Autonomous Systems Coordination: Scenarios and Use Cases

29 July, 2025

Presented by: Dick Stottler, 650-931-2714
stottler@StottlerHenke.com

Motivation

- High costs to get equipment on lunar service -> reuse capabilities (i.e. solar panels at South Pole) and cooperate instead of every platform can do everything alone
- Moon is stepping stone to Mars (very high latency, autonomous cooperation)
- Much research and development has been done in the development of individual autonomous systems (AS)
- Minimal development of the information exchange standards
- NASA's plans for a permanent lunar base envision multiple autonomous systems working in coordination to perform complex coordinated tasks such as robotic assembly, mining and crew rescue/support
- Multiple AS will need to share plans, schedules, procedures and FDIR
- Heterogenous set of AS will come from different countries and different companies
- Need interoperability standards for communication between autonomous systems

Overview

Motivation

Relevant Stottler Henke Background

Overview

Levels of Interactivity, in roughly increasing order of complexity and difficulty

Example scenarios, use cases, and information exchange

- Level 1: Deconfliction
- Level 2: Cooperating Entities
- Level 3: Aid in Recovery

Lessons Learned / Pitfalls

Stottler Henke Background

Since the 1980s: 100s of projects involving information exchange between Intelligent Systems

Each time, did precisely what this initiative seeks to avoid:

- Definition of individual pairwise information exchange ”standards”

By recounting and generalizing our experiences, hopefully this helps provide the Interoperable Autonomous Systems Workshop some input to develop interoperability standards for communication between autonomous systems

Interactivity Levels

Level 0: Two entities do not interact and neither they nor their effects are ever in the same place or domain (e.g. RF, sunlight shading) at the same time. No interaction or exchange of information needed.
Level 1: Multiple entities do not intend to interact or cooperate in any way, but they, or their effects may be in the same place or domain at the same time. They need to exchange enough information to deconflict their presence/actions/effects.
Level 2: Multiple entities intend to cooperate to jointly accomplish an objective; Common Types: X-as-a-service, Independent Entities Synchronizing, Joint Real-Time Control

Level 3: Aid each other in Recovery/Repair (external/internal)

Considerations for Information

Physical versus Cognitive Interactions

Human(s)-Human(s) and Human(s)-Machine(s) interactions as models for Machine(s)-Machine(s) interactions

- At Stottler Henke, we take AI to often mean the mimicking of human thought processes.
  - We have gotten a LOT of mileage from this

Shared Understanding/Background (models, playbooks, etc.)

Different Levels of Replanning/Rescheduling Sophistication

Important to think about deviations from normal

Level 1: Deconfliction

Avoid collisions either physically or in the EM spectrum

Effects they have on their shared environment

- Do not destroy (break/undo) the work of another
- Do not make some change that prevents one from doing what they need to do. (E.g., don't dig a trench on ground that another robot is planning to traverse over.)

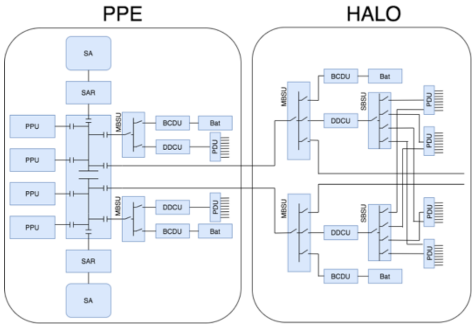
Shared resources have to be deconflicted (Physical-Space, CommRelay/Antenna, Power, RF, Sunlight, X-as-a-Service, etc.)

Unintended Interactions! Be careful! Sketch a proof

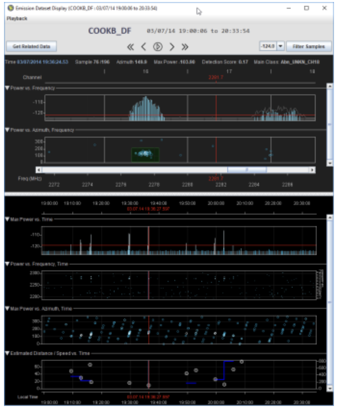
Level 1 Stottler Henke Project Examples

Satellite Control Network
- Scheduling (deconflict ground antennas)

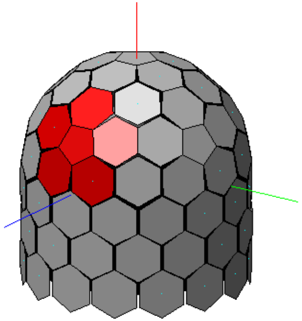
Spacecraft EPS
- Scheduling/Fault-Diagnosis/Recovery/Replanning



RF Interference (sensing / comms can conflict)



Level 1 Stottler Henke Project Examples



Beam transversal over radiating elements of phased array antenna

Aggressive Aircraft Maneuvering

Planning Military Unit Movements

Scheduling/sequencing Space Shuttle (and SLS) Processing (and aerospace manufacturing): Prevent one entity from destroying another's work (Prep work, e.g. long powering up and down sequences)

Level 1 Required Information Exchange

Avoid collisions: Special Case of shared resources

Effects they have on their shared environment

- Can be cast as effects on shared/reserved resources

Shared resources have to be deconflicted

- Physical-Space: Size(bounding Cylinder?), Location(time)
- RF: BeamWidth, Frequency/Power/2-D-Angle(time)
- Power: ConnectionID, PowerDraw(time)
- Sunlight: Sun-Blocking Profile, Location(time)
- X-as-a-Service: LocationID, X(time)

Effects: Reserve Resource (t1, t2) where $-\infty \leq t \leq +\infty$

- E.G. Reserve Physical-Space$_k$(5, $+\infty$), Where Physical-Space$_k$ is a defined volume

Level 2: Cooperating Entities

2.1 X as a service; X == Power (e.g. Astrobotic's VOLT Rover), Comms, PNT, mobility (one rover caries another robot from one place to another), Recovery/Repair (Lunar AAA!)
2.2 Multiple relatively **independent** entities jointly perform tasks to achieve a common goal. They primarily need to synchronize their activities logically to achieve their objectives. One robot creates the floor before another robot installs the walls (which must already have been created or unloaded by another robot) before a third puts on the roof (which must already have been created or unloaded by another robot).
2.3 Multiple robots are intimately involved in the same tasks, especially in a way that requires joint real-time control. E.g. two robots are carrying a heavy slab by each holding up one end as they transit together. Or a pair of robots hold two parts together while a third welds them.

Level 2.1: Cooperating Entities: X as a service

Advantages:

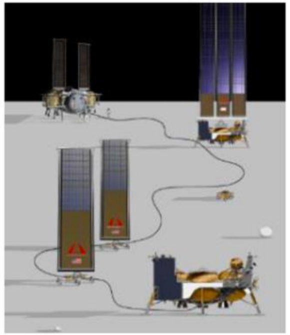
- Simplifies interface and interactions (considerably)
- Facilitates Multiple Organizations: simply have to conform to physical and software interfaces

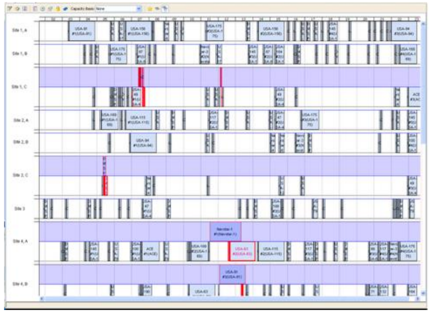
Disadvantages:

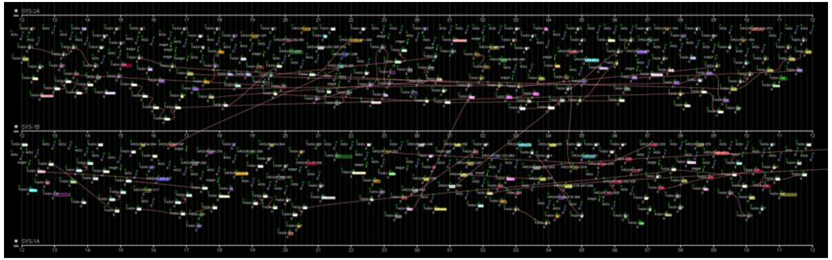
- Having to conform to the interface
- Leaving capabilities “on the table” (crane example)
  - Possibly mitigated by defining ability to circumvent service API (which may decrease some of the advantages)
- Cascade of failures (Many robots depending on a failed service)

Level 2.1 X as a Service Examples

- Astrobotic, Electrical Power as a Service
- SCN/Phased Array Antennas (different perspective from above)
- Satellite-Based Sensors
- ISS (Cooling, Power)
- Spacecraft bus (Power, Thermal Management, CDH, etc.)







Level 2.1 X as a Service Required Information Exchange

Service Publishes its availability(time)

Clients request service (Requirement (amount(time)), Time Window, Other Constraints)

Service confirms or denies/suggests

- Note: Denial can not just be a simple “No”, because what next?

Service updates availability(time) based on assignments and status changes (e.g. partial failures or degradation)

Level 2.2: Cooperating Entities

Physically Independent entities perform tasks to achieve a common goal

Need to synchronize their activities logically

- E.g. Finish-to-Start, Start-to-Start Temporal Links, Shared Resources

More flexible and more complicated than X-as-a-Service

Requires Planning and Scheduling:

- Centralized (Master-Slave), Hierarchical (with or without feedback), Peer-to-Peer
- Negotiation Cycles

Level 2.2: Cooperating Entities Examples

LunaNet Comm Relay Satellites: DREAMS (peer to peer)

Smart Coordination Of Unmanned Teams (SCOUT) Projects:

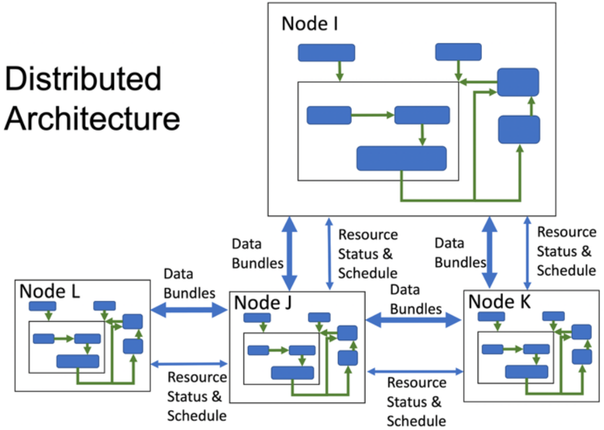
- For very constrained communication environments (no or limited bandwidth)
- Autonomously determine how to meet objectives, divvy up tasks, monitor/repair COA
- Search and X
- Sensor Array
- DoD EW Mission
- Mostly Hierarchical (some with feedback) planning
- Plays (like football(with roles))
- When Playbook is preloaded, per mission, provides tasks, task assignments, coordination, and information with very few bytes.
- For each entity, just reference the Play # and its role. Now it knows its tasks and everyone else's and can assume, when out of contact, they are performing (useful for replanning)

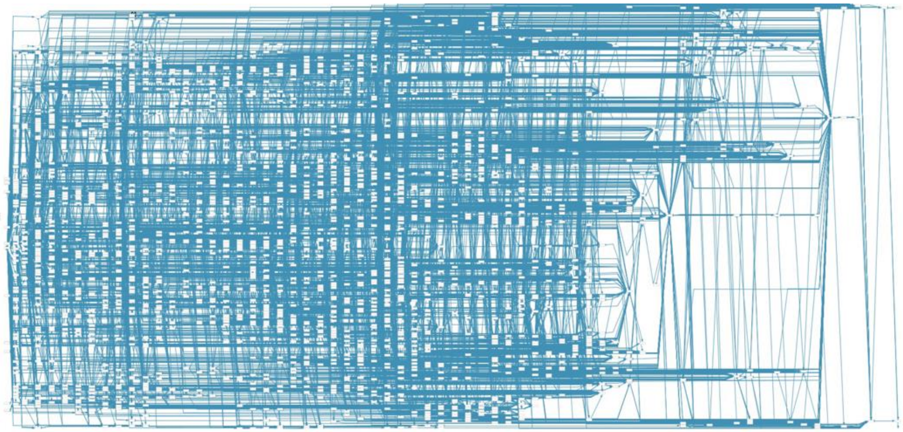
Space Shuttle Processing Scheduling

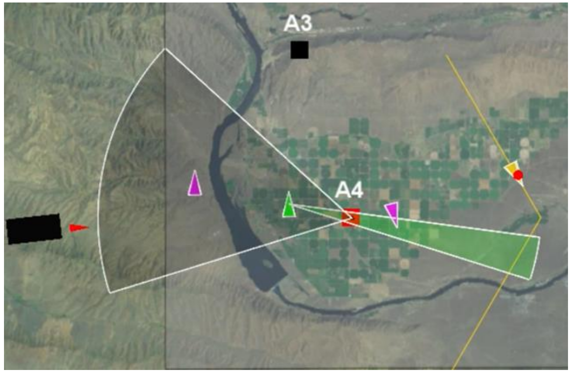
Boeing 787 Dreamliner Assembly

Level 2.2: Cooperating Entities Examples

Distributed Architecture







Level 2.2: Cooperating Entities, Required Information Exchange

Eventually: Each entity must have its own list of tasks and tasks assigned to other entities, when some relationships exists between those tasks or entities and its own tasks or self.

Its-task -FS-> Own-task Or   Own-task –FS-> Its-task

Its-task uses a resource that Own task does, so must wait for it

Potential-conflict/deconfliction (See Level 1)

Entity/Task-List will get updated over time

Process/sophistication of initially developing tasks (COA) and assigning them will impact information flow AND CONTENT, potentially

- E.g. can an entity replan/reschedule for the unexpected, needs info to do so

Level 2.2: Cooperating Entities, Required Information Exchange

Task Lists require planning (generate tasks) and scheduling (assign tasks)

Enough Information exchanged to plan and schedule

- Who is planning and scheduling and with what kind of process:
  - Centralized (Master-Slave), Hierarchical (with or without feedback), Peer-to-Peer

  Planning Model (several different types, e.g. ANML, Hierarchical Task Decomposition, etc.)

  Scheduling Model (tasks (time, resources needed, constraints), resources)

Procedure Execution Languages for Representing Tasks?

- E.G. PLEXIL, PRIDE

Level 2.3: Closely Coordinating Entities

Two robots are intimately involved in the same tasks
Especially in a way that requires joint real-time control
E.g. two robots are carrying a heavy slab by each holding up one end as they transit together.
Issue: “Cooperating” can be “Conflicting”
Example: two independent Arms holding same object in place.
Consider 1 dimension: If both have closed loop control on the position, small disagreements or noise in the two sensors will cause a “tug-of-war”
If both have closed loop control on force, small disagreements or noise in the two sensors will cause drift
Solution is, for any one dimension, one controls for position and one controls for force. I.e. one “leads” and one “follows”, like couples dancing

Level 2: Closely Coordinating Entities, Examples

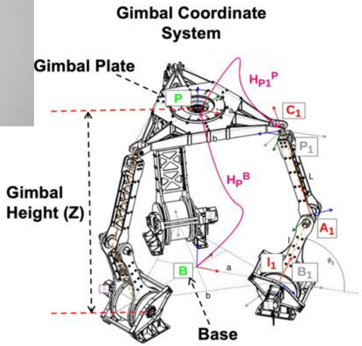
Jointly carrying/placing an object (many construction tasks are “two-person jobs”)

Formation Flying (ACMan Project): One leader and the rest are followers

Cooperative sensing (cross-fixing or synthetic aperture) CADANCE project

Astrobotic VOLT’s 3-armed Gimbal

Stanford 3-Fingered Hand



Level 2: Closely Coordinating Entities, Information Exchange

More detailed

Task dependent

Action (subtask) list

Time-synchronization, action-begin-times

Real-time controller division of labor, who follows, who leads which dimensions,

- Entity A will lead movement in all 6 dimensions, Entity B will follow movements, maintaining K push force in direction of Travel …
- Real-time action changes/coordination (e.g. tip the couch up to go around a corner, which end goes up)

Level 3: Mission Recovery or Repair

External Aid:

- Rover is stuck and needs a push, pull, or lift
- Rover has tipped over and needs to be right-sided.
- Battery has runout (perhaps in a shaded region) and needs to be recharged.
- Rover is immobilized by a lack of comms and needs a comm relay.
- Deployable component is stuck and needs a “jiggle” or a pull or push.

Internal Repair Some internal component

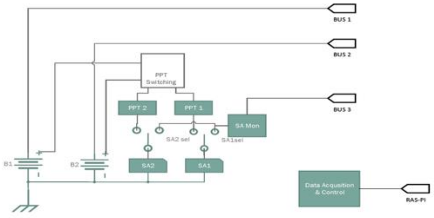
- LRU needs to be replaced or removed/repaired/reinstalled
- Component needs to be repaired in place

Level 3: Mission Recovery or Repair Examples

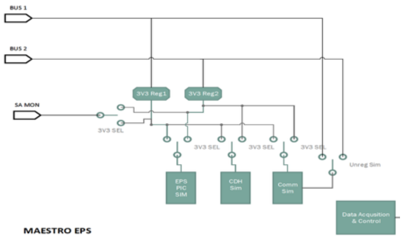
MAESTRO projects: Automatic Fault Detection, Diagnosis, Recovery, Replanning, and Rescheduling

- Information exchanged needed between different intelligent systems related to component failures/degradation
- High-Level Closed Loop Control
  - First suggested in [Robinson, P., “Fault Management (FM) as a Controller”, AIAA Infotech@Aerospace Conference, Boston, MA, 2013]
- Specific Projects: Astrobotic VOLT Rover, ISS Urine Processing, Graywater, xEMU PLSS, EPSs (Cubesats, LabSats, DoD’s PWSAs, Gateway EPS HW Model, Etc.)

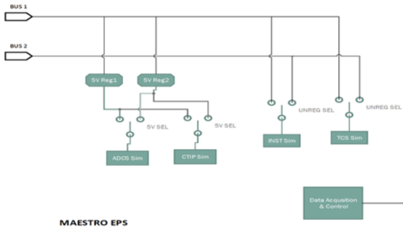
Level 3: Mission Recovery or Repair Examples



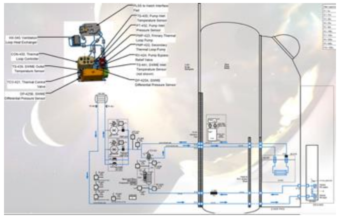
MAESTRO EPS
BOARD 1

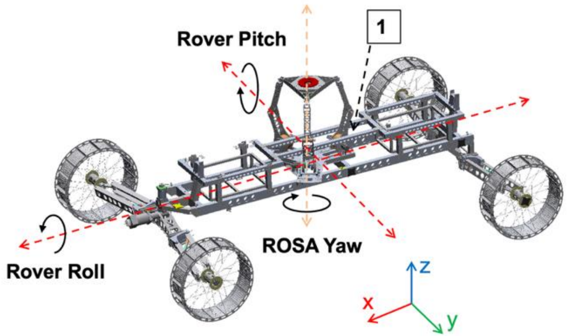


MAESTRO EPS
BOARD 2

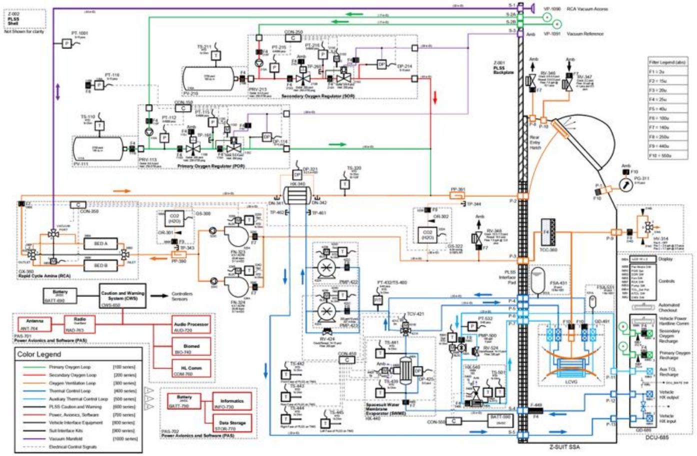


MAESTRO EPS
BOARD 3





Rover Coordinate System



Level 3: Mission Recovery or Repair Required Information Exchange

External versus Internal

External:

- Physical characteristics: Dimensions, Masses, Moments of Inertia
- Physical Connection Points along with maximum, squeeze, force and torque vector spaces
- Electrical Power Charging points along with electrical parameters
- Comm sides, frequencies, minimum power required, max allowed

Internal:

- LRU extraction procedure, LRU installation procedure
- Component Repair Procedures: Most Difficult, Open Scope

Lessons Learned / Pitfalls /Considerations

Use-case timelines/sequences, e.g. if an entity just says No, what can possibly happen next

Non-Happy Case Sequences/Deviations!

- E.g. A and B are to rendezvous at P and do Task T together.
- What if A arrives early? Start early? Wait and do nothing? Do something else?
- What if A will arrive late?
- When there are small/large deviations from plan, people apply their common sense and domain knowledge to adapt. With shared background and knowledge
- With autonomous systems, that common sense and domain knowledge has to be encoded/communicated, ideally in advance.

Lessons Learned / Pitfalls /Considerations

Interaction between sophistication of execution system and data that needs to be provided and/or comms capabilities

Replanning/rescheduling versus adapting schedule/plan

Important issue: communications yes/no/maybe/costly?

Ambiguation in interface standards: Examples are important!

- Stottler Henke is often on the receiving end of information

Integration TESTING (every conceivable direction of information flow)

Questions?
Discussion