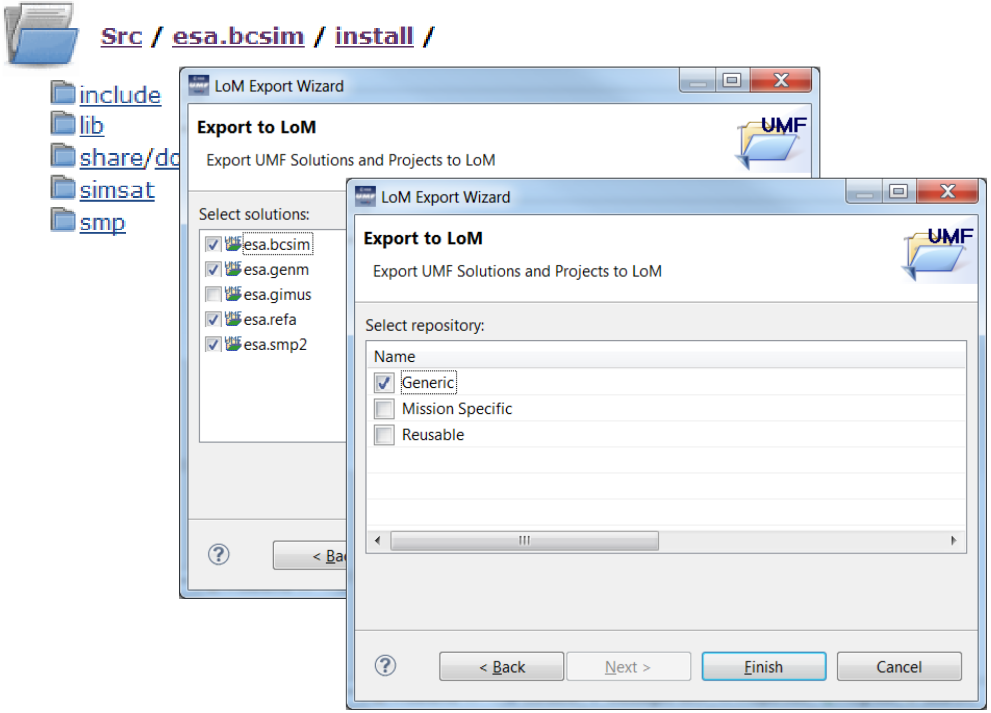
```
Thread [main] (Suspended)
    DocumentItemSetCommand.execute() line: 194
    ComponentItemProvider$CustomSetCommand(AbstractOverrideableCommand).execute() line: 127
    DefaultCommandStack.doExecute(Command) line: 152
    DefaultCommandStack.execute(Command) line: 82
    MultiSelectItemPropertyDescriptor(ItemPropertyDescriptor).setPropertyValue(Object, Object) line: 1441
    MultiSelectPropertySource(PropertySource).setPropertyValue(Object, Object) line: 116
    PropertySheetEntry.valueChanged(PropertySheetEntry) line: 782
    PropertySheetEntry.setValue(Object) line: 723
    PropertySheetEntry.applyEditorValue() line: 146
    PropertySheetEntry$1.applyEditorValue() line: 103
    CellEditor$1.run() line: 335
    SafeRunner.run(ISafeRunnable) line: 42
    JFaceUtil$1.run(ISafeRunnable) line: 49
```

```
/// Returns the characteristics of the curve.
/// @param curve The returned curve
void SolarPanel::GetCurveCharacteristics(::Eps::IvCurve* curve)
{
    // INSERT HERE OPERATION BODY
}

/// Returns the voltage and current the solar generator is currently working at.
/// @param voltage The voltage the solar generator is working at.
/// @param current The current the solar generator is working at.
void SolarPanel::GetWorkingPoint(::Smp::Float64* voltage, ::Smp::Float64* current)
{
    // INSERT HERE OPERATION BODY
}

/// Sets the required power in the solar generator.
/// @param power The required power in Watts.
```



SIMULATION LIFECYCLE: DELIVERY AND DEPLOYMENT

- UMF provides a **Document Generator** to simplify deliveries (ECSS E-40)
- UMF provides packaging mechanisms to simplify deployment
  - Support for dependency management
  - Support for the new Library of Models (LoM) packaging & deployment



Src / esa.bcsim / install /

- include
- lib
- share/dc
- simsat
- smp

LoM Export Wizard

Export to LoM

Export UMF Solutions and Projects to LoM

Select solutions:

☑ esa.bcsim
☑ esa.genm
□ esa.gimus
☑ esa.refa
☑ esa.smp2

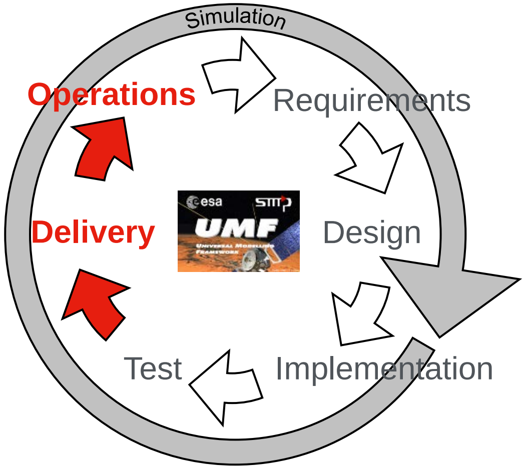
LoM Export Wizard

Export to LoM

Export UMF Solutions and Projects to LoM

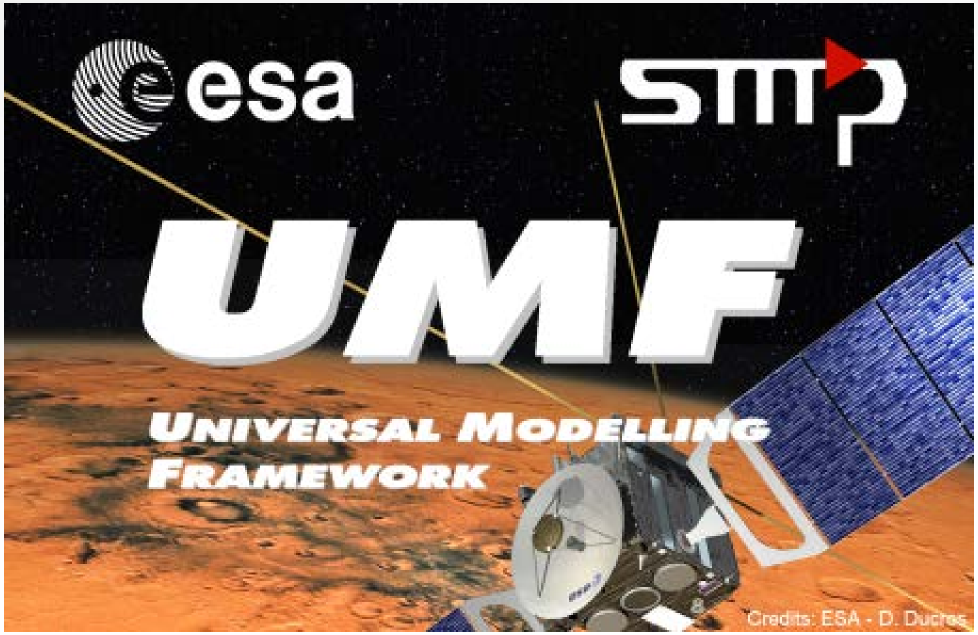
Select repository:

Name
☑ Generic
□ Mission Specific
□ Reusable



UMF Concepts and Features

Dependency Management and Deployment



UMF SOLUTIONS AND PROJECTS (1)

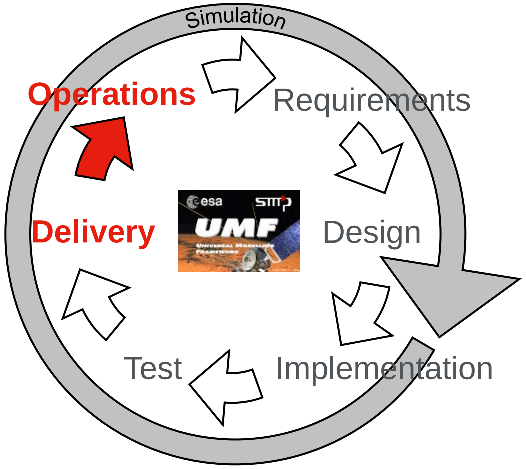
- Design goals of the new UMF **Dependency Management** facility
  - Support large scale simulation developments with model reuse or CFIs
  - Simplify delivery and deployment of the developed simulator
  - Support more Agile processes with frequent deliveries

Solution

- Top-level **grouping** mechanism
- Specifies **dependencies** to other solutions
- Holds **common configuration** for its projects

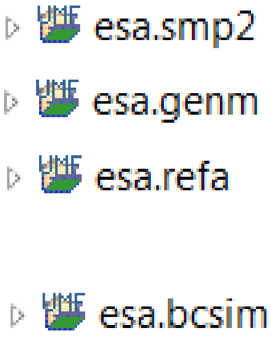
Project

- Contains UML/SMDL design, C++ code,
  and runtime configurations of SMP2 **models**
- Belongs to exactly one solution

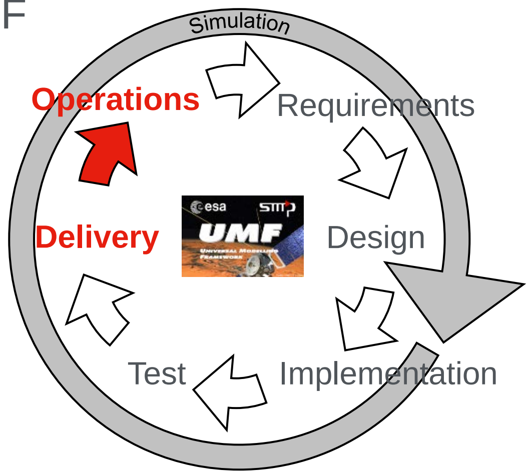


UMF SOLUTIONS AND PROJECTS (2)

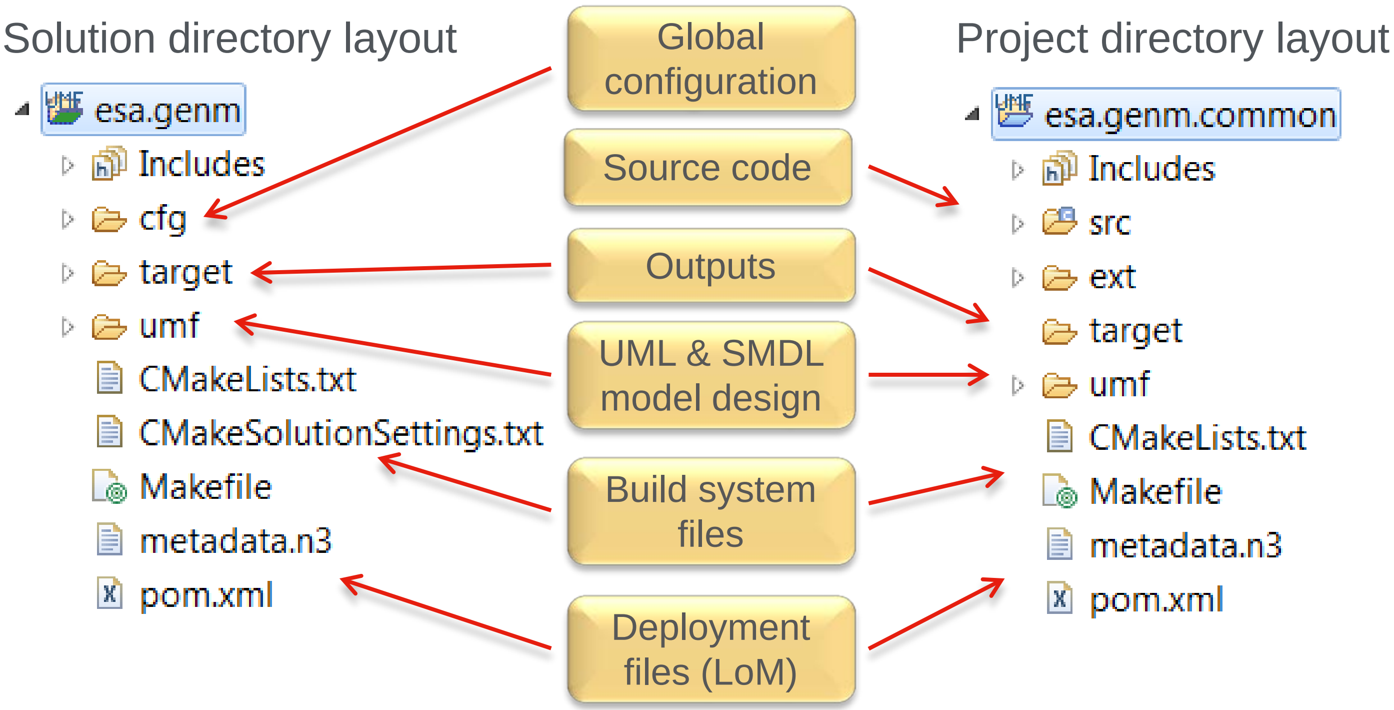
- **Convention over configuration** approach
  - UMF largely defines directory layout and naming in solutions/projects
  - In line with Eclipse CDT requirements and LoM packaging approach
- Solutions form a **dependency tree**
  - Only direct dependencies need to be specified
  - Indirect dependencies are auto-detected by UMF
- Solutions typically involved in the context of an ESOC operational S/C simulator



esa.smp2	SMP2 basics, e.g. MDK (in UMF)
esa.genm	ESOC Generic Models (CFI)
esa.refa	ESOC Spacecraft Simulator Reference Architecture (CFI)
esa.bcsim	Operational S/C Simulator (under development)



UMF SOLUTIONS AND PROJECTS (3)



Solution directory layout

- esa.genm
  - Includes
  - cfg
  - target
  - umf
  - CMakeLists.txt
  - CMakeSolutionSettings.txt
  - Makefile
  - metadata.n3
  - pom.xml

Global configuration

Source code

Outputs

UML & SMDL model design

Build system files

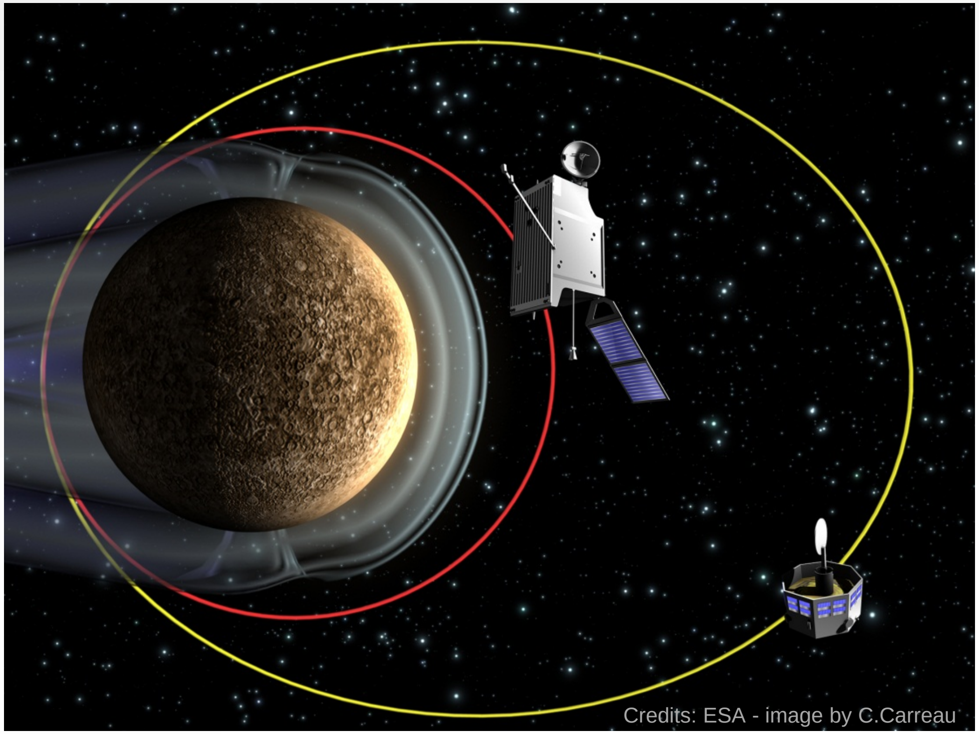
Deployment files (LoM)

Project directory layout

- esa.genm.common
  - Includes
  - src
  - ext
  - target
  - umf
  - CMakeLists.txt
  - Makefile
  - metadata.n3
  - pom.xml

Case Study

The BepiColombo Simulator (BCSIM)



Credits: ESA - image by C.Carreau

BCSIM OVERVIEW

- Project kicked off early 2012
- **Distributed** development team across three geographies
  - Different parts of the simulator are assigned to consortium members
  - Use shared development environment
- **Agile** development process with frequent Sprint deliveries
  - Automation is crucial to reduce overheads
  - Continuous integration and test to meet quality requirements
- **Baseline** is to use latest SimSDE products for development
  - UMF v2, REFA v2, and GENM v5
  - Continuous feedback to the SimSDE team leads to many UMF usability and performance improvements

BCSIM APPROACH

- **Project break-down** via UMF solutions and projects concept
  - Source hierarchy: *esa.bcsim* → *esa.refa* → *esa.genm* → *esa.smp2*
  - Test Suite hierarchy: *esa.bcsim.test* → *esa.bcsim, esa.genm.test*
  - Break-down of source solution into 30 projects
- **Automated build and test** of the simulator
  - Setup makes use of all UMF capabilities
  - Automation is achieved via **continuous integration** (Hudson server)
- **Binary delivery and deployment**
  - ISO files are created automatically after automated build & test
  - Full simulator is **installed via script**, including CFIs such as SIMSAT, GROUND, SLEGM and the ESOC emulator (reliable & reproducible)

SUMMARY AND CONCLUSIONS

- Experience and initial feedback from BCSIM
  - ✓ Setup and automation via UMF v2 mechanisms works well in practice
  - ✓ UMF v2 was easy to introduce to the team and is well accepted
  - ! Overall turn-around time (from design change to C++ code update) is a critical area for developer acceptance, largely improved in UMF v2
  - ! Setup with shared storage (SAN) is problematic (high I/O load)
  - ✓ UMF v2 is a robust and productive SMP2 development environment

- Current Status
  - UMF v2 (as part of SimSDE) is in Provisional Acceptance (PA) phase
  - UMF v2 release planned in 2012