Black Lion: a Software Simulator for Heterogeneous Spaceflight & Mission Components

Mar Cols-Margenet
Graduate Research Assistant, CU Boulder

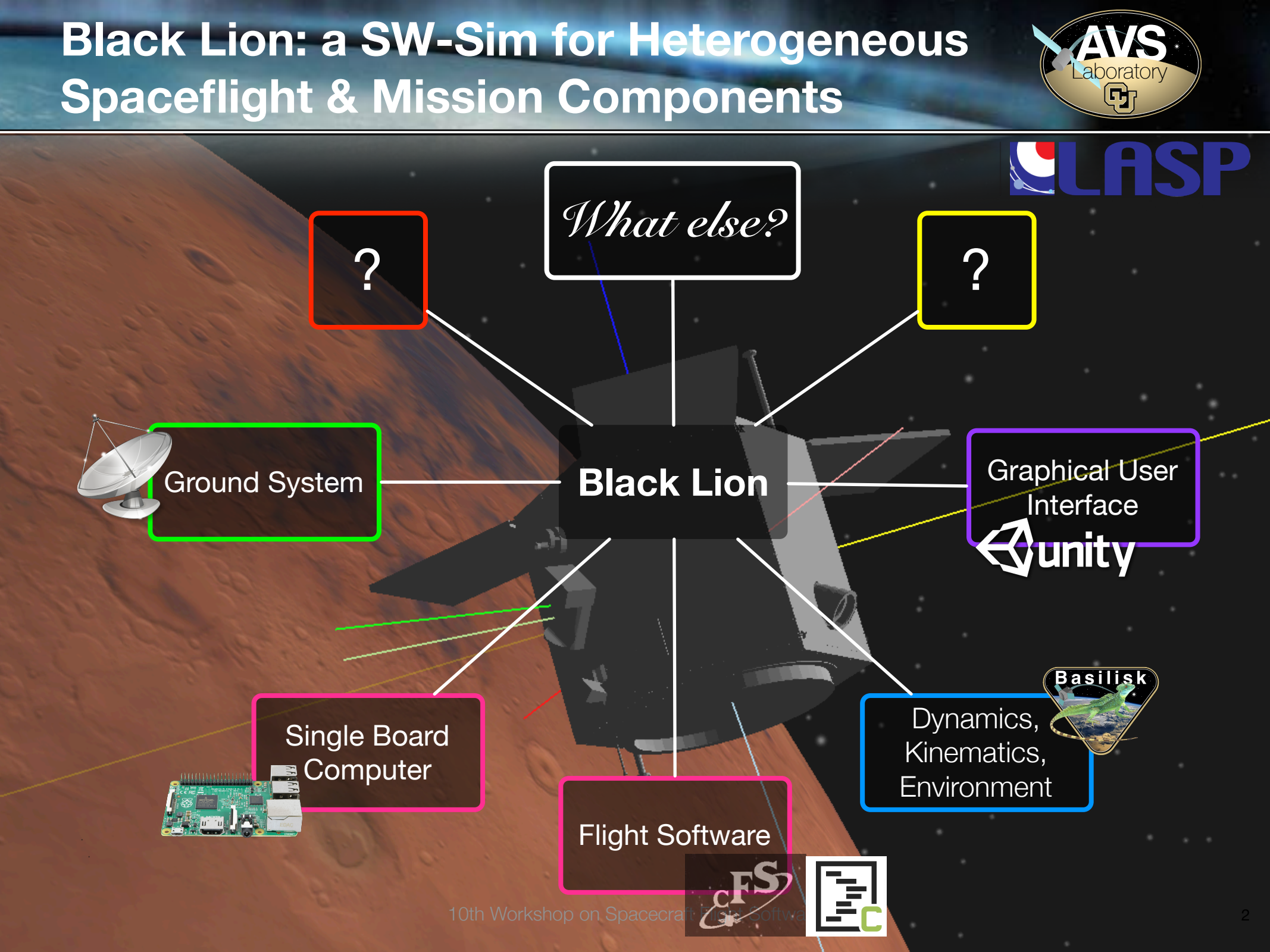
Patrick Kenneally
Graduate Research Assistant, CU Boulder

Hanspeter Schaub
Glenn L. Murphy Endowed Chair, CU Boulder

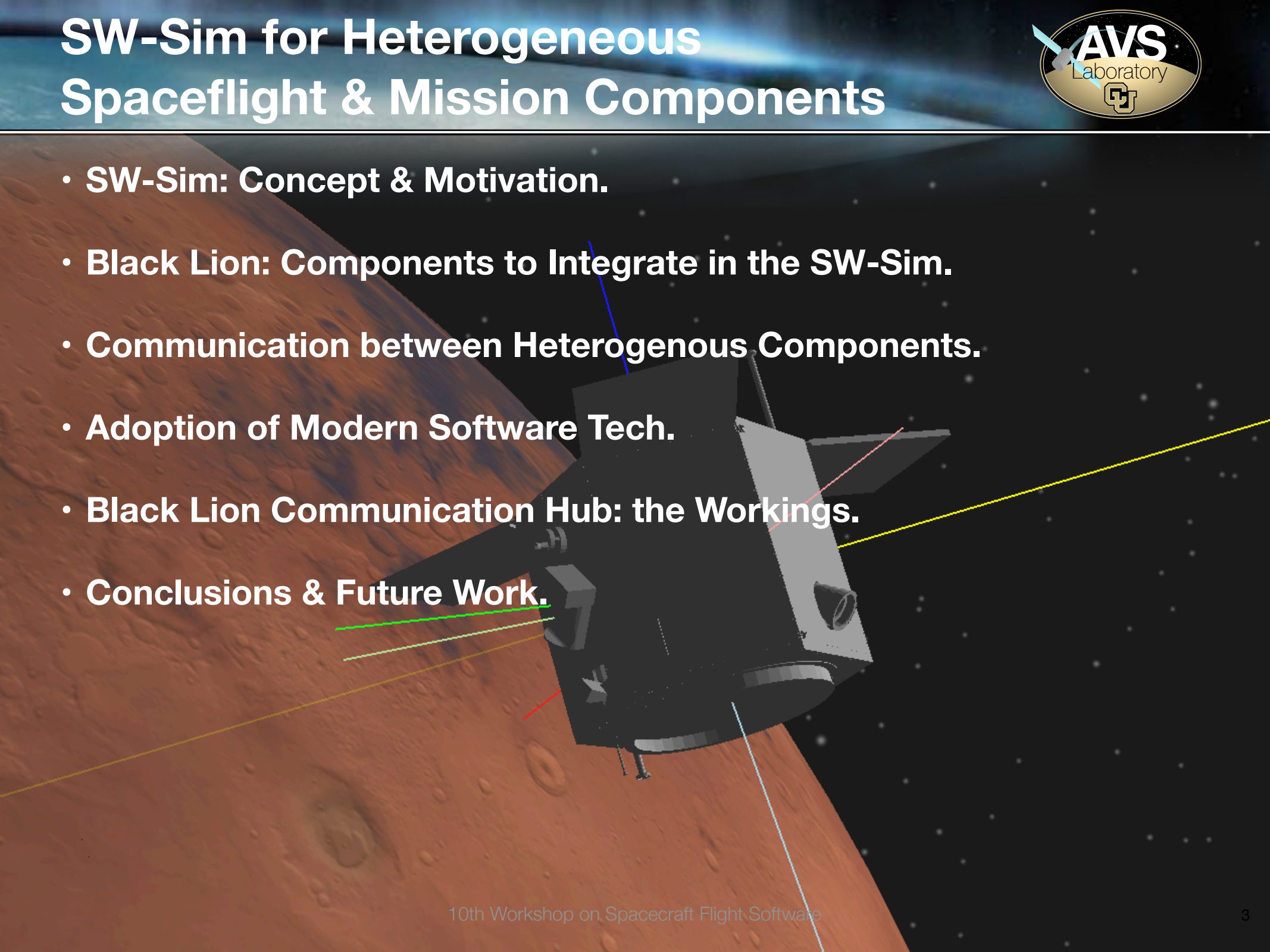
Scott Piggott
ADCS Software Lead, LASP

10th Workshop on Spacecraft Flight Software, The Johns Hopkins University Applied Physics Laboratory, Maryland, Dec. 4 – 7 2017.

Black Lion: a SW-Sim for Heterogeneous Spaceflight & Mission Components

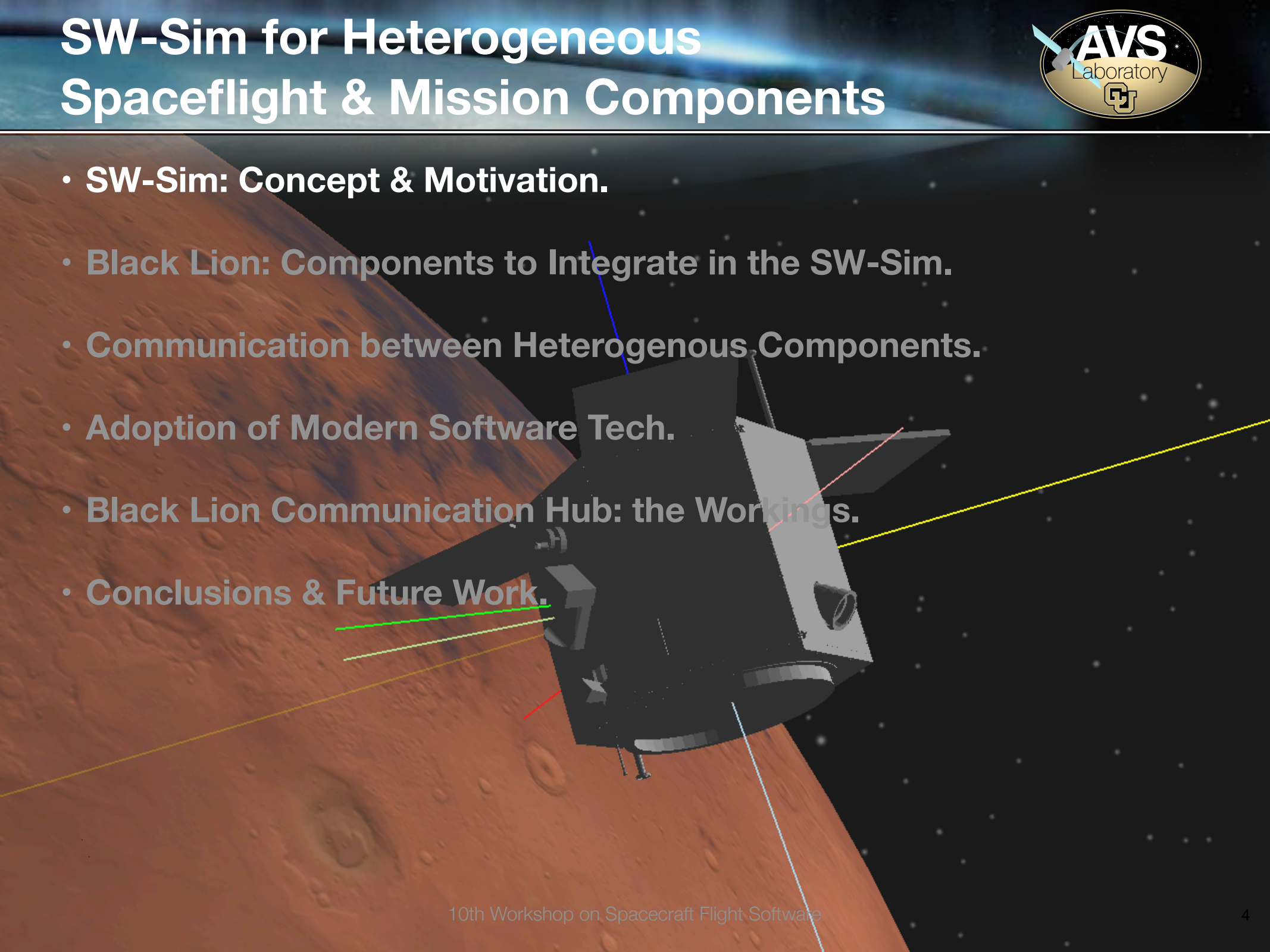


SW-Sim for Heterogeneous Spaceflight & Mission Components



- SW-Sim: Concept & Motivation.
- Black Lion: Components to Integrate in the SW-Sim.
- Communication between Heterogenous Components.
- Adoption of Modern Software Tech.
- Black Lion Communication Hub: the Workings.
- Conclusions & Future Work.

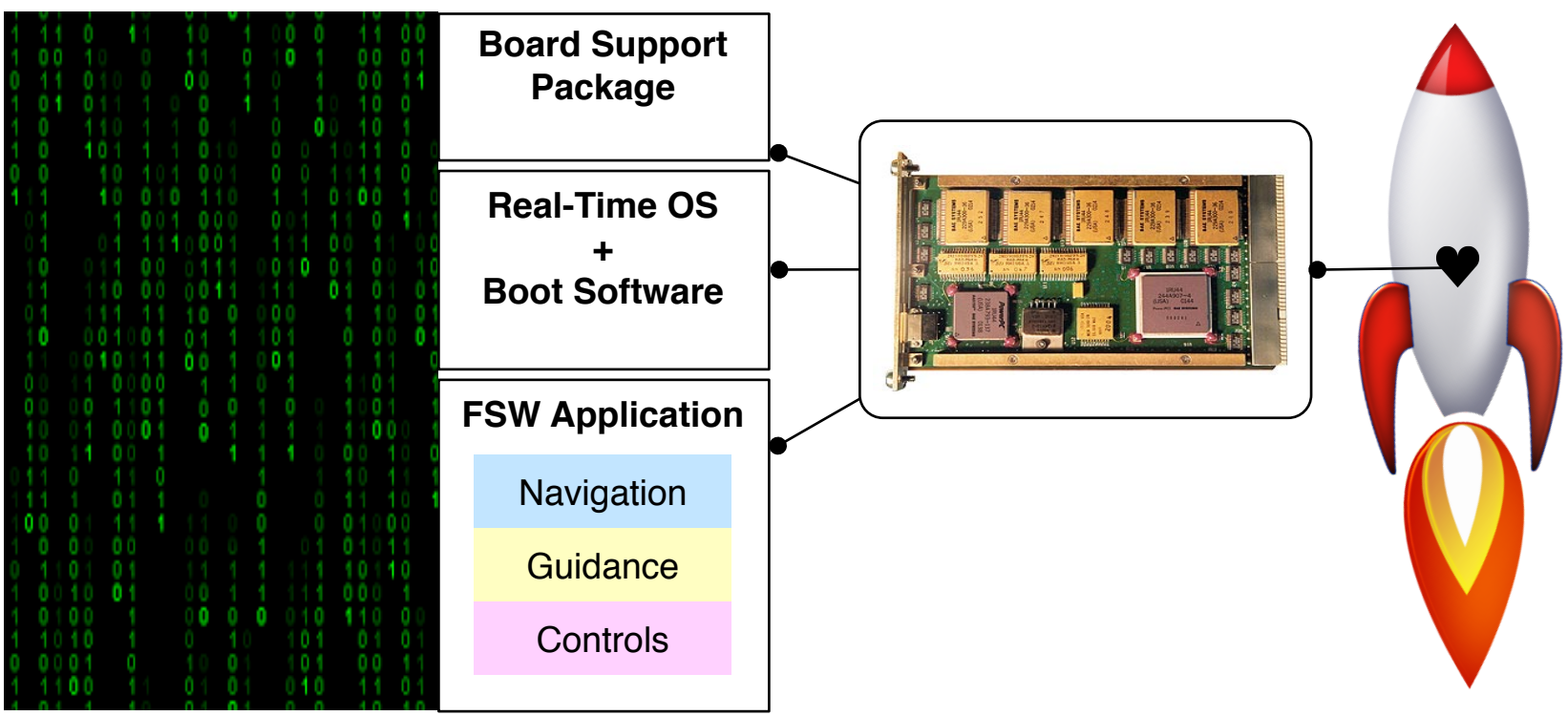
SW-Sim for Heterogeneous Spaceflight & Mission Components



- **SW-Sim: Concept & Motivation.**
- Black Lion: Components to Integrate in the SW-Sim.
- Communication between Heterogenous Components.
- Adoption of Modern Software Tech.
- Black Lion Communication Hub: the Workings.
- Conclusions & Future Work.

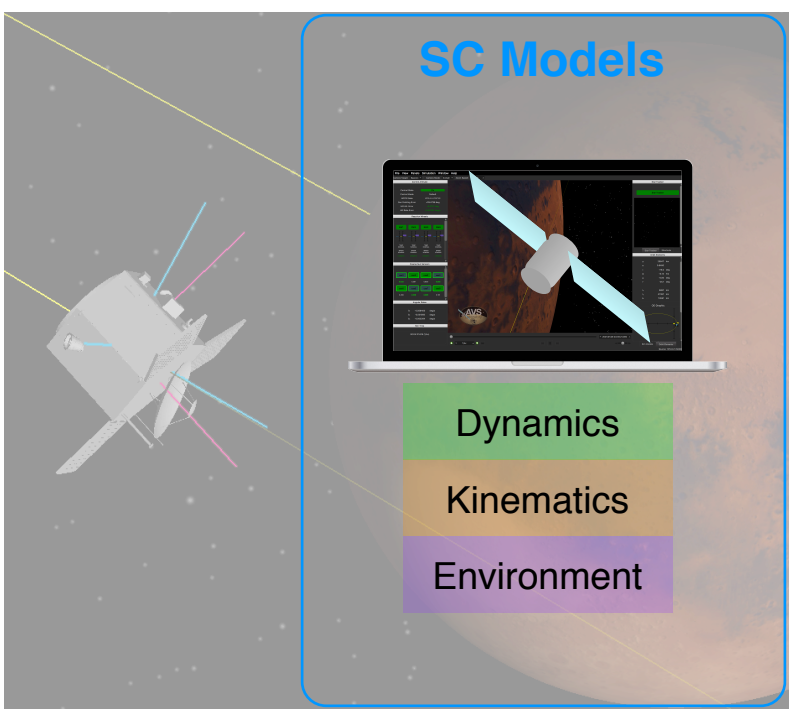
What are we testing?

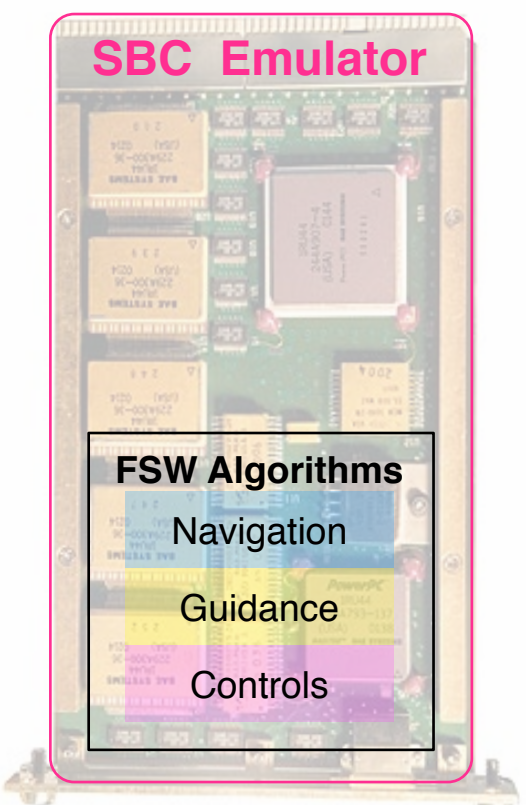
- **Purpose of SW-Sim (& one of Flat-Sat):** testing onboard FSW.
- **Onboard FSW: binary image loaded to spacecraft.**
  FSW App + RTOS & Boot SW + Board Support Package.
- **Testing scope:** analyze FSW response to dynamic conditions and stimuli in a spacecraft operational environment.



How are we testing?

- Testing of FSW components interacting with HW
  - SW-Sim does not replace Flat-Sat, but can **avoid bottle-neck**.
  - SW-Sim advantages: **cost-effective**, **flexible resourcing & early on testing.**

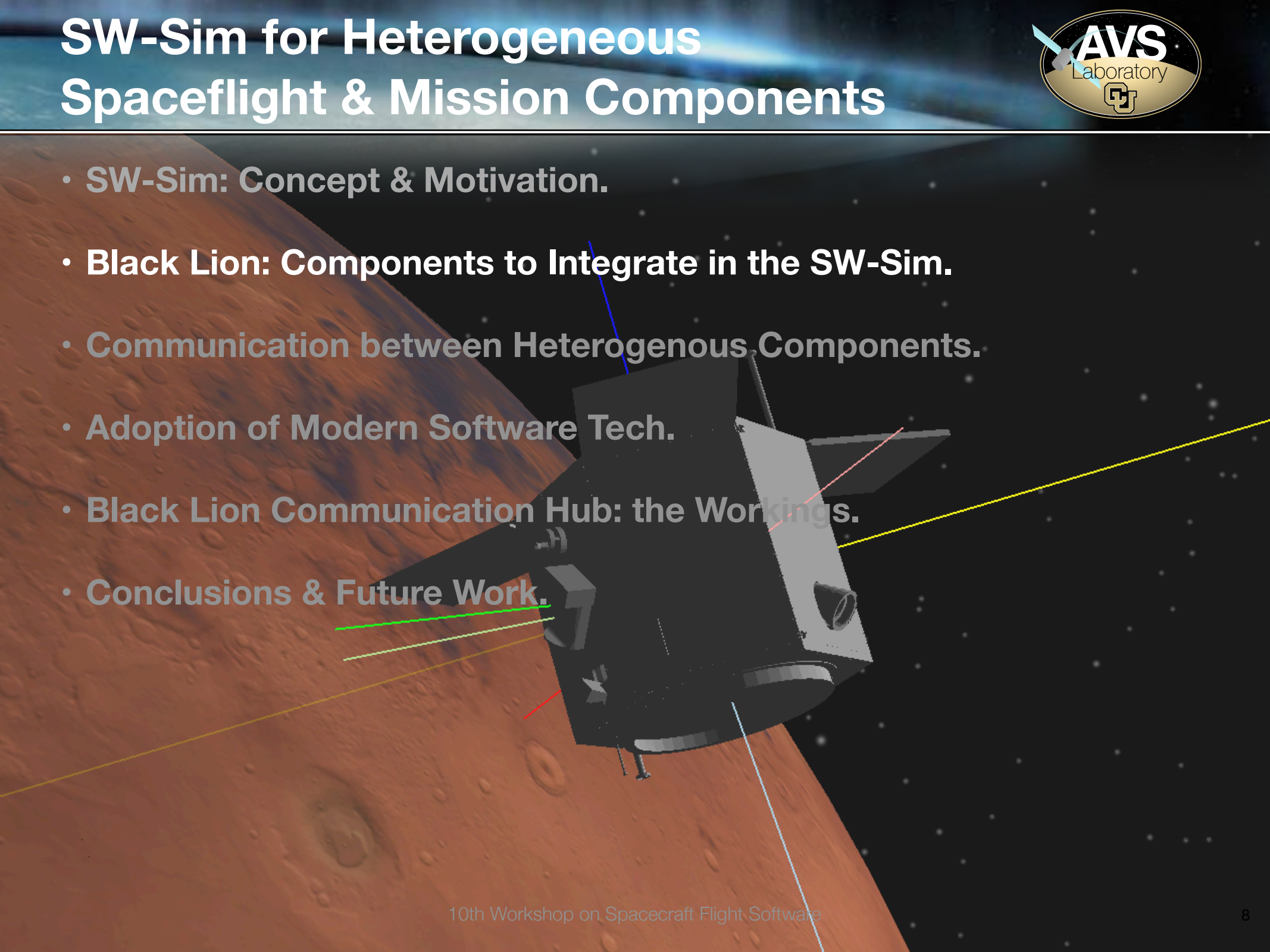




SW-Sim in Use

- **Missions:** James Webb, SLS, Juno, Osiris-Rex…
- **Different flavors of architectures:** in-house development, Lockheed's SoftSim…
- **Common functionality:** Test **unmodified** FSW executable in a software-only environment.

SW-Sim for Heterogeneous Spaceflight & Mission Components



- SW-Sim: Concept & Motivation.
- **Black Lion: Components to Integrate in the SW-Sim.**
- Communication between Heterogenous Components.
- Adoption of Modern Software Tech.
- Black Lion Communication Hub: the Workings.
- Conclusions & Future Work.

Where it all started…
The ADCS FSW Component.

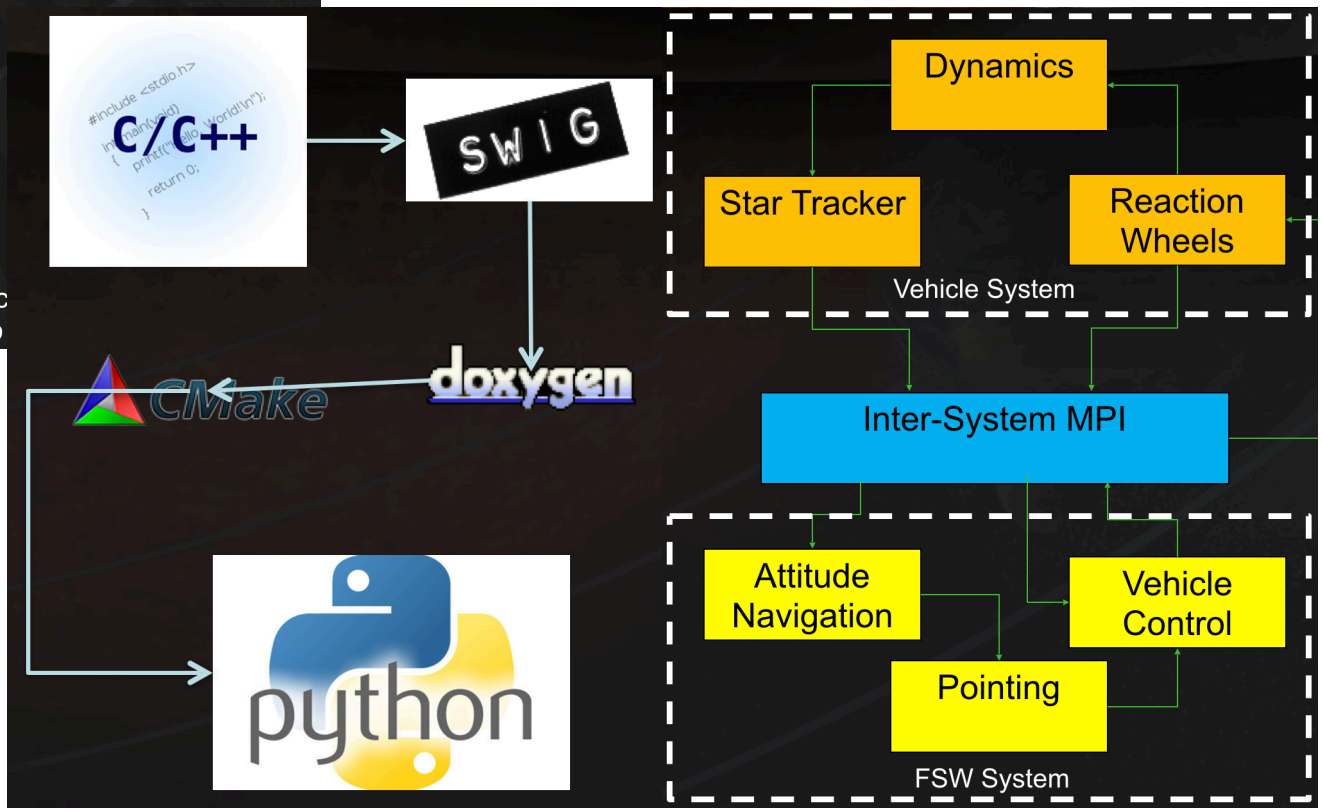
Laboratory for Atmospheric and Space Physics
University of Colorado **Boulder**

Flight Software Development Through Python

Scott Piggott, Maria Cols Margenet, John Alc
P. Kenneally, and Hanspeter Schaub

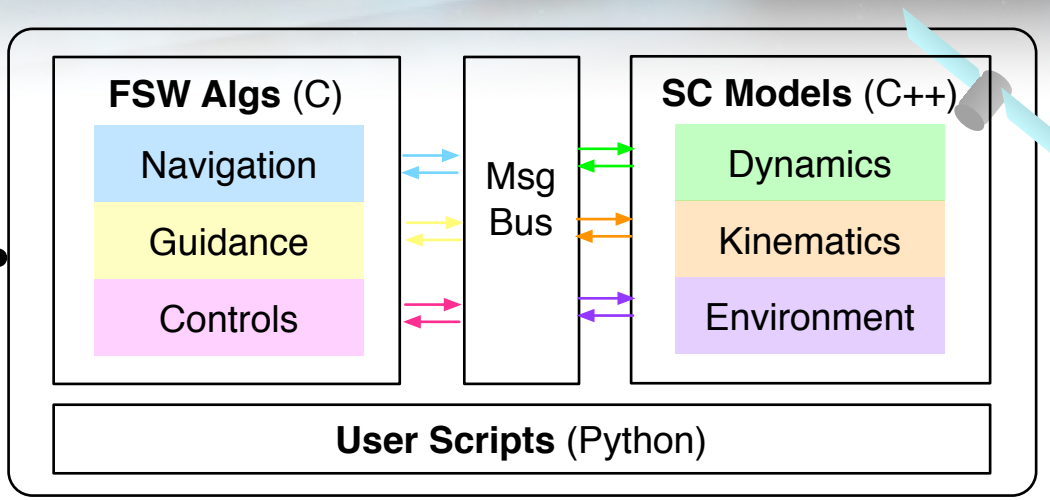
- **LASP & AVS:** ongoing spacecraft interplanetary mission.
- **FSW Workshop'16:** ADCS flight algorithm dev w/ Basilisk.

- **Basilisk:** open-source SW platform for prototyping & testing FSW in realistic closed-loop dynamics simulation.

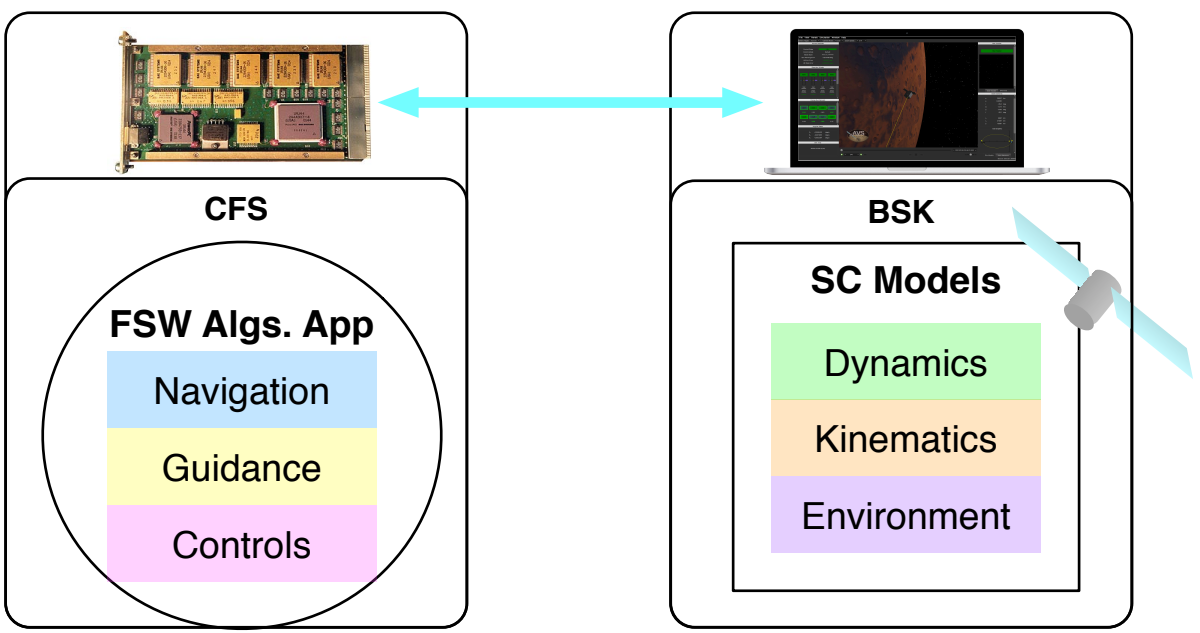


Last time… ADCS FSW development. From BSK to CFS

- **BSK** (Basilisk)**:** dev environment.

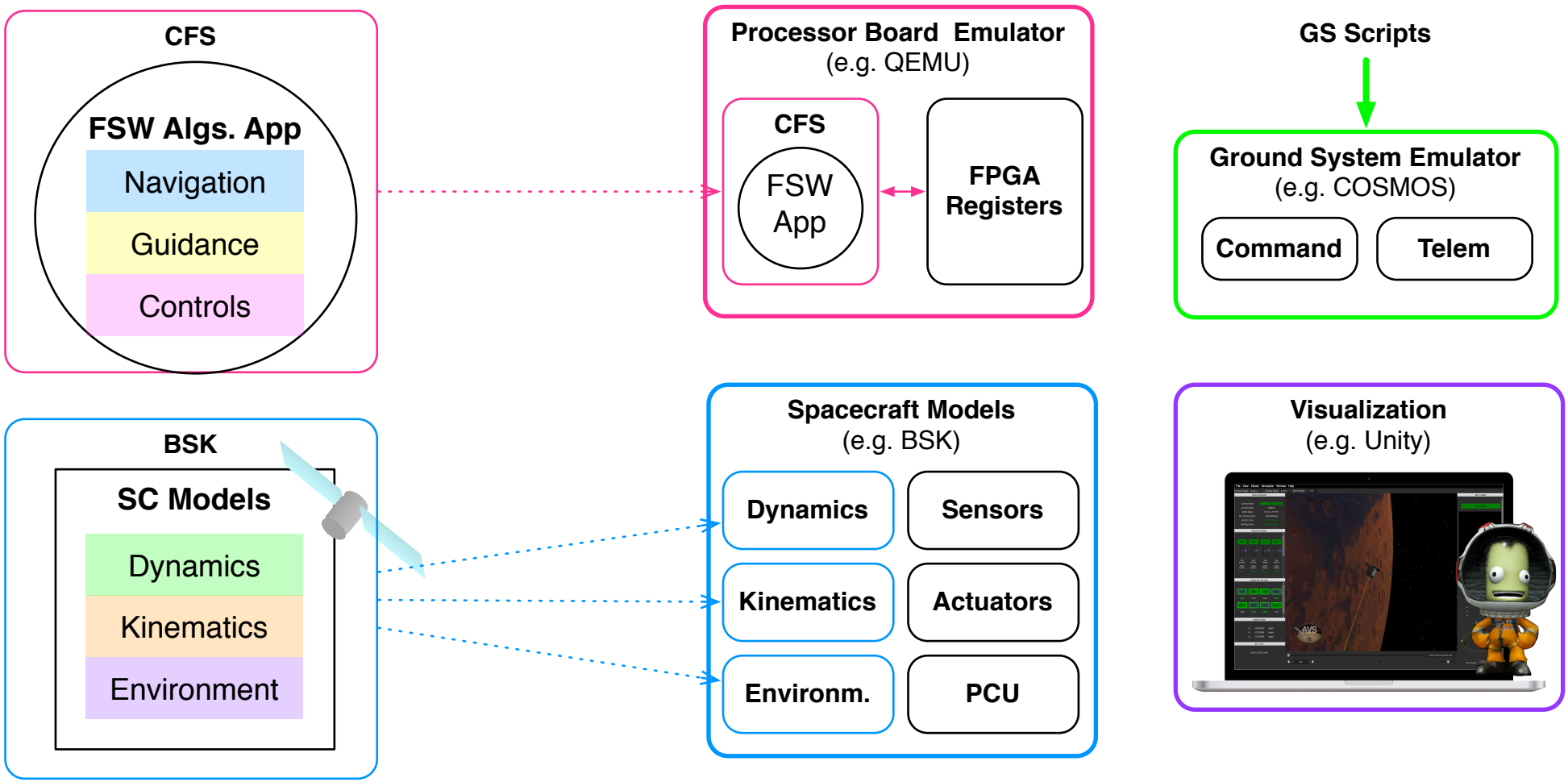


- **CFS** (Core Flight System)**:** migration of the flight application.



Towards SW-Sim: Multiple Nodes.

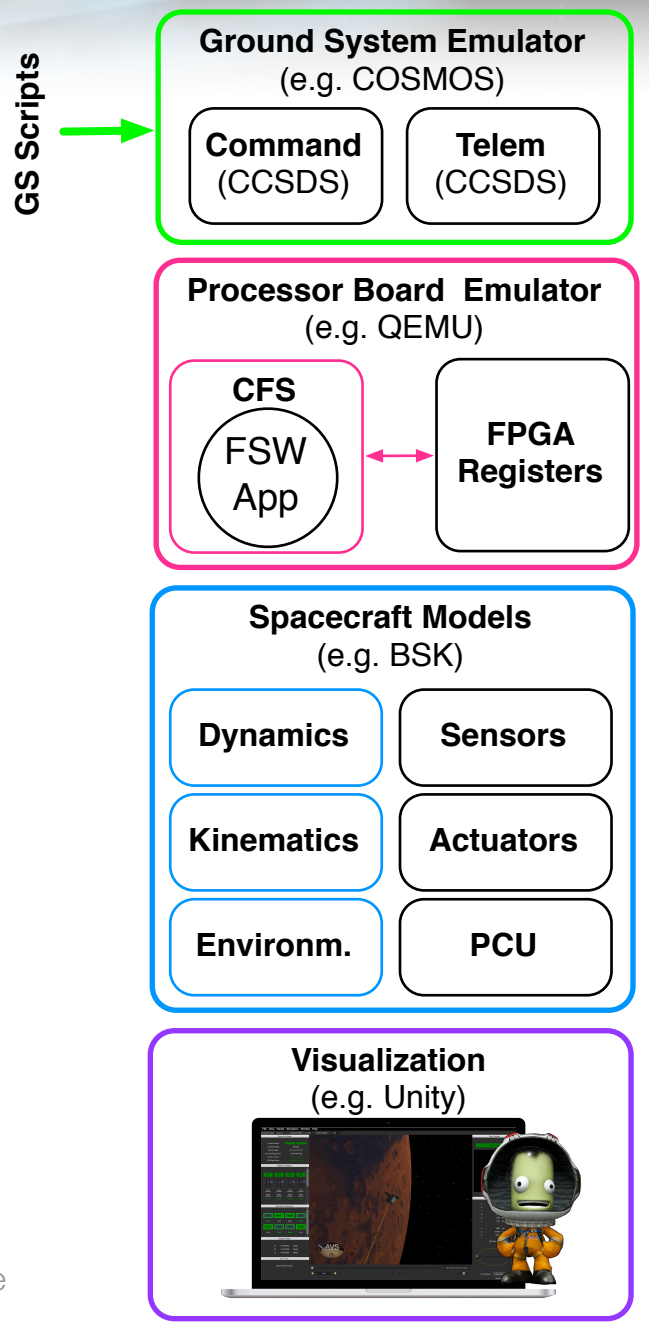
- Expanding simulation coverage… before vs. after:



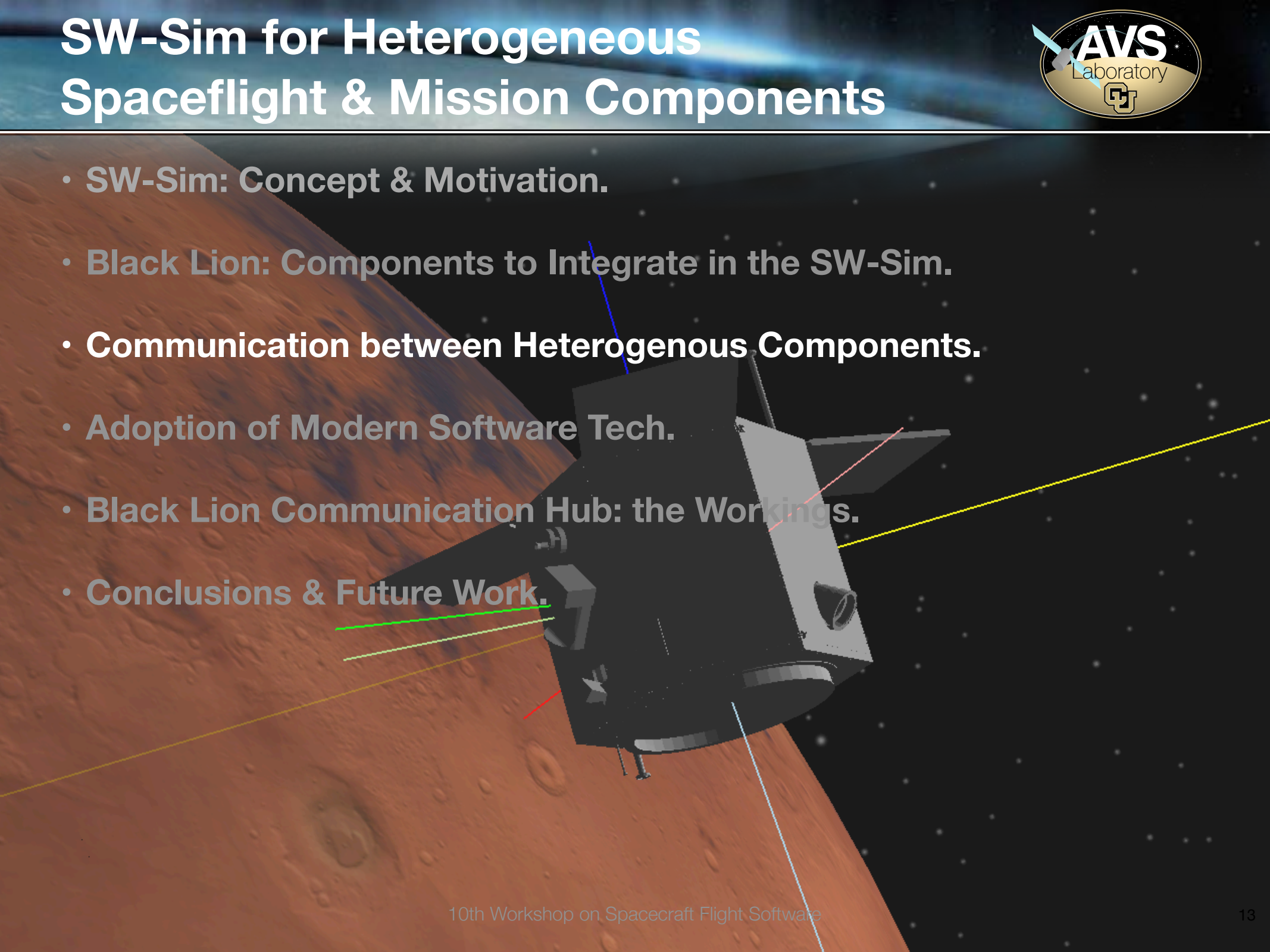
- **Simultaneous, fast testing** from multiple engineering groups: ADCS, FMEA, GS…

Inside Each Black Lion Node

- Operation **Ground System** Emulator.
  - Runs actual **GS scripts**.
  - **Commands** out, **telemetry** in.
  - CCSDS packets.
- **Virtualized model of the SBC**
  - **Unmodified FSW** App executable.
  - FPGA-like **registers**: MM IO for raw data.
- **Spacecraft Models: Basilisk** dynamics engine.
  - **Hardware** components.
  - High-fidelity **DKE** models.
- **Visualization: GUI.**
  - **Unity** game engine.
  - Reproduces **spacecraft physical behaviors**.



SW-Sim for Heterogeneous Spaceflight & Mission Components



- SW-Sim: Concept & Motivation.
- Black Lion: Components to Integrate in the SW-Sim.
- **Communication between Heterogenous Components.**
- Adoption of Modern Software Tech.
- Black Lion Communication Hub: the Workings.
- Conclusions & Future Work.

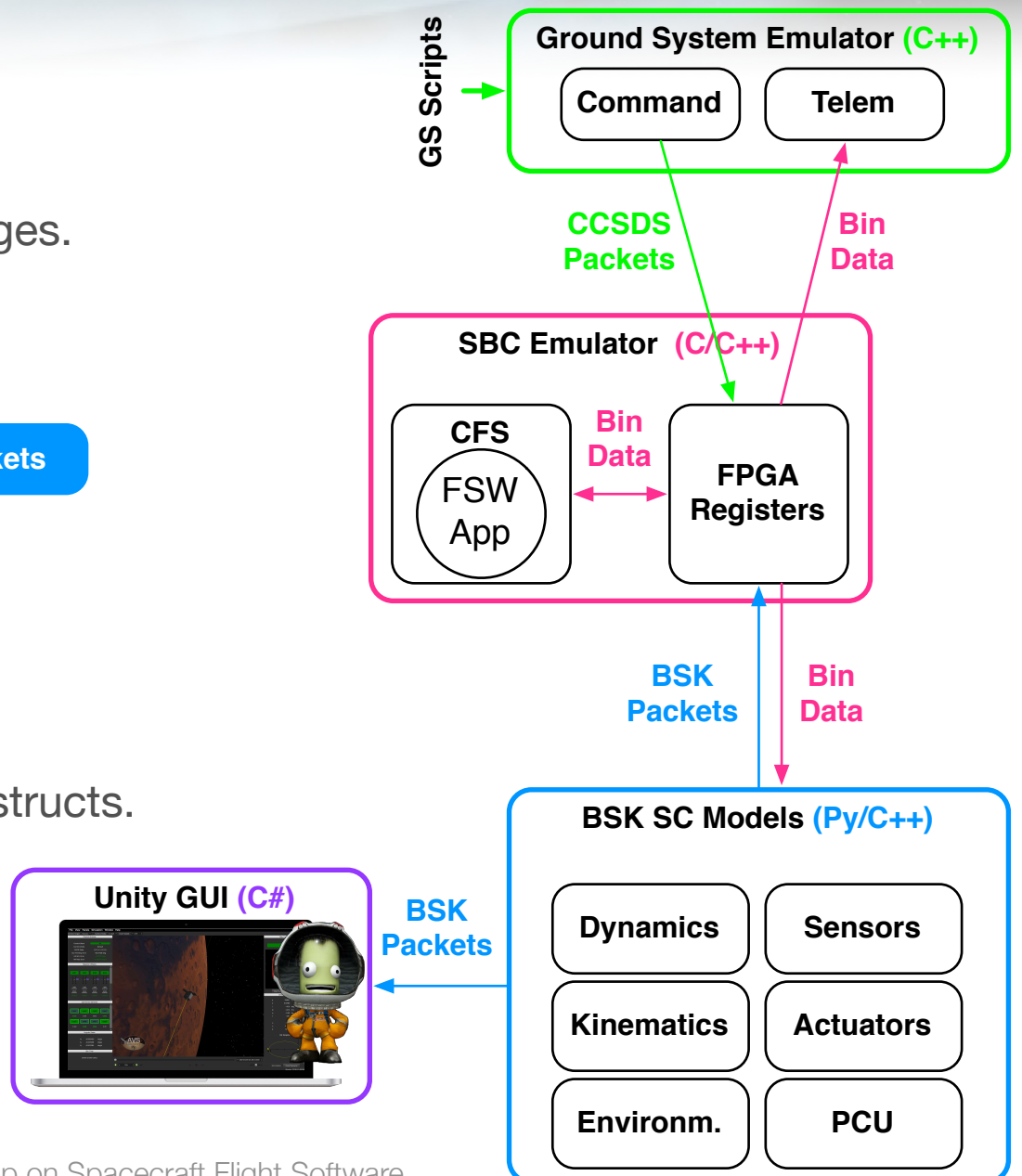
If Connections Were Peer-to-Peer…

- **Node Facts:**
  - Stand-alone programs.
  - Heterogenous programming languages.

    C++ C/C++ Py/C++ C#
  - Heterogenous data packets.

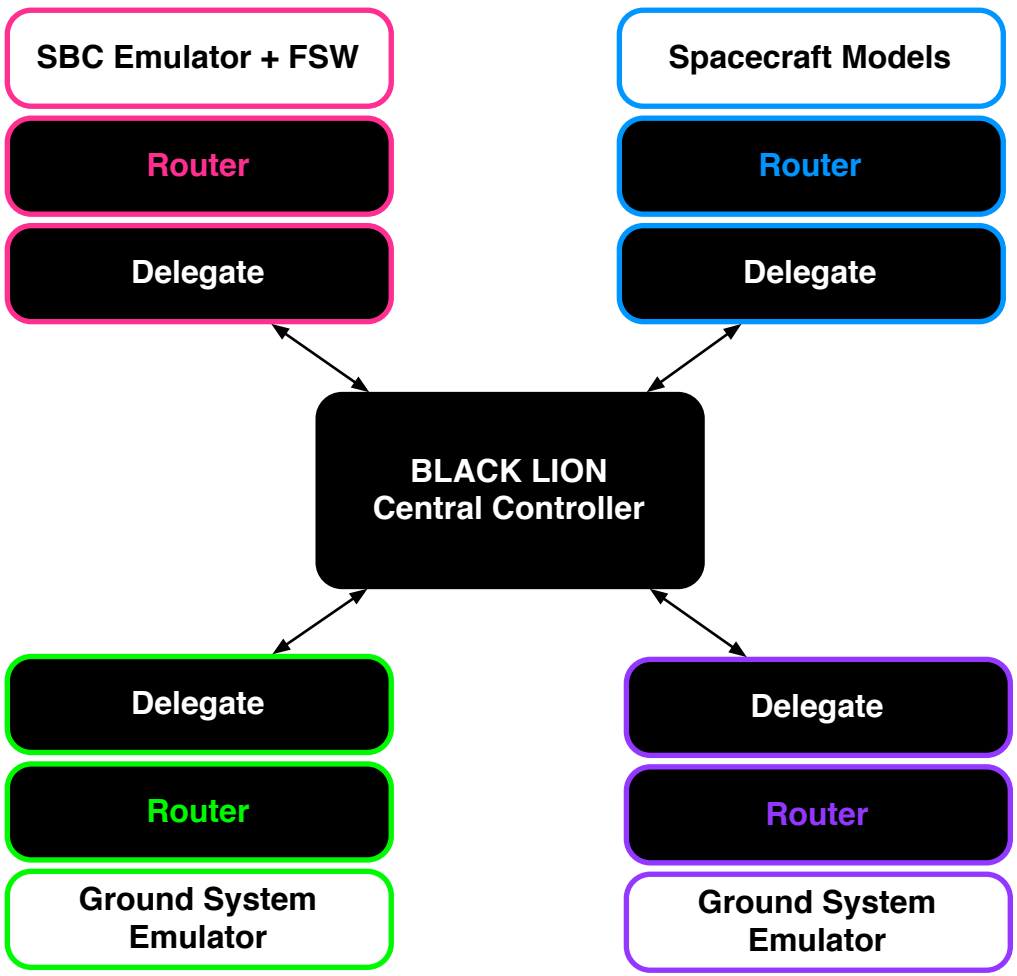
    CCSDS Packets | Raw Bin Data | BSK Packets

- **Communication Goals:**
  - Transport binary data: share bytes.
  - Serialize binary data: from bytes to structs.
  - Synchronize nodes: step forward simultaneously.
  - Dynamic connections map.

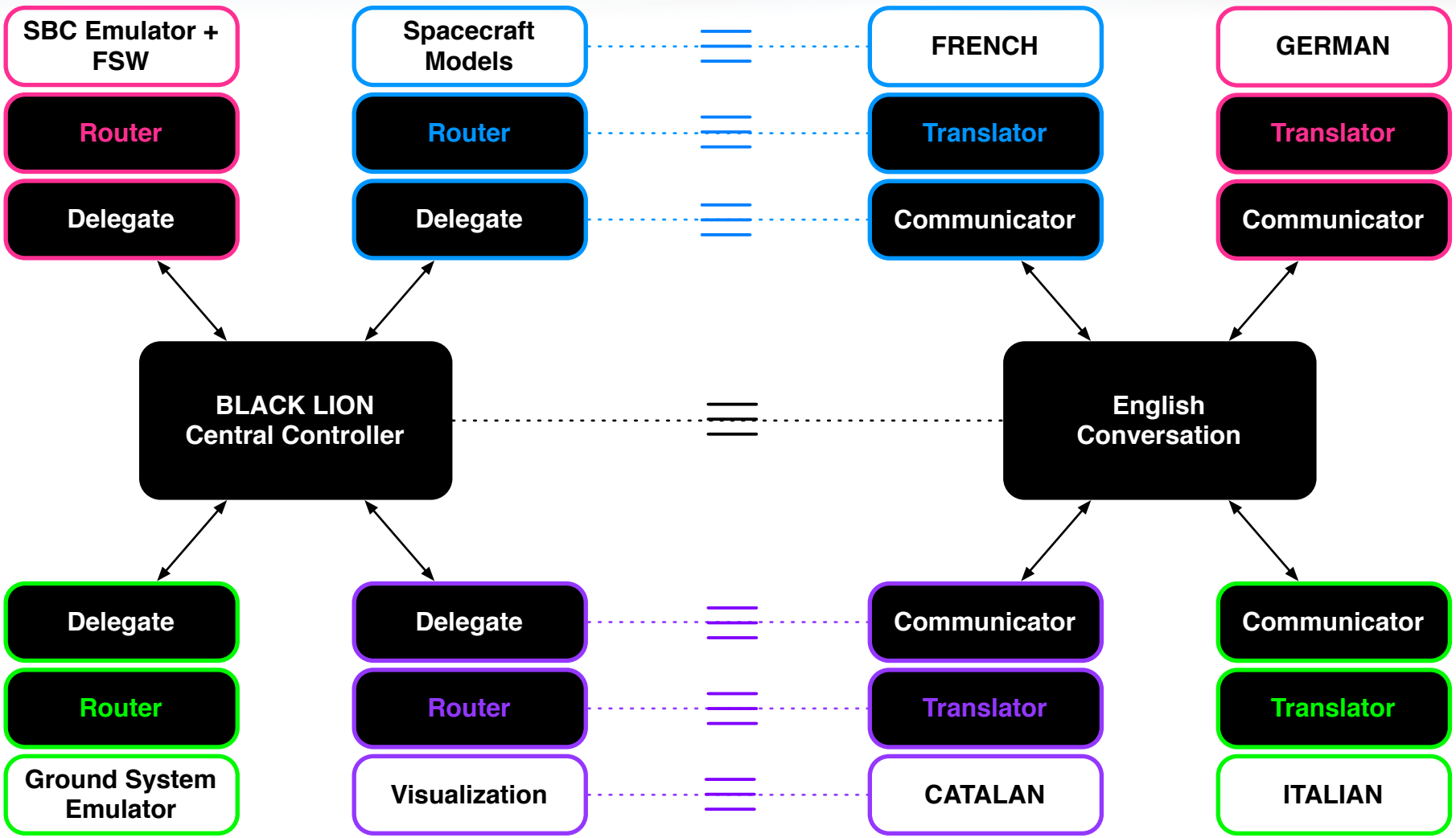


Black Lion (BL) Architecture: Central Controller + 2 Node APIs

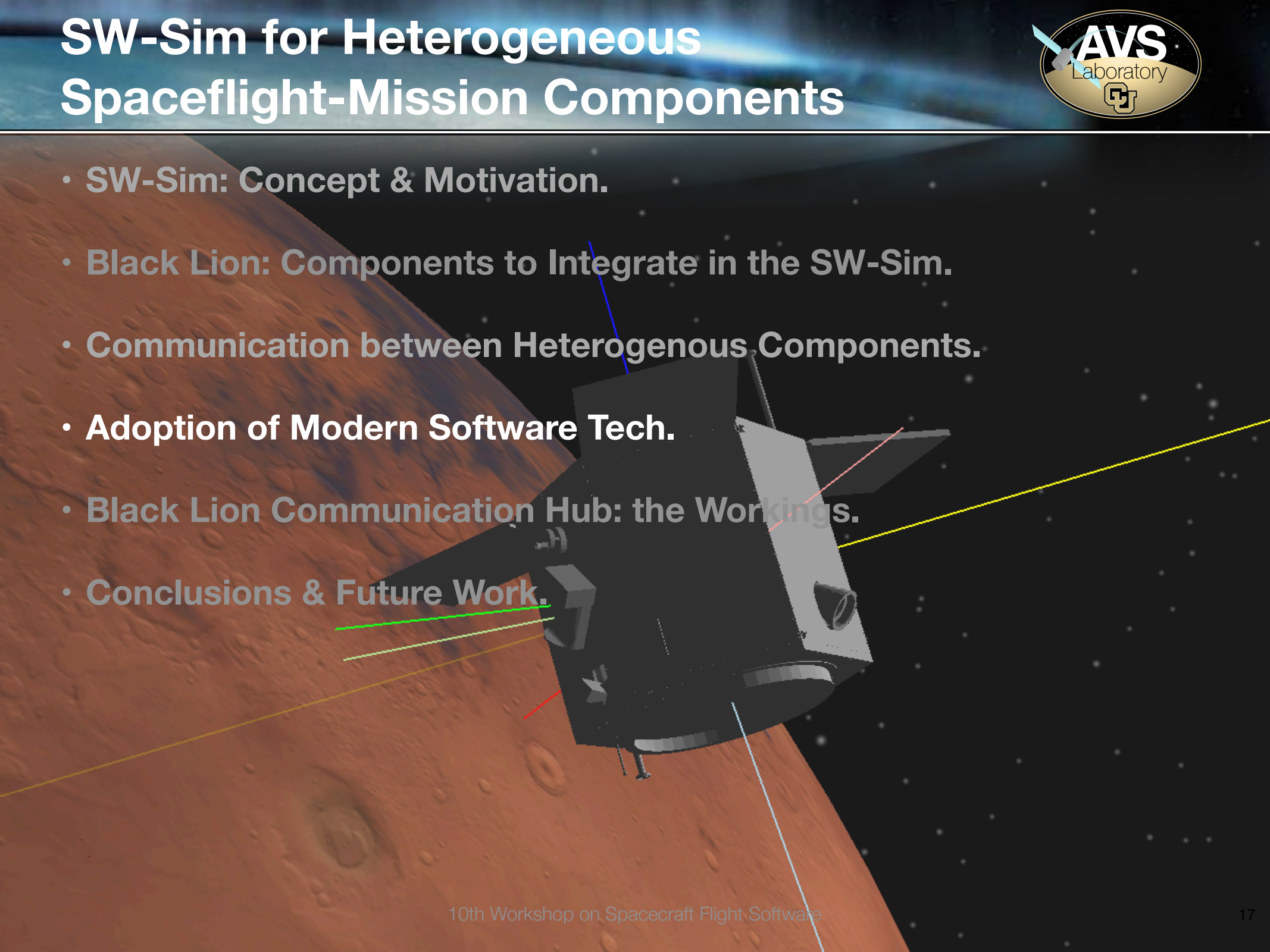
- **BL Central Controller:**
  - One & only static point in the network.
  - Acts as master & broker.
  - Python.
- **Delegate API:**
  - Manages sockets & connections with BL.
  - Same script in all nodes.
  - Python & C++ versions developed.
- **Router API:**
  - Has node-specific callbacks.
  - Route data in & out of the node internals.
  - Gathers node internal data & translates into a common format.
  - Makes data available to the Delegate.



Black Lion (BL) Architecture: Language Analogy



SW-Sim for Heterogeneous Spaceflight-Mission Components

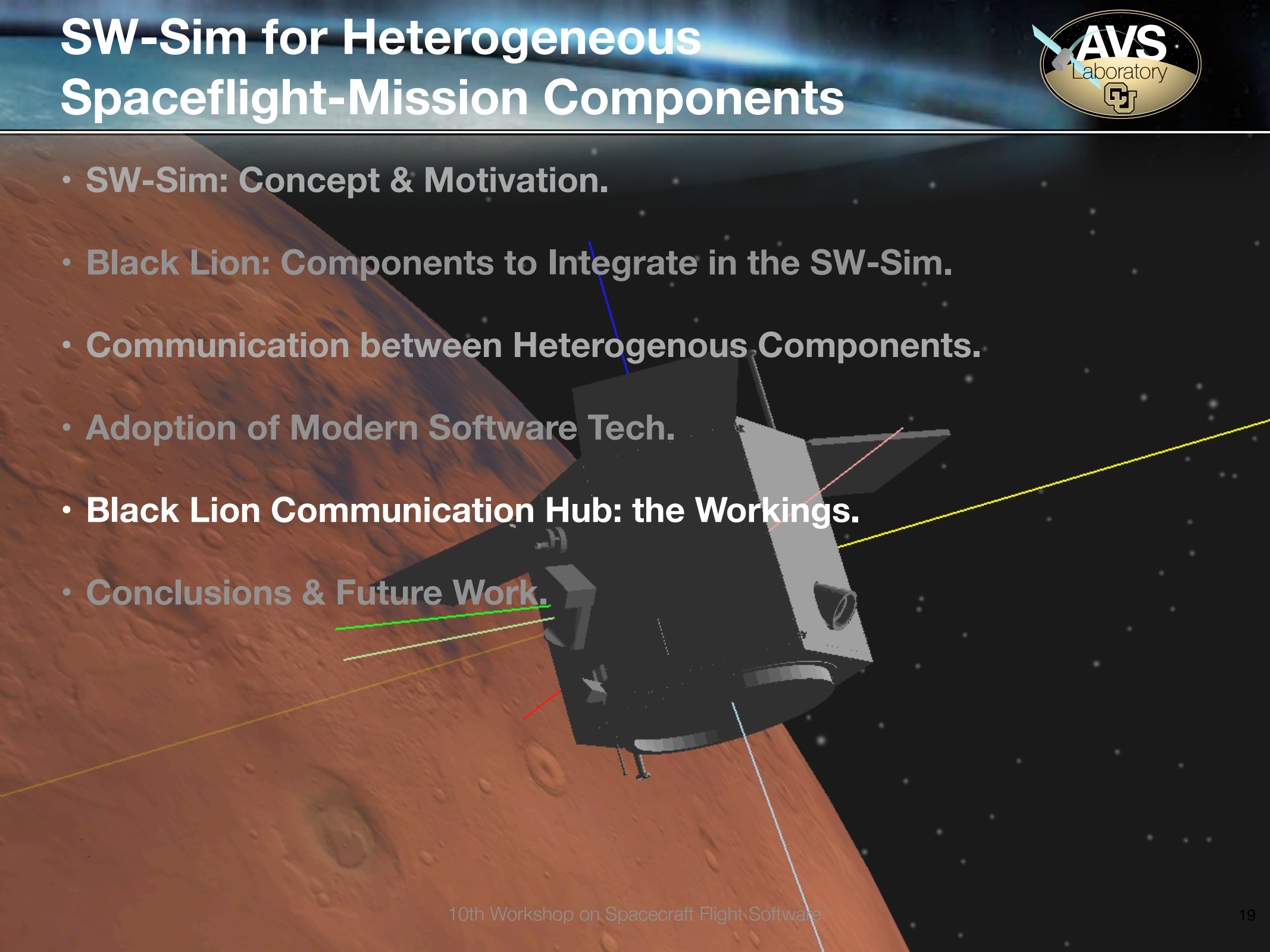


- SW-Sim: Concept & Motivation.
- Black Lion: Components to Integrate in the SW-Sim.
- Communication between Heterogenous Components.
- **Adoption of Modern Software Tech.**
- Black Lion Communication Hub: the Workings.
- Conclusions & Future Work.

Modern SW Technology and Techniques.

- **Technology**: open source, cross-platform, cutting-edge.
  - **ZeroMQ library:** transport data (fast, reliable & protocol independent).
  - **Google Protobuf**: marshall/unmarshall data.
  - **Basilisk** spacecraft models.
  - **Unity** GUI.
- **Techniques**:
  - **Component-based** (modular) **development:** gradual, node by node integration.
  - **Dynamic architecture:**
    - Central controller: one & only static piece (server).
    - Nodes: dynamic clients (can come & go on the fly).
  - **Agile programming:**
    - **Continuous delivery** to mission users from each engineering group.
    - **User feedback** integration.
    - **Fast build-test-deploy cycles.**

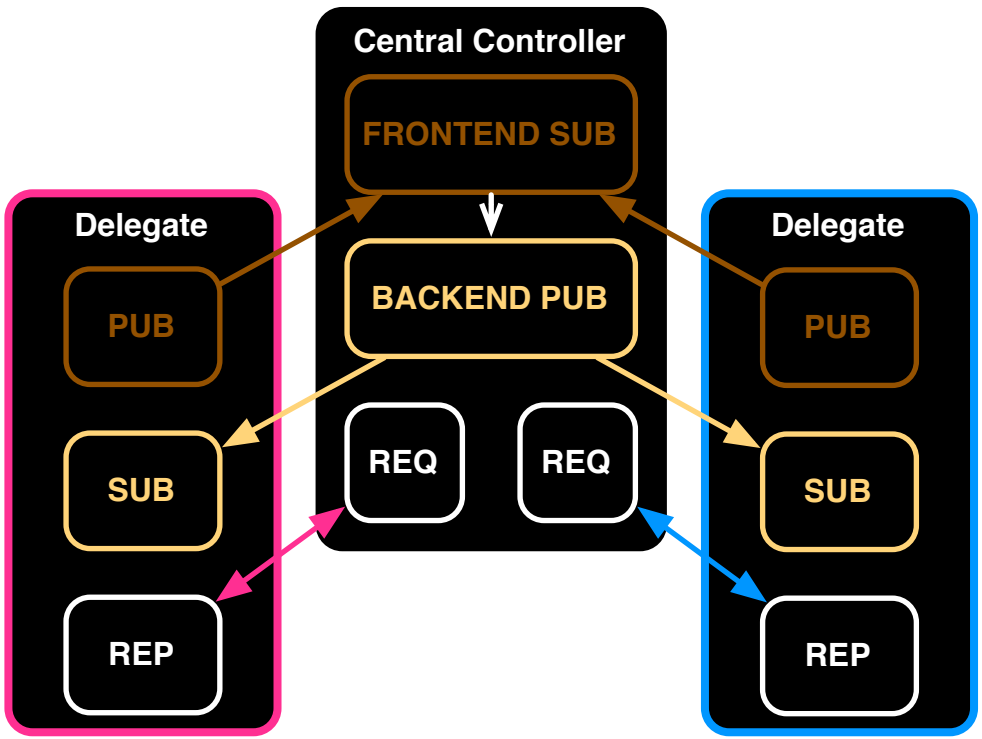
SW-Sim for Heterogeneous Spaceflight-Mission Components



- SW-Sim: Concept & Motivation.
- Black Lion: Components to Integrate in the SW-Sim.
- Communication between Heterogenous Components.
- Adoption of Modern Software Tech.
- **Black Lion Communication Hub: the Workings.**
- Conclusions & Future Work.

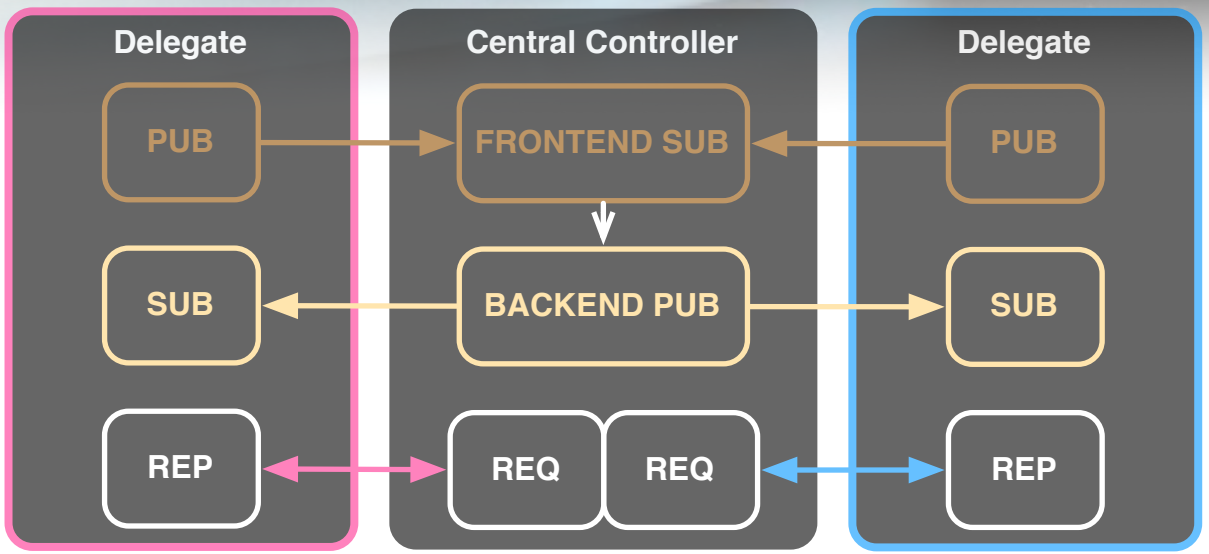
Types of ZeroMQ Sockets

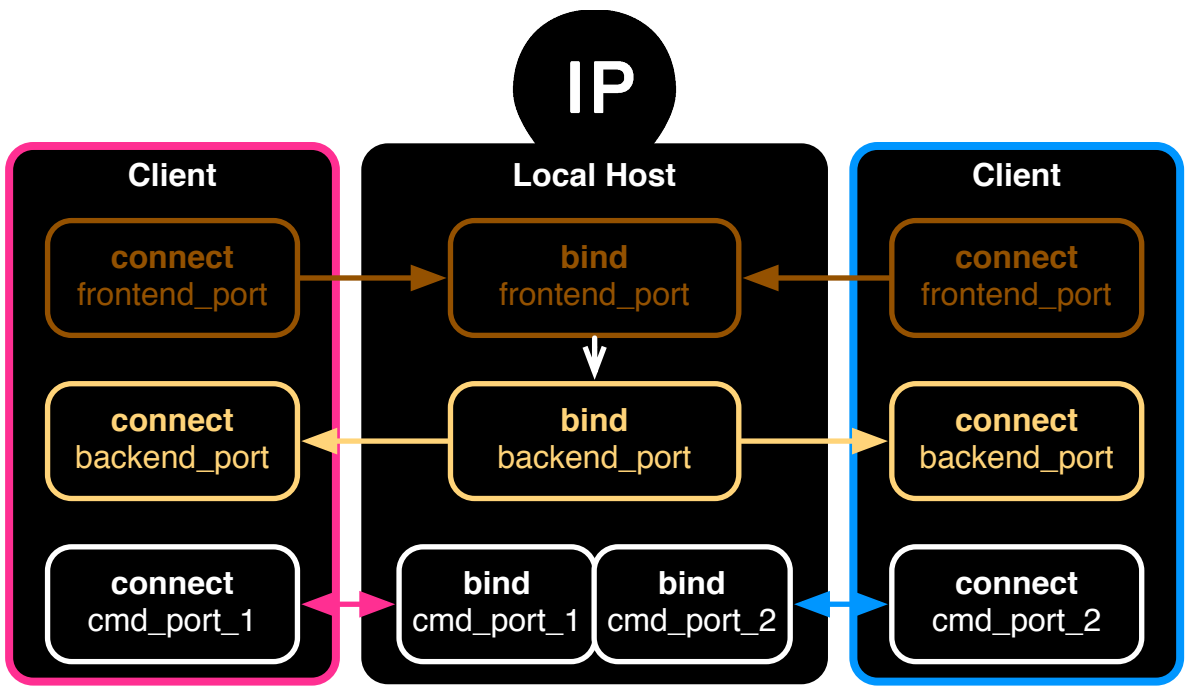
- Three types of ZeroMQ sockets to move data around:
  - **REQ-REP**:
    - **Controller REQ**: make requests.
    - **Node REP**: parse request and reply.
  - **PUB-**Frontend**SUB**:
    - **Node PUB**: publish data.
    - **Controller** Frontend**SUB**: subscribes to publications from all nodes and routes them to the backend.
  - Backend**PUB-SUB**:
    - **Controller** Backend**PUB**: re-publish data.
    - **Node SUB**: subscribes to desired data.



Types of ZeroMQ Connections

- Central Controller:
  - Is a **static "server"**.
  - **Binds** to a static IP address.
  - **2 + N ports**: frontend + backend + N*command.
  - **Protocol independent**: TCP, IPC…
- Delegate (Node API):
  - Node becomes a **dynamic "client"**.
  - **Connects** to the Central Controller.





REQ-REP Relation between the Controller & the Delegate

Black Lion

Req: START (multipart msg)
- This is my **frontend_address**.
- This is my **backend_address**.

Rep: STARTED
- Self-init node.
- Connect PUB to **frontend_address**.
- Connect SUB to **backend_address**.

Req: UNKNOWN_MSG
- Which msg names do you want to subscribe to?

Rep: UNKNOWN
- List of msg names potentially interested in receiving: **sub_list**.

Req: MATCH_MSG (multipart)
- List of msg names other nodes want: **pub_list**. Can you publish any?

Rep: MATCHED
- Look internally for msg name matches: **matched_list**.

Req: TICK (multipart)
- Next frame time-step: **dt**.

Rep: TOCK
- Publish: send out internal data.
- Subscribe: receive external data.
- Step Simulation (**dt**).

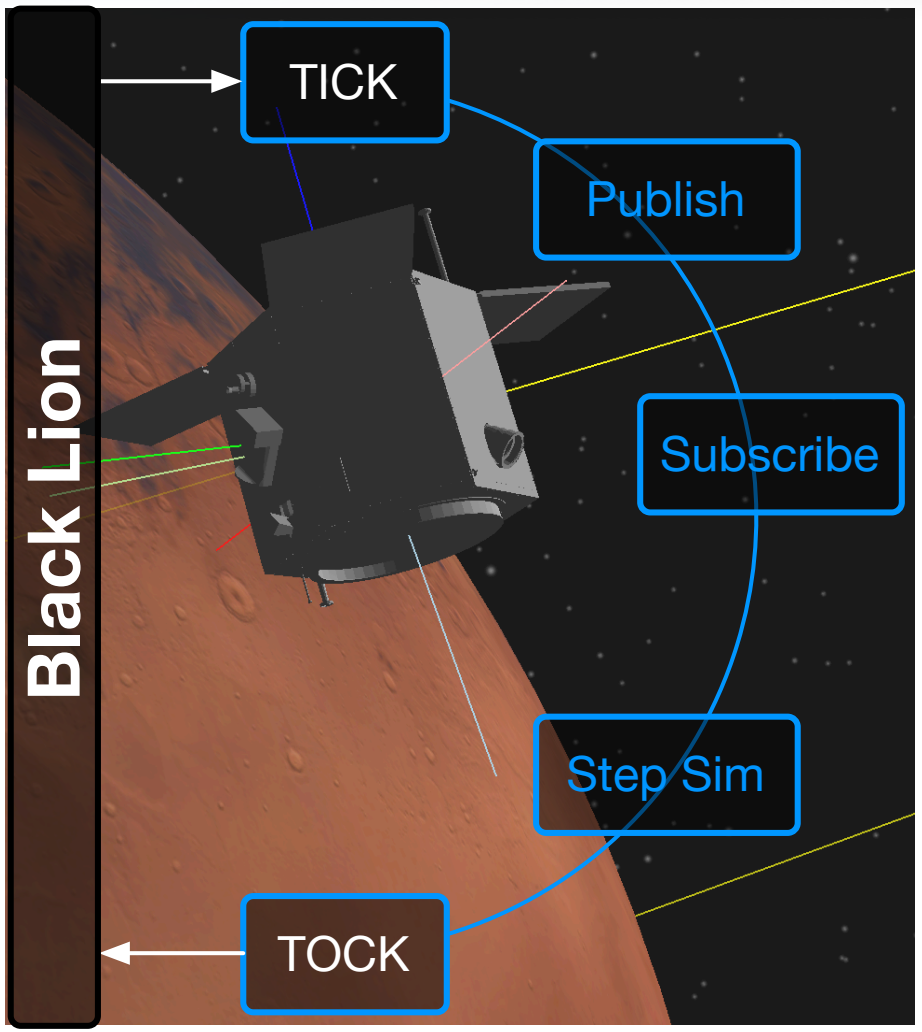
Req: FINISH

Rep: FINISHED
- Close connections and/or quit app.

Node

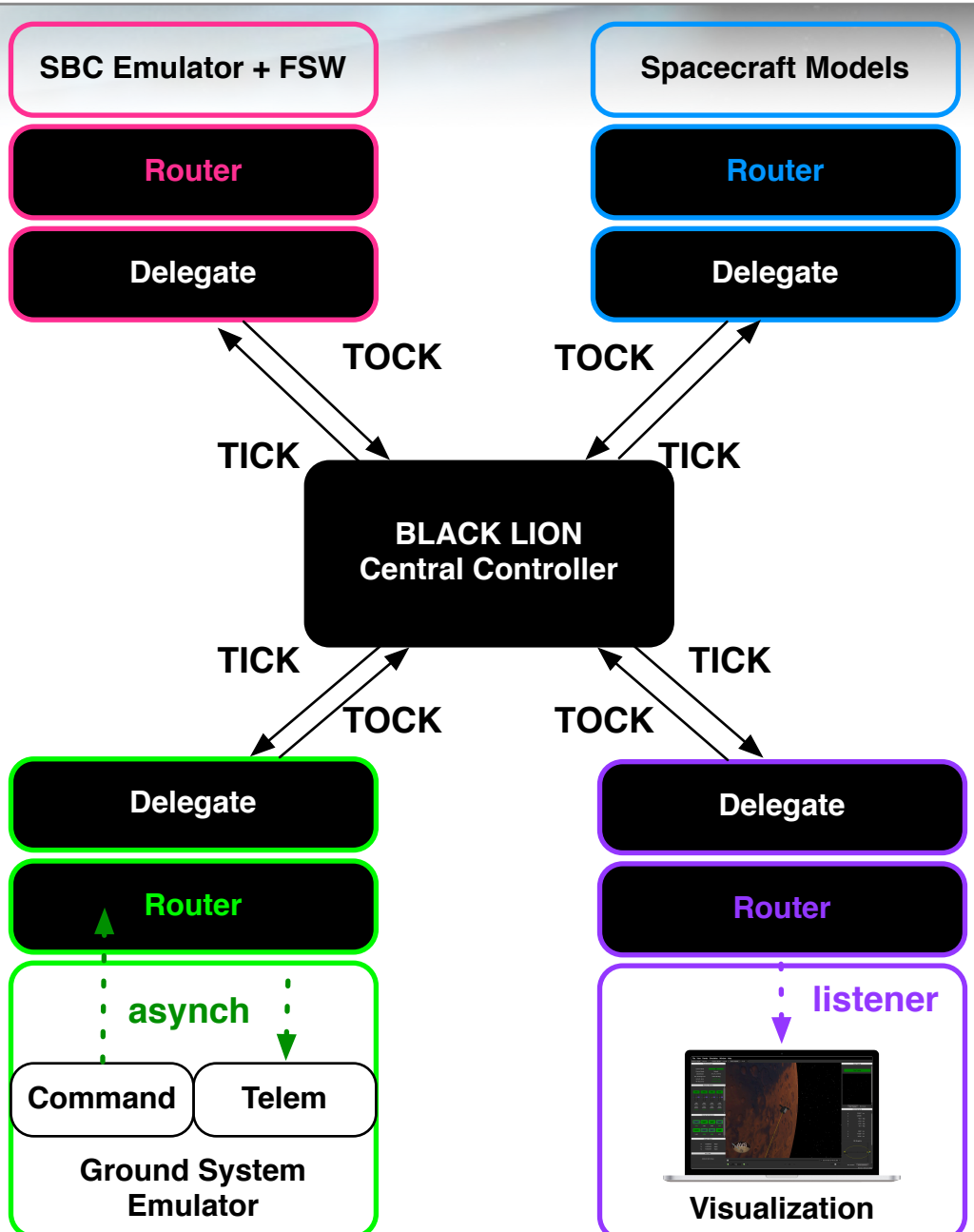
Between TICK Request and TOCK Reply

- **Parse TICK Request.**
- **Publish:**
  - **Router:** Collect internal data.
  - **Delegate:** Send out internal data.
- **Subscribe:**
  - **Delegate:** Receive external data.
  - **Router:** Route in external data.
- **Step Simulation:**
  - **Execute** during **dt** (generate internal data).
- **Send TOCK Reply.**



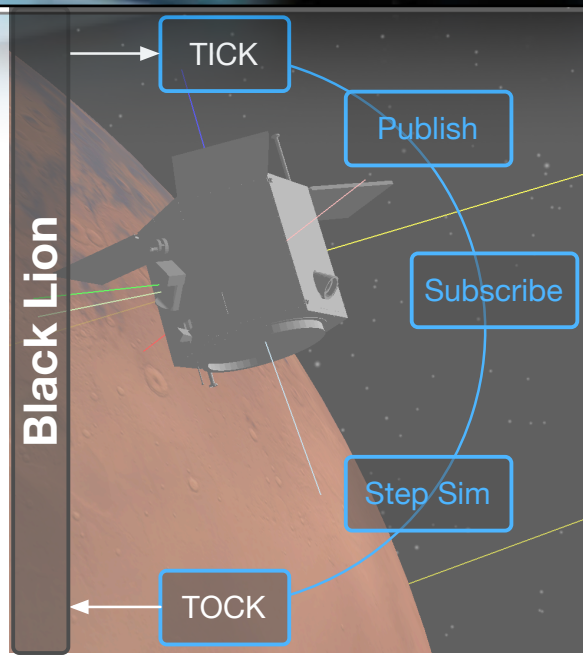
TICK-TOCK: Nodes Synchronization

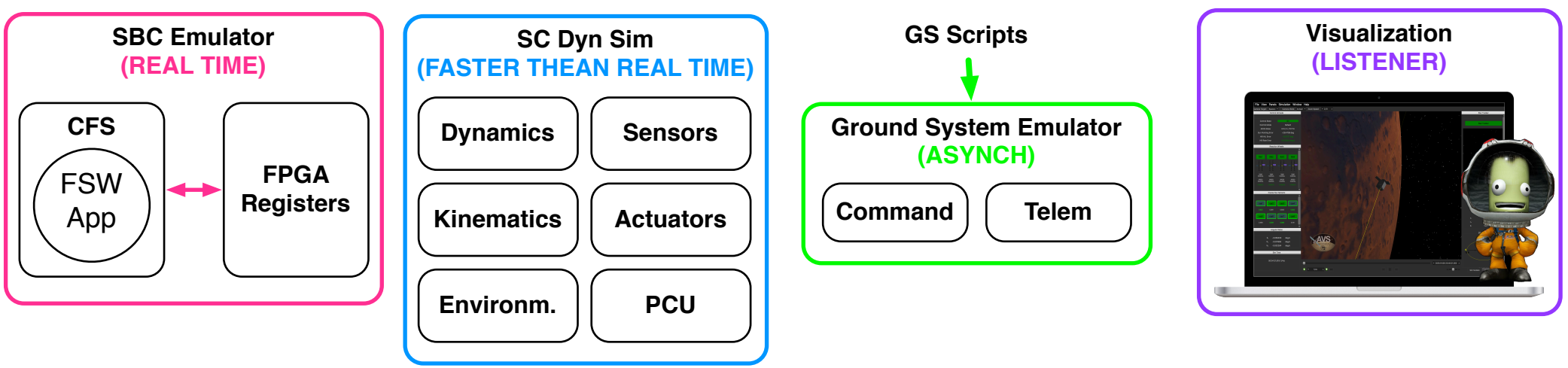
- **FSW**:
  - **Synchronous** nature (cycles or rates)
  - **Real-Time** speed.
- **Spacecraft Dynamics**:
  - **Synchronous** nature (cycles or rates).
  - **Faster than Real Time**.
- **Ground System**:
  - **Asynchronous** nature.
  - Discrete time events.
- **Visualization**:
  - **Listener** only.



TICK-TOCK: Meaning of Step Sim

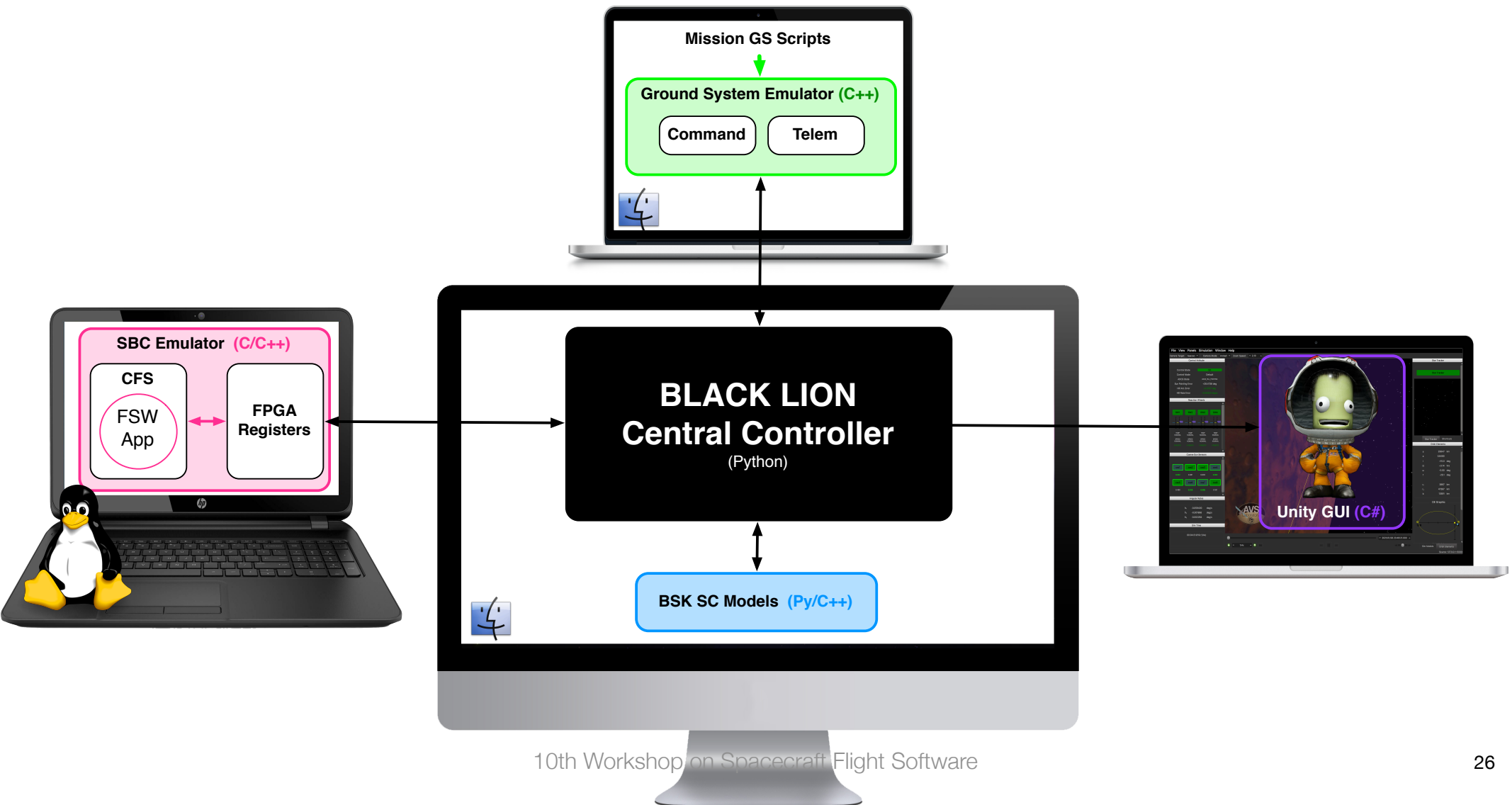
- **Step Sim (dt):**
  - **FSW:** run as many cycles there are within "**dt**".
  - **SC Dyn Sim:** run as many cycles there are within "**dt**".
  - **Ground System:** any command scheduled @ **[t, t+dt]** ?
- Nodes run at **different speeds**, but for the same "**dt**".
- **Once done** with Step Sim**:** send TOCK & **wait** for new request.



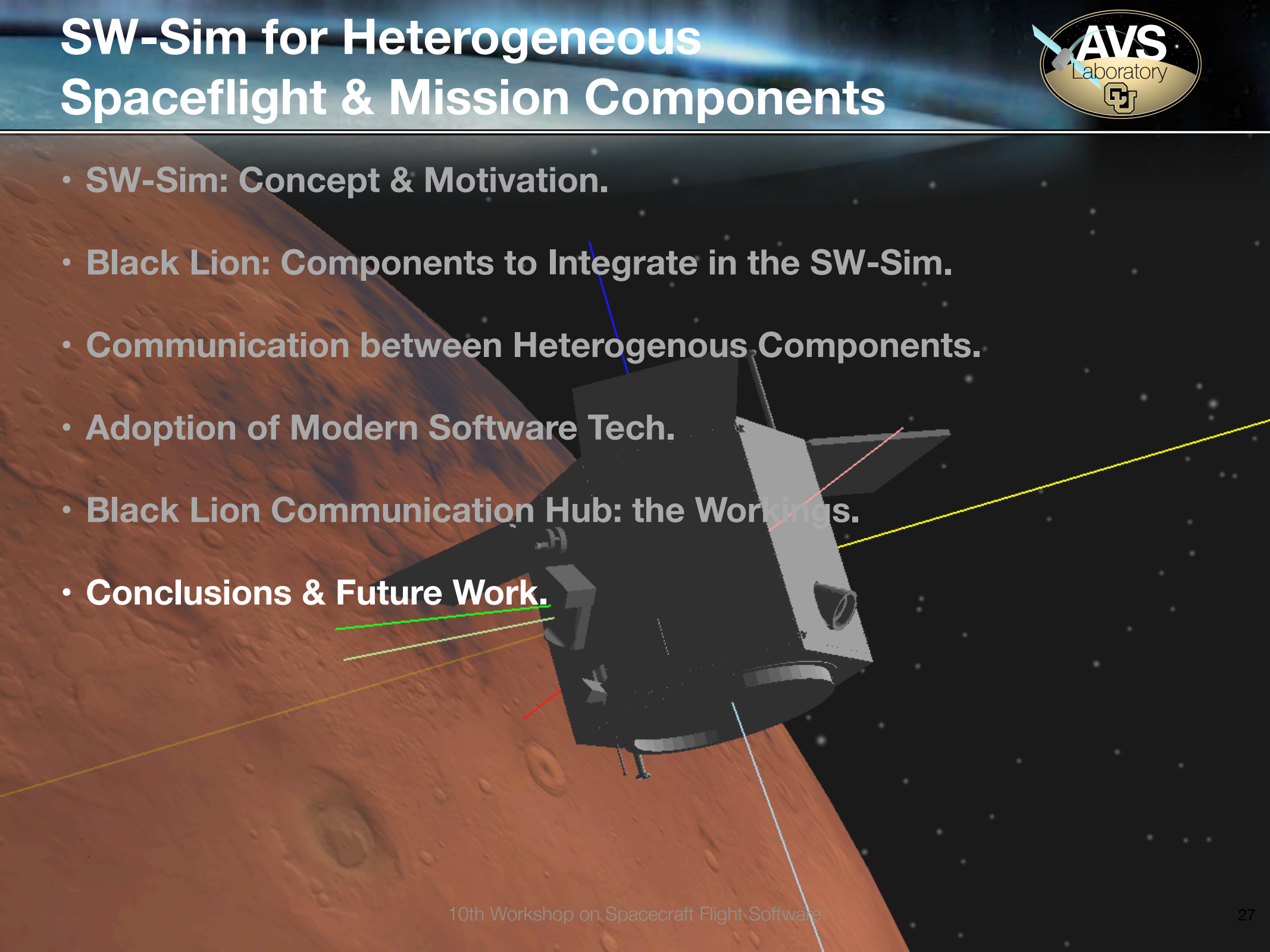


Out-of-the-box User Functionality: Distributed Architecture.

- Capability to run Black Lion in a distributed system.
- Multiple machines running different OS's, all talking to each other.

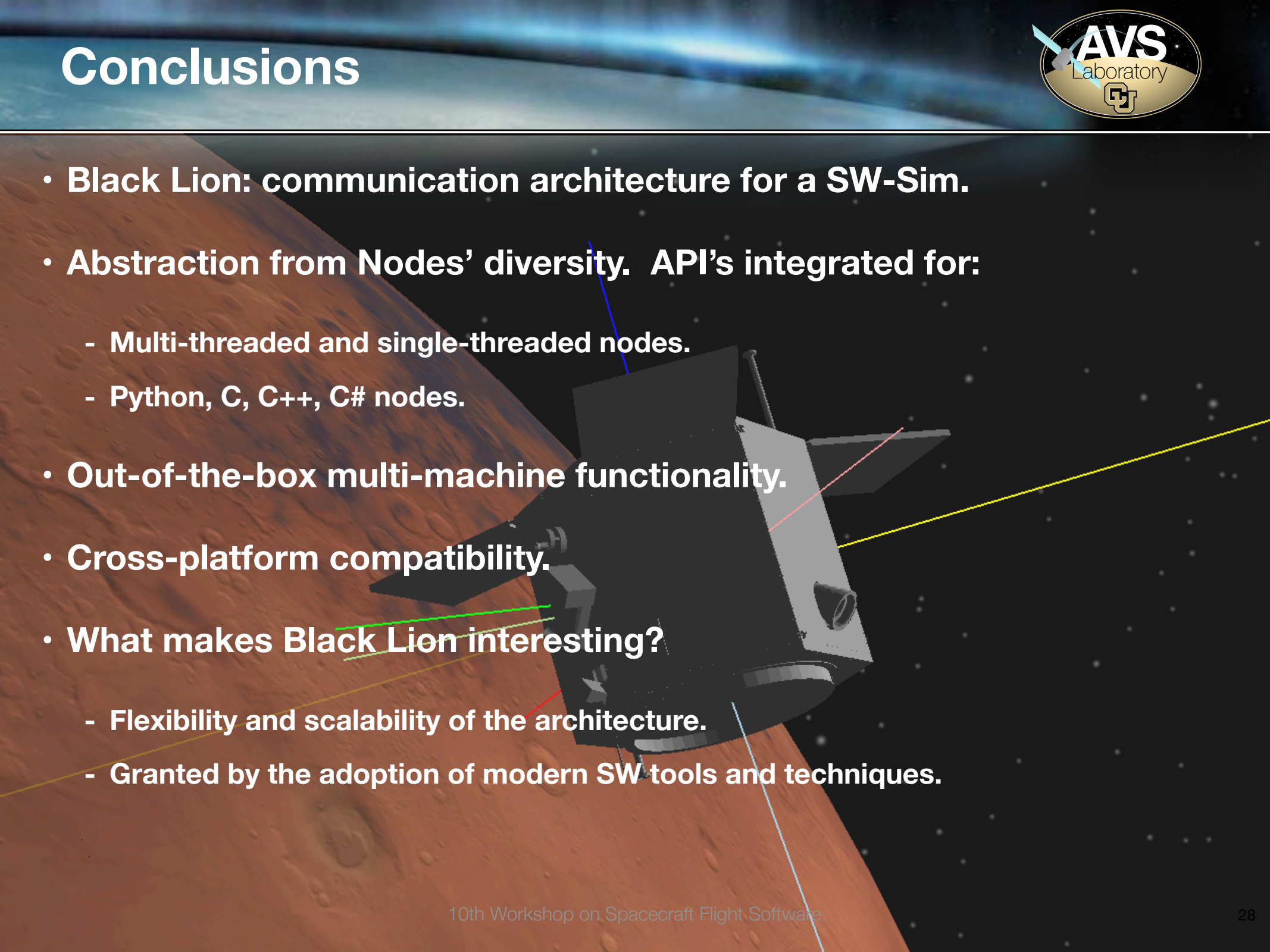


SW-Sim for Heterogeneous Spaceflight & Mission Components



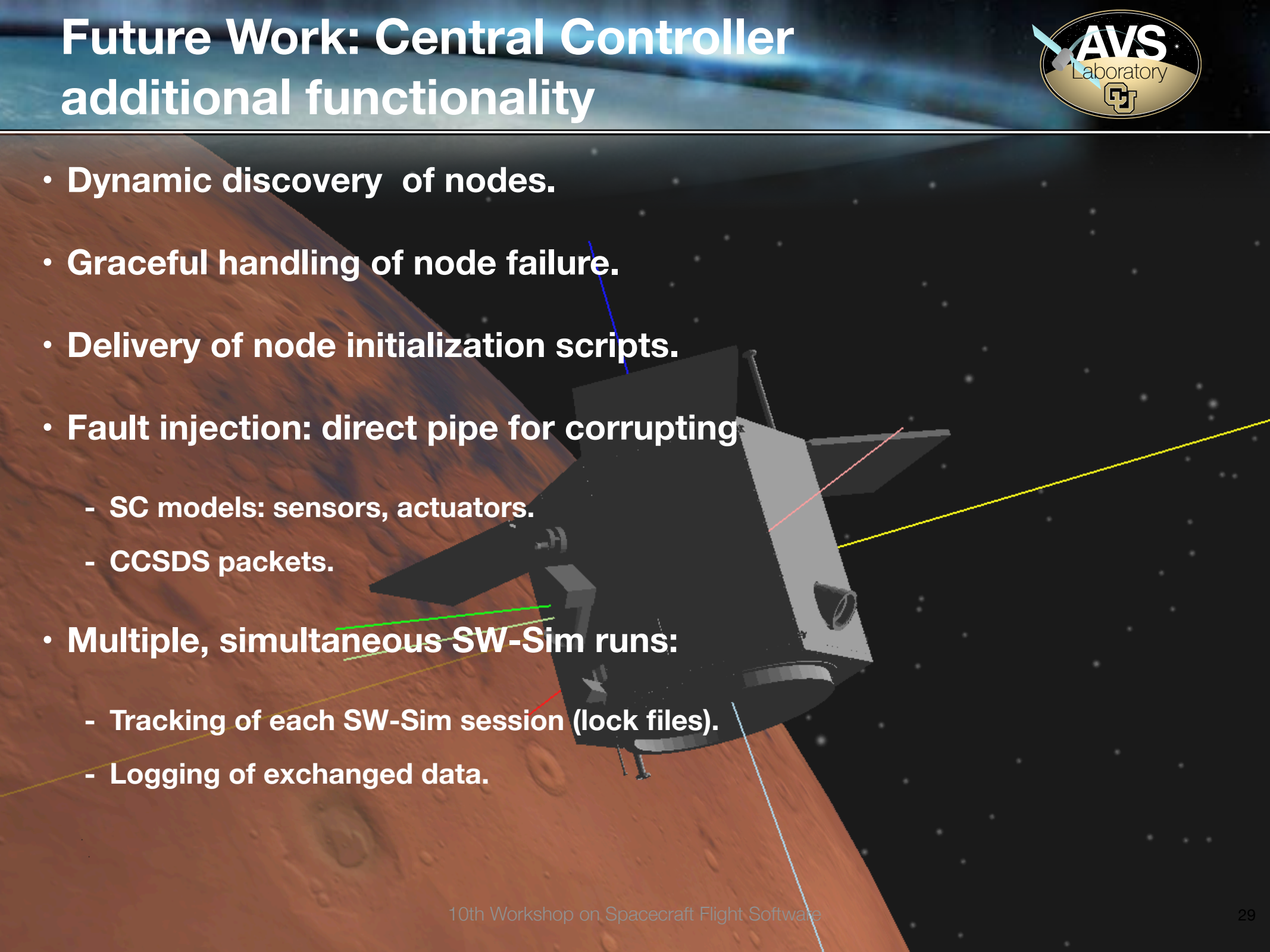
- SW-Sim: Concept & Motivation.
- Black Lion: Components to Integrate in the SW-Sim.
- Communication between Heterogenous Components.
- Adoption of Modern Software Tech.
- Black Lion Communication Hub: the Workings.
- **Conclusions & Future Work.**

Conclusions



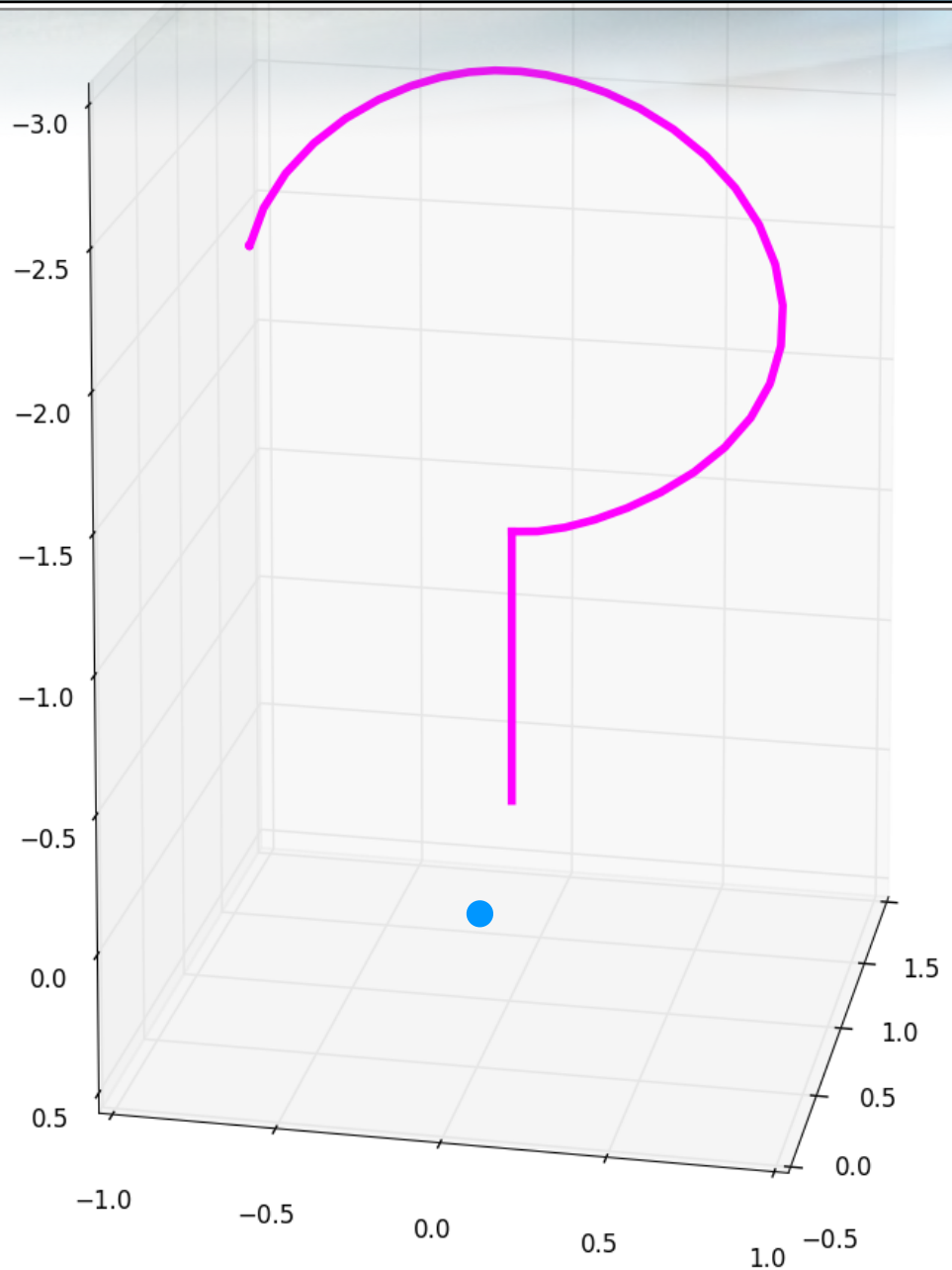
- Black Lion: communication architecture for a SW-Sim.
- Abstraction from Nodes' diversity. API's integrated for:
  - Multi-threaded and single-threaded nodes.
  - Python, C, C++, C# nodes.
- Out-of-the-box multi-machine functionality.
- Cross-platform compatibility.
- What makes Black Lion interesting?
  - Flexibility and scalability of the architecture.
  - Granted by the adoption of modern SW tools and techniques.

Future Work: Central Controller additional functionality



- Dynamic discovery of nodes.
- Graceful handling of node failure.
- Delivery of node initialization scripts.
- Fault injection: direct pipe for corrupting
  - SC models: sensors, actuators.
  - CCSDS packets.
- Multiple, simultaneous SW-Sim runs:
  - Tracking of each SW-Sim session (lock files).
  - Logging of exchanged data.

Questions?



Bore-sight pointing
Spacecraft