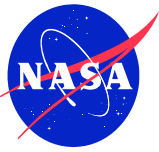


Software Productivity of Field Experiments Using the Mobile Agents Open Architecture with Workflow Interoperability

William J. Clancey,

Michael Lowry, Robert Nado, Maarten Sierhuis

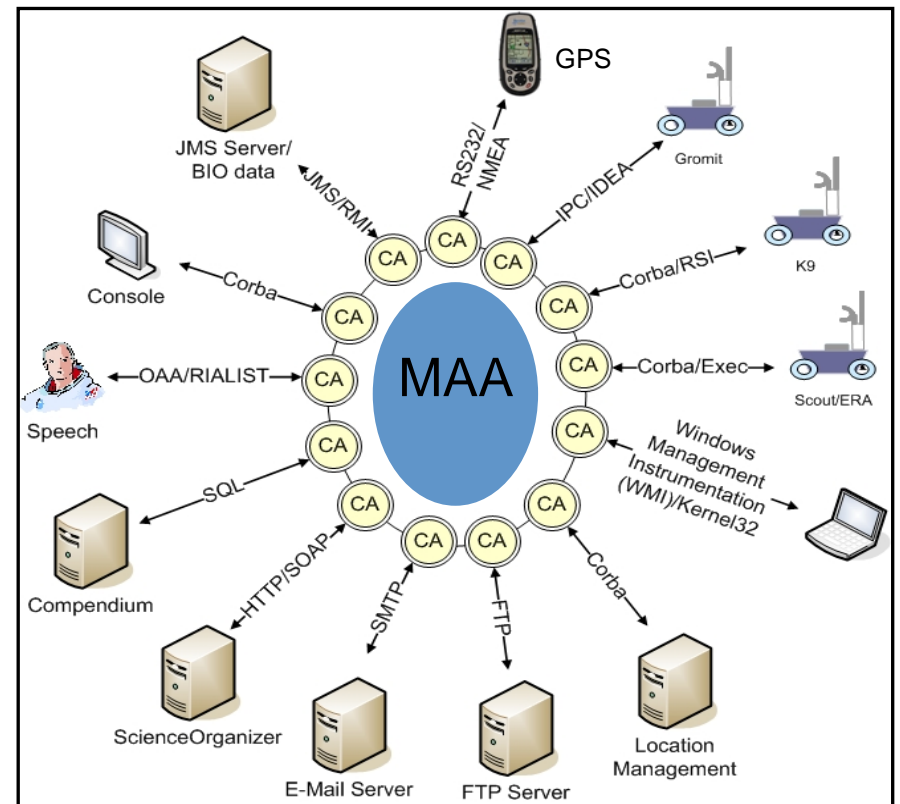
NASA Ames Research Center, Intelligent Systems Division



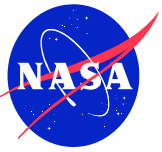
Software Productivity Analysis



- We analyzed a series of ten systematically developed surface exploration systems that integrated a variety of hardware and software components.
- Design, development, and testing data suggest that incremental buildup of an exploration system for long-duration capabilities is facilitated by
 - an open architecture with appropriate-level APIs,
 - specifically designed to facilitate integration of new components.
- This improves software productivity by reducing changes required for reconfiguring an existing system.



Example systems integrated with different protocols using the Mobile Agents Architecture



Agent-Based Systems Integration: Apps Using Mobile Agent Architecture



Mobile Agents 2003-05



Power Agents 2006



Desert-RATS 2006



Metabolic Rate Advisor 2007

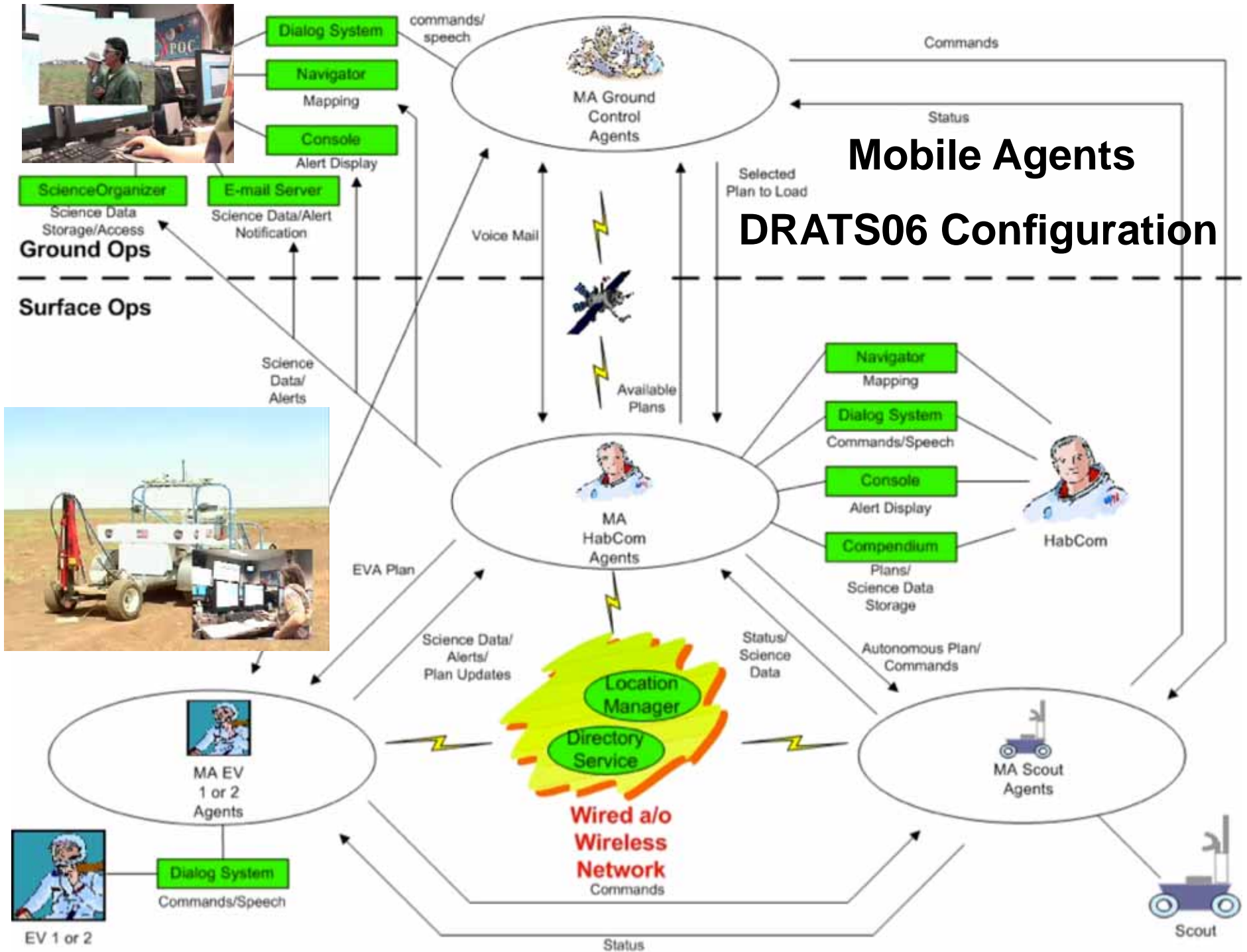


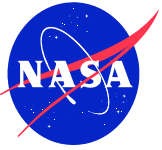
iMAS 2008



OCAMS 2008

Mobile Agents DRATS06 Configuration





Plug'n'Play Perspectives

Data-Control
Bus + Protocol



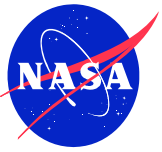
Information-Task Command

Agents + Message Protocol + Transport/
Directory/Data Exchange etc. Services

Communication
Agent

Camera
API

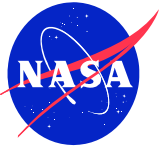




“Agentify” Component (Software Wrapper) =>
Reprogrammable **Task-Level Interoperability**



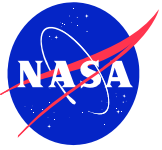
“Boudreaux, take a picture of Astronaut-2”



MAA System Configurations



SYSTEM	FTE	# Platforms	# Functions	Robotic Systems (adapting ERA CA)					# External Systems
				ERA 1	ERA 2	SCOUT	K9	Gromit	
DRATS02	2	1	3	X					4
MDRS03	1.4	4	29	X					7
MDRS04	2.8	4	53	X					10
MDRS05	2.9	5	82	X	X				13
DRATS05	0.9	4	77			X			14
CDS05	0.9	4	80				X	X	11
PA06	1.1	5	45						10
iMAS06	.01	1	51						5
DRATS06	0.7	5	91			X			16
POGO07	0.3	1	63						7
iMAS08	0.05	1	50						5



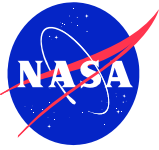
Workflow Automation Examples

■ Habitat Automation

- What is the {battery | generator} {volts | amps | volts and amps}?
- What was the {Max|Min|Avg|Value} of {habitat | generator | batteries | solar} {since | during} {time in past | period}?
- Are the batteries charging?
- When did the {generator | batteries} change status?
- Tell {me | <person> | everyone} when the hab{itat} {amps | volts | voltage} {exceeds | drops below} <#>.

■ EVA Automation

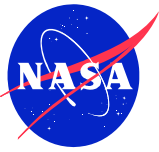
- <Robot name>, {follow | watch | take a picture of} {me | <name>}.
- <Robot name>, inspect <region or waypoint> using <method>.
- Name this place waypoint <#>.
- Start <activity> at <location> for <duration>.
- Tell <support person> that <message>.



Workflow Automation Capabilities



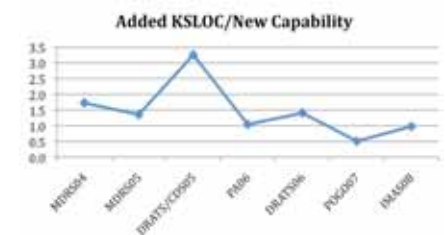
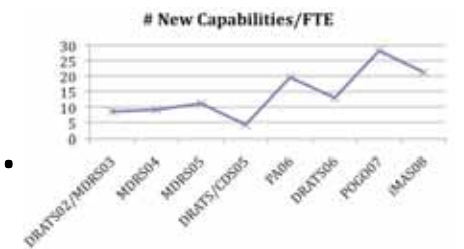
CATEGORY	CAPABILITIES	#
Hardware Integration	Robots, cameras, sensors, instruments, displays, etc.	27
Software Integration	Separate SW components, e.g., EUROPA, Excel, RIALIST	6
Astronaut Health Monitoring	Available data and alerts, e.g., heart rate	6
System Health Monitoring	Computer, power, & life support systems	21
Location Tracking	Logging, tracking, finding assets	13
Human-Robot Coordination	Commands involving robotic systems	31
Plan Management	Getting status and changing the work plan	23
Science Data Logging	All aspects of data collection during EVA	22
Voice Mail	Crew communication via voice messages	4
Alert Management	Control of alert types and modality	7
Voice Command Controls	Control of voice interface	5
		165

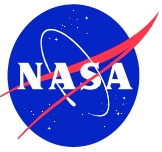


Hypotheses Confirmed: An Agent-Oriented Workflow System Using Composite APIs

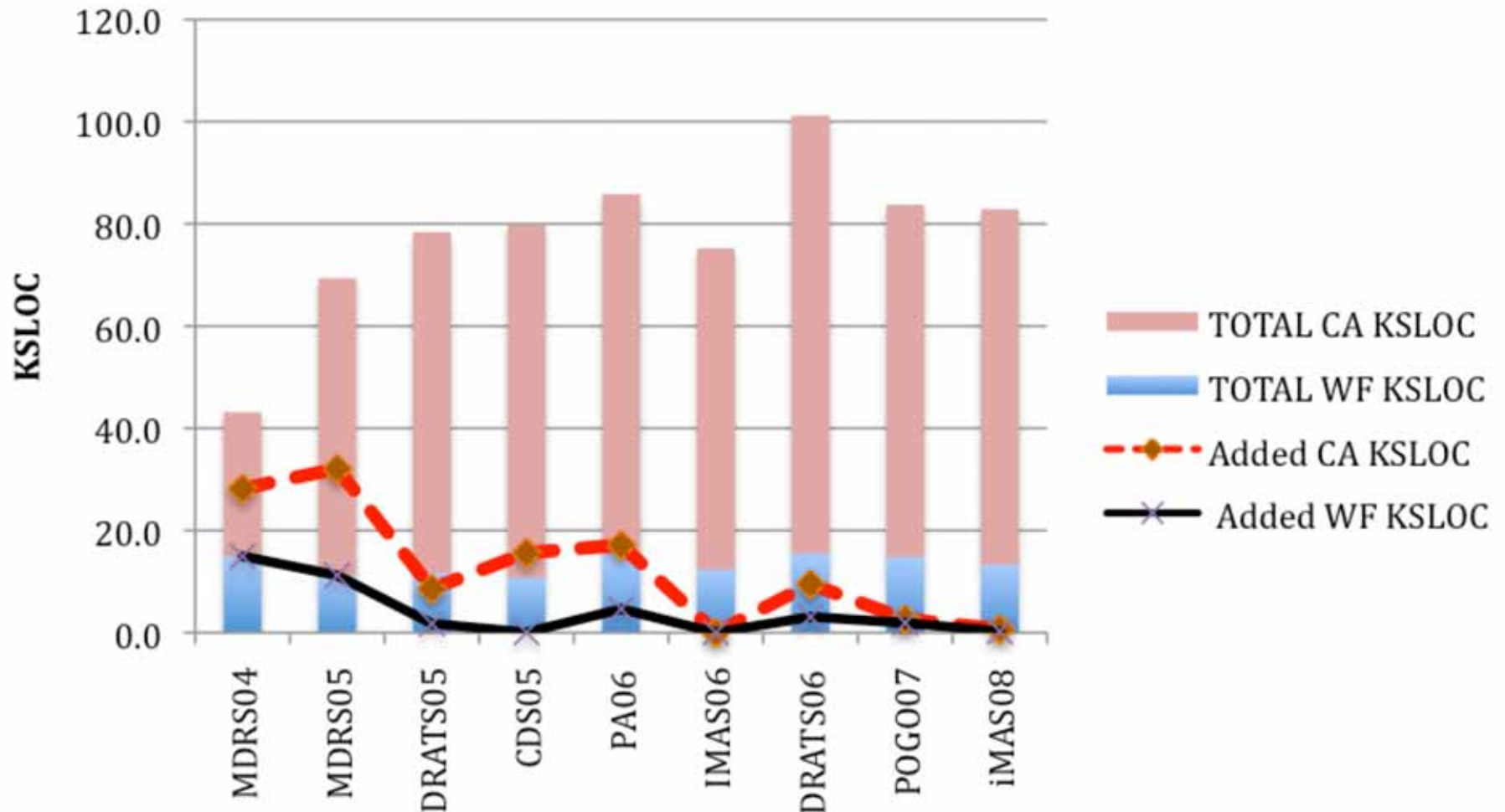


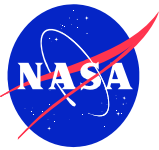
1. New components (e.g., biomedical algorithms, robots, databases) can be integrated without modifying them.
2. Incremental Cost/capability is linear or decreasing as new capabilities are added.
3. Incremental KSLOC/capability is linear or decreasing as new capabilities are added.



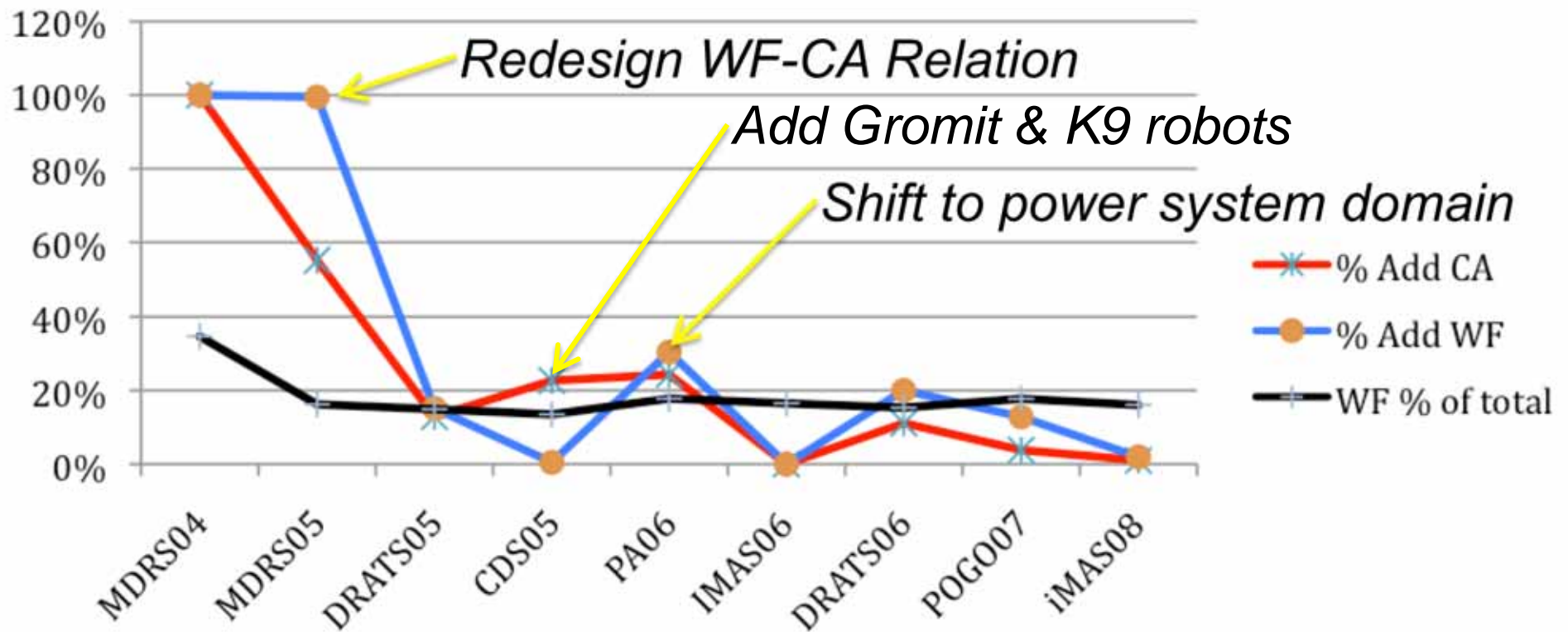


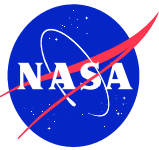
Total Thousands of Lines of Code (KSLOC) Dominated by ComAgents



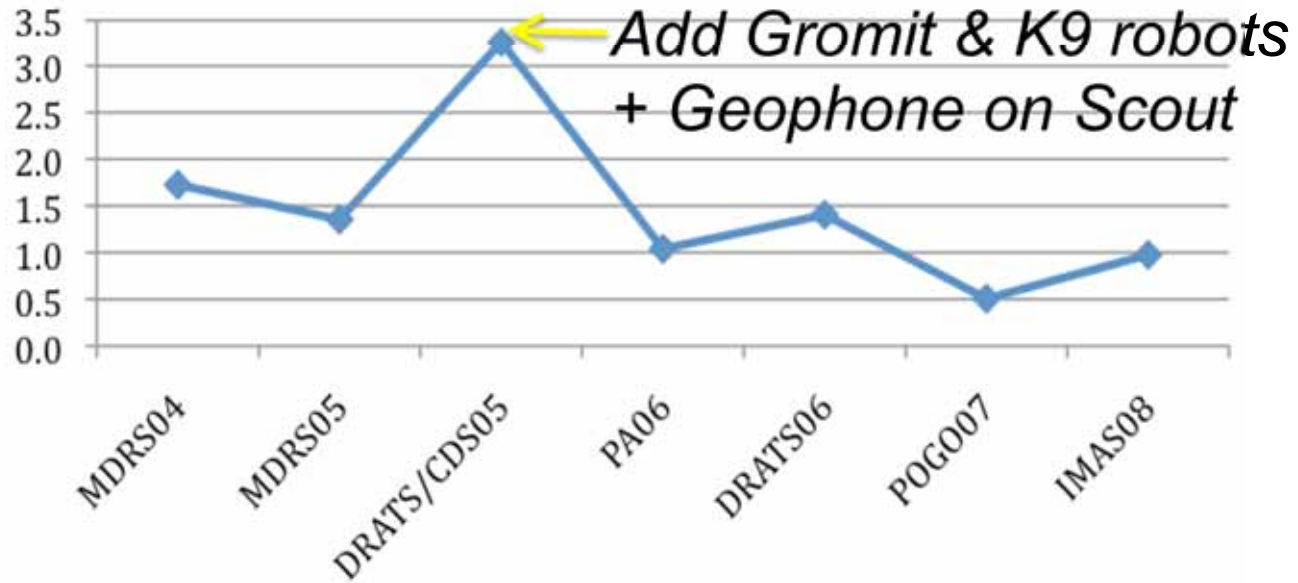


Percent KSLOC Added and Workflow KSLOC as Percent of Total System KSLOC

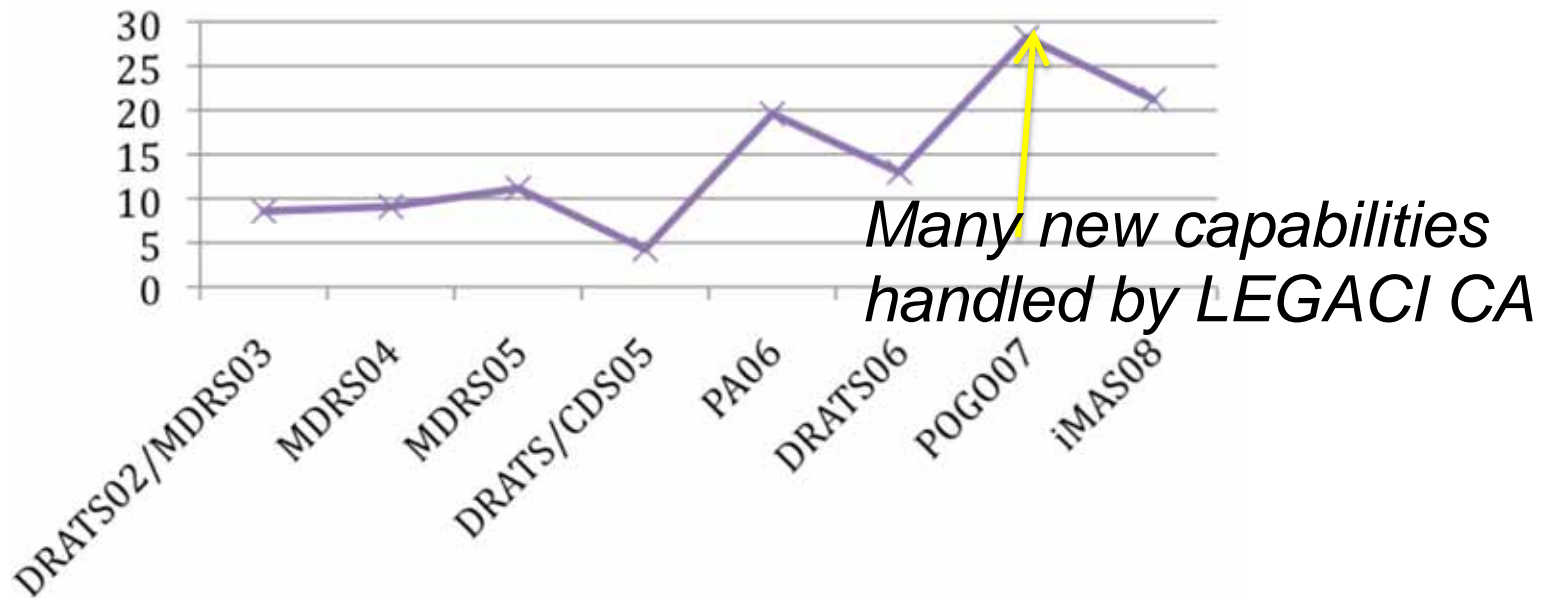


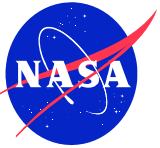


Added KSLOC/New Capability

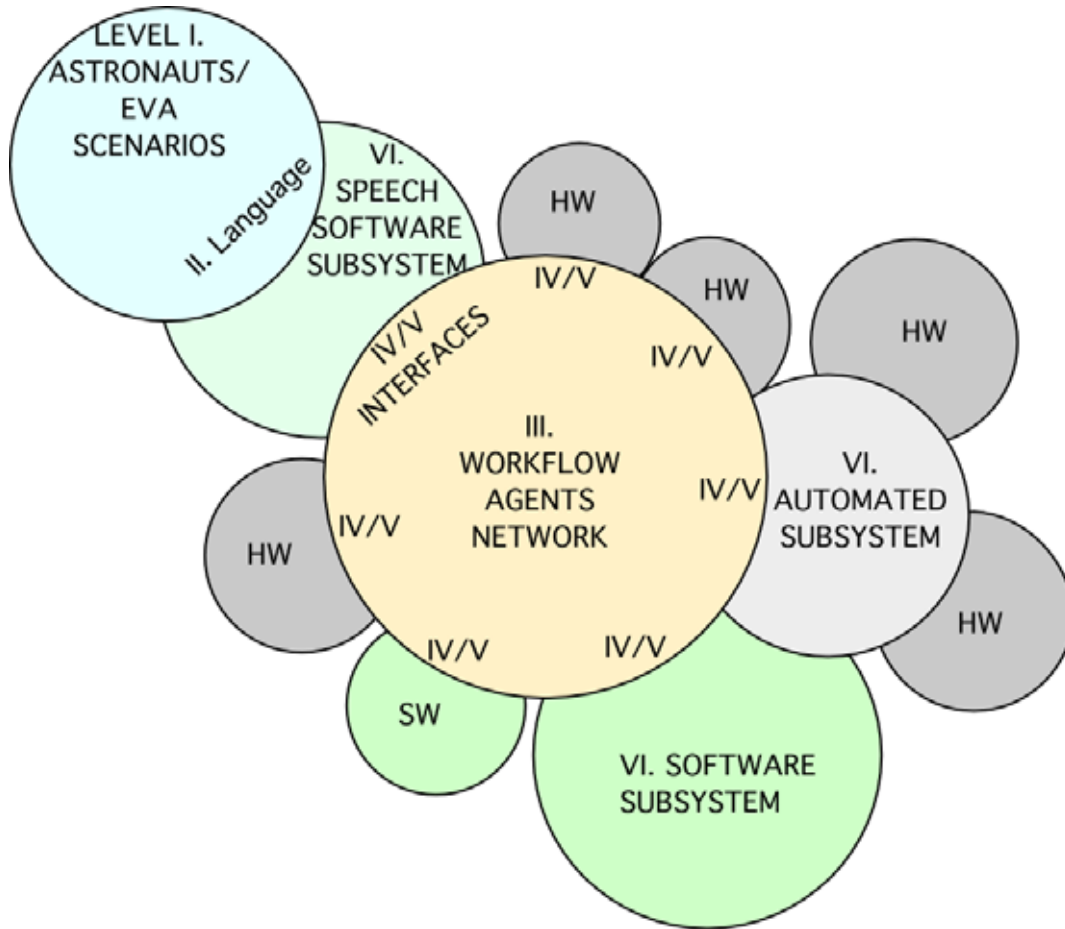


New Capabilities/FTE

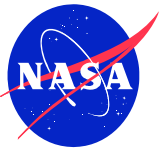




Logical Design Levels => Team Interactions



- I. Scenario designers and project managers
- II. RIALIST (voice commanding) programmer
- III/IV. Agent modelers/programmers (Workflow Agents = III; CA = IV)
- V. API programmers for components/subsystems
- VI. External system developers (e.g., robots).



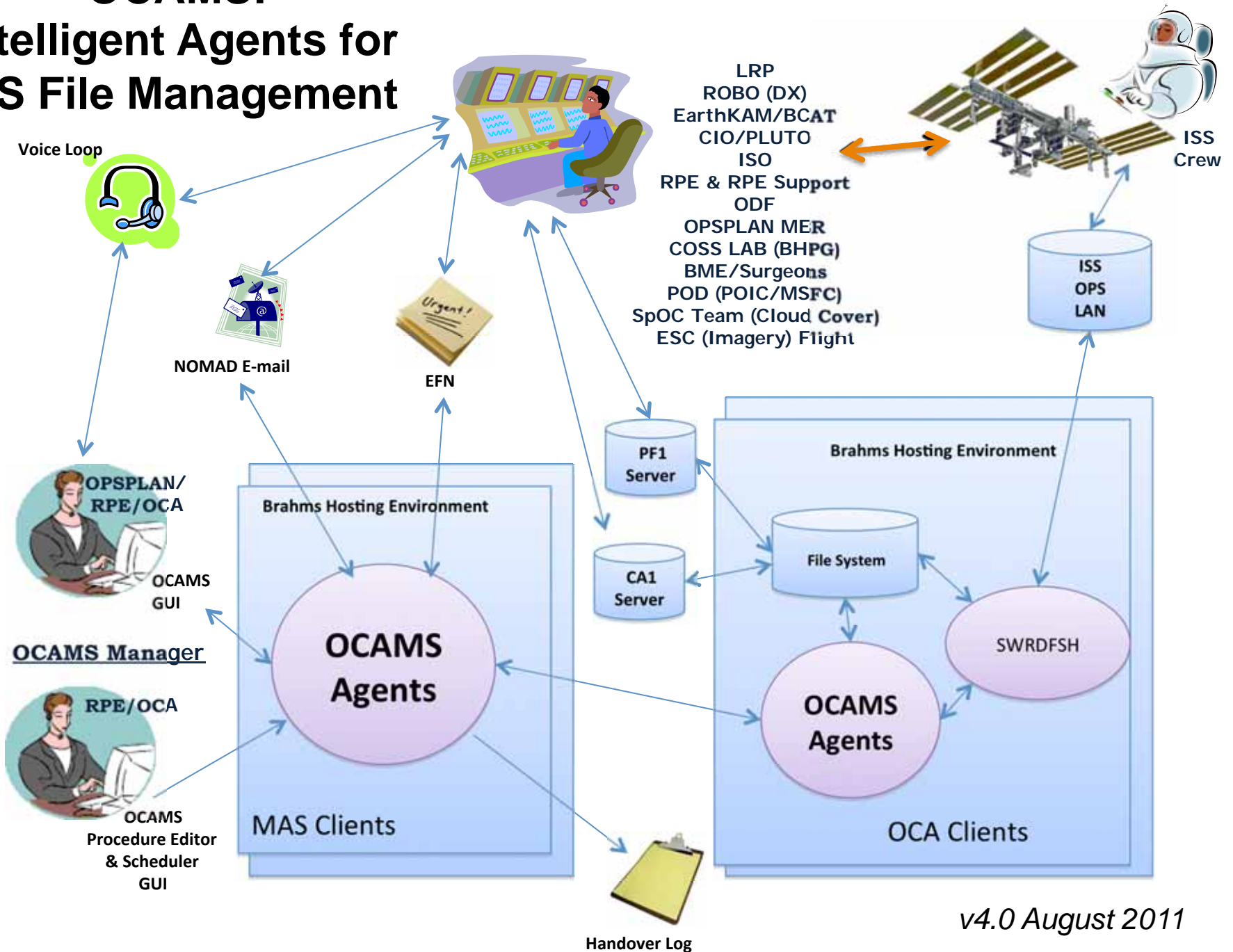
Conclusions: Open Architecture & Interoperability using MAA



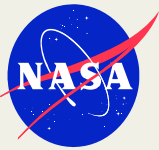
- Modifications and effort required are generally predictable and constant.
- New capabilities neither interfere with existing capabilities nor require increasing complexity of interactions.
- Agent architecture enables modularity that reduces interactions among subsystem teams.
- Voice commanding—in the language of the task—provides high-level protocol for integrating arbitrary HW& SW
=> operations easier to learn and more efficient.



OCAMS: Intelligent Agents for ISS File Management



v4.0 August 2011



For more information...

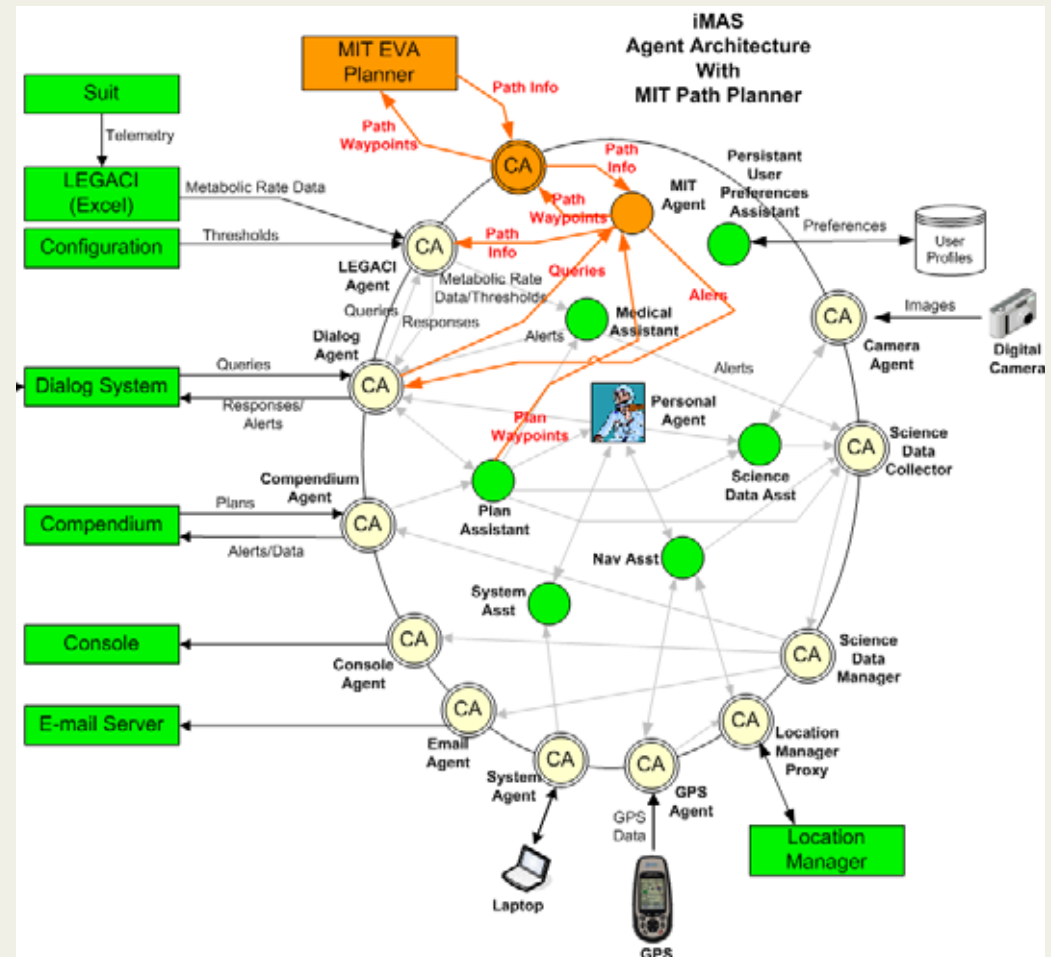


<http://Bill.Clancey.name>

- “Automating CapCom” papers
- “Roles for Agent Assistants in Field Science” (IEEE)
- OCAMS

Youtube.com videos

- “Clancey Power Agents”
- “Clancey Desert RATS”



iMAS Metabolic Rate Advisor + MIT Navigation Assistant