Diffusion Limitée



SKE: emulation of on-board computer environments in operational simulators

Fabien GUNST (FENTISS) – fgunst@fentiss.com
Julien Galizzi (CNES) – julien.galizzi@cnes.fr







Diffusion Limitée



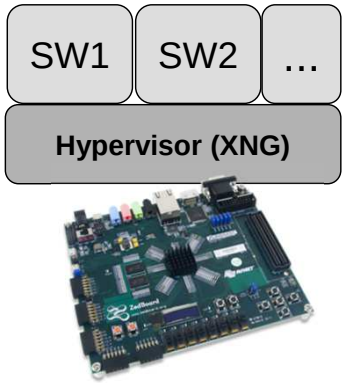
SKE



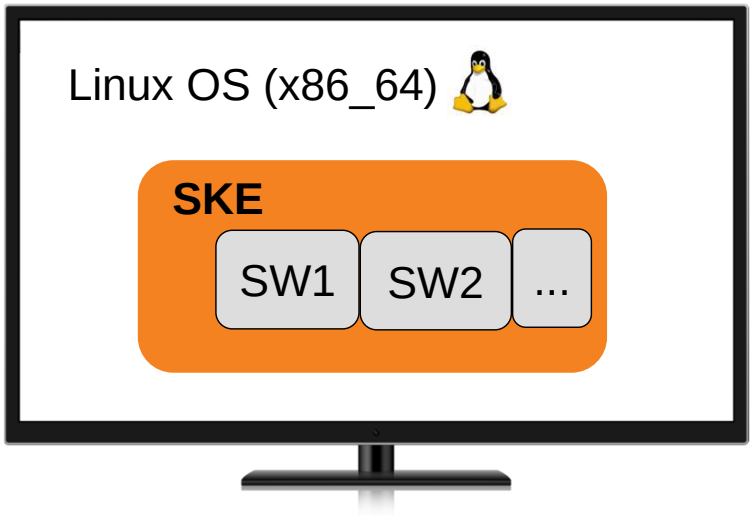
FENTiss
FENT INNOVATIVE SOFTWARE SOLUTIONS

cnes

SKE in a nutshell



A **SW** that **enables** the execution...



… of target flight software...

of a **partitioned** system designed to **operate on XNG** on a specific **HW**,

… in a Linux environment.

in a **general purpose** machine, by using a monitoring SW that **abstracts** HW, **mimics XNG** and provides **emulation** of the **partitioned** environment.

FENTISS FENT INNOVATIVE SOFTWARE SOLUTIONS cnes

SKE as support product of XNG

- **Access product to XNG** for new customers



- **Development of FSW applications** for XNG systems in Linux environments
  - Applications to execute on host CPUs
  - Avoid HW complexities for non-HW-oriented developers
  - HW is expensive and not always at hand
  - Easy-to-use SW development environment provided by Linux
  - Useful for testing and debugging





- Particularly relevant for **faster-than-real-time simulation** of long FSW scenarios
  - Powerful processors in Linux host computers/servers wrt target space processors



SKE as TSP / XNG emulator

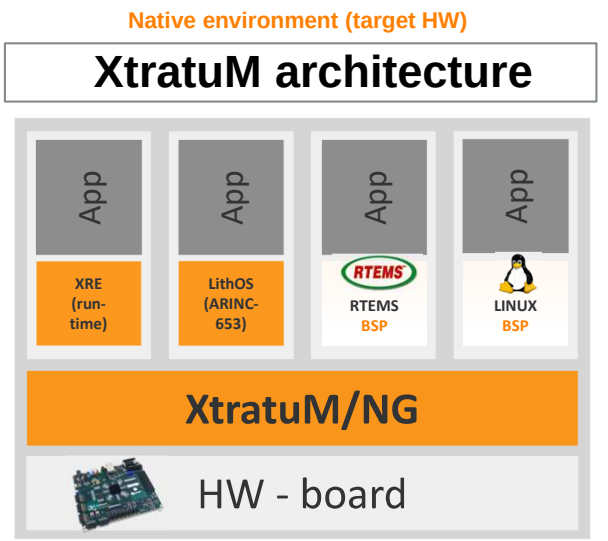
SKE is SW application for Linux host environments that:

1. Provides **TSP** (**XtratuM-like architecture) emulation**:
   i. *Spatial isolation* → by memory isolation of processes
   ii. *Temporal isolation* → by implementing a cyclic scheduler with fixed time slots (configuration)
   iii. *Virtualization of low level layers* → by means of abstracting and giving functional representativity for some HW resources (e.g. memory) by using mechanisms provided by the Linux environment

1. Provides **emulation of the XNG hypervisor layer** by offering the same:
   a) Features & functionalities
   b) Configuration
   c) API to partitions

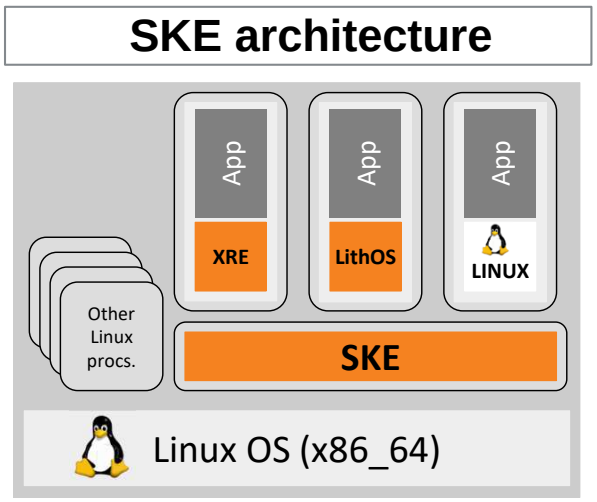
1. Provides **interface for coupling with external models** → allowing the interaction of the SKE ecosystem (SKE, partitions) with other numerical models part of the OBC whose functionality is not provided by SKE (e.g. I/O driver models for I/O emulation)

Native environment (target HW)



VS

Host environment (Linux OS)



Diffusion Limitée



The SWARM exploratory project



FENTiss
FENT INNOVATIVE SOFTWARE SOLUTIONS

cnes

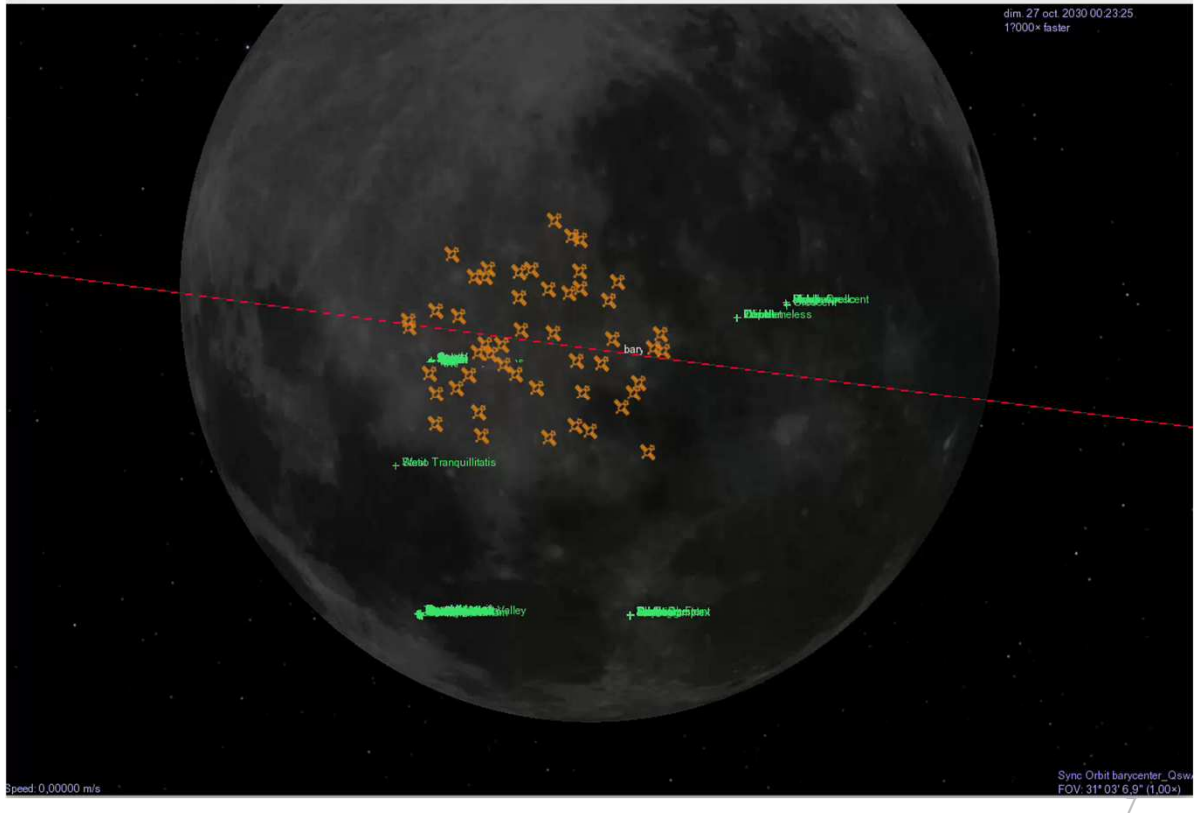
SWARM Project objectives

Swarms are complex, low TRL, multi-satellites systems

CNES launched a pluri-anual, transverse activity to increase the TRL of the technologies involved in such systems and make them achievable

- ISL
- Ground segment
- On board applications: more autonomy !
  - FDIR @ swarm level
  - Absolute & relative navigation
  - Absolute & relative orbit control
  - Vision-based navigation
  - Network Management
  - Ground proxy management
  - Distributed mass memory
  - Distributed Mission Programmation
  - Distributed payload
- Validation means @ swarm level

LIRA - E. Rouille - NOIRE



FENTISS
FENT INNOVATIVE SOFTWARE SOLUTIONS

cnes

Validation strategy

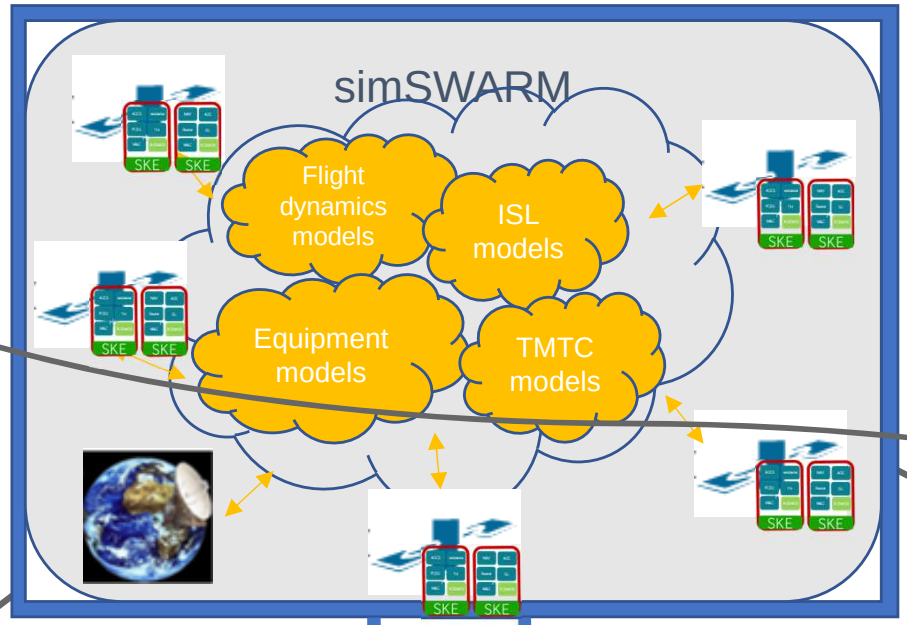
1 simulator

BASILES

Build for host

Discrete-event simulation infrastructure, SMP/SMP2 simulation standard

FSW Gitlab

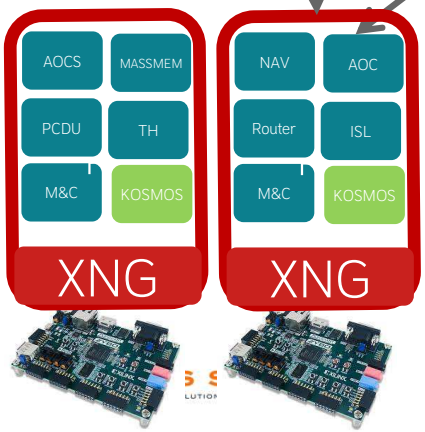


Build for target

Guarantee functional behaviour of the full numerical version of the software with SKE complies with the real one

HW in the loop

Processor in the loop – 1 HW board for each computer



TTC and ISL links managed through UART links

2 configurations

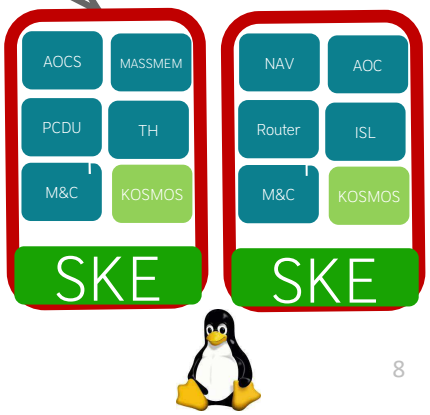
Used for mono-satellite tests with stubbed neighbours – no scalability

TTC and ISL links managed through A653 ports

Full simulated

Used for mono-satellite tests with stubbed neighbours – easily scalable

Used for system tests, with each satellite running the real FSW simulated



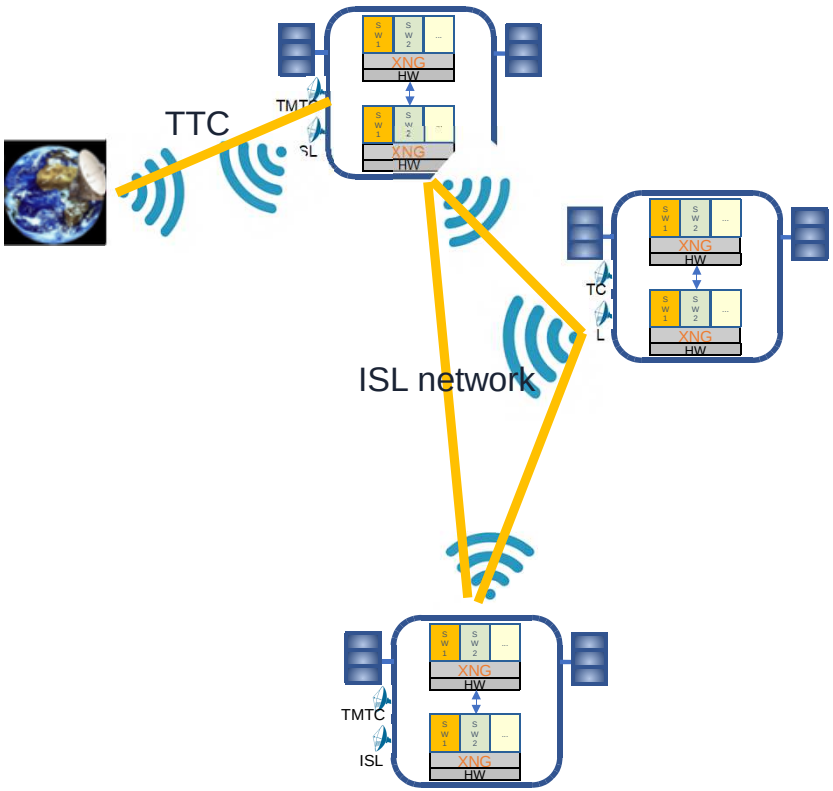
SWARM simulator view

```
Every 1.0s: ps -eo psr,pcpu,pmem,cmd --sort=-pcpu | grep ske tu-apis-p01.cst.cnes.fr: Mon Sep 29 14:26:39 2025

 11 21.4  0.1 ske_bsl_standalone 90 91
 42 21.2  0.1 ske_bsl_standalone 109 110
 24 21.1  0.1 ske_bsl_standalone 71 72
 17  9.5  0.0 nav-xng_ske-debug.elf
 32  9.5  0.0 nav-xng_ske-debug.elf
 28  9.5  0.0 nav-xng_ske-debug.elf
 33  4.5  0.0 apppus-xng_ske-debug.elf
 43  4.5  0.0 apppus-xng_ske-debug.elf
 15  4.5  0.0 apppus-xng_ske-debug.elf
 46  4.0  0.0 isl-xng_ske-debug.elf
 27  4.0  0.0 isl-xng_ske-debug.elf
 36  4.0  0.0 isl-xng_ske-debug.elf
 26  3.9  0.0 rap-xng_ske-debug.elf
  5  3.9  0.0 rap-xng_ske-debug.elf
 23  3.9  0.0 rap-xng_ske-debug.elf
 37  3.5  0.0 instrum-xng_ske-debug.elf
  5  3.5  0.0 instrum-xng_ske-debug.elf
 40  3.5  0.0 instrum-xng_ske-debug.elf
 41  3.4  0.1 ske_bsl_standalone 74 75
  6  3.4  0.1 ske_bsl_standalone 93 94
 45  3.4  0.1 ske_bsl_standalone 112 113
 12  3.1  0.0 ccsw-xng_ske-debug.elf
  5  3.1  0.0 ccsw-xng_ske-debug.elf
  7  3.1  0.0 ccsw-xng_ske-debug.elf
 18  3.1  0.0 router-xng_ske-debug.elf
 29  3.0  0.0 router-xng_ske-debug.elf
 23  3.0  0.0 router-xng_ske-debug.elf
 21  2.9  0.0 gnss-xng_ske-debug.elf
 14  2.9  0.0 gnss-xng_ske-debug.elf
  2  2.9  0.0 gnss-xng_ske-debug.elf
 26  2.6  0.0 ios-xng_ske-debug.elf
 32  2.6  0.0 ios-xng_ske-debug.elf
```

Full numeric SWARM simulator
(real applicative code compiled for Linux x86_64)

simulates



Real SWARM
(real applicative code compiled for target)

SKE representativity limits...

- **High level of functional representativity** that complies with our needs in this exploratory project
  - Still have **some limits** that are **not a real problem at SWARM validation level**

Feature / Functionality	Limitations	Consider
Memory architecture	• Memory map not representative of the native one	• Extra memory areas can be instantiated using POSIX shm objects (transparently created by SKE according to system cfg) • Use specific SKE services to access those mem areas
I/O access	• No direct access from partitions to HW addresses • For indirect access, external interaction out of SKE is required	• Use XNG/SKE specific services for emulating differente types of I/O access: at register or “message passing” level • Coupling with external SW
FPU accuracy	• Host processors much more powerful for vector operations	• Discrepancies with target can be mitigated using a scale factor
Interrupts latency	• Jitter due to host’s standard Linux	• Different latencies than in target • Optimizations in the Linux scheduler, or use of RT Linux.
Partition’s sources	• Flight SW functionality is identical, source code is *almost* identical, binaries are not (due to different compilers)	• Adaptation of sources and build environment is required, with limited impact • However, functional behaviour should be robust wrt to target (same applicative functions are executed in host)

FENTISS FENT INNOVATIVE SOFTWARE SOLUTIONS cnes



Thank you!

Fabien GUNST (FENTISS) – fgunst@fentiss.com
Julien Galizzi (CNES) – julien.galizzi@cnes.fr



FENTISS
FENT INNOVATIVE SOFTWARE SOLUTIONS

cnes