A Multi-Agent Architecture Provides Smart Sensing for the NASA Sensor Web[1][2]

Dipa Suri, Adam Howell
Lockheed-Martin Space Systems Company
Advanced Technology Center
3251 Hanover Street,
Palo Alto, CA. 94304
650-424-2092 dipa.suri@lmco.com
650-424-2737 adam.s.howell@lmco.com

Doug Schmidt, Gautam Biswas, John Kinnebrew, Will Otte,
and Nishanth Shankaran
Vanderbilt University Institute for Software Integrated Solutions (ISIS)
2015 Terrace Place,
Nashville TN 37203
615-294-9573 schmidt@isis.vanderbilt.edu
615-343-6204 biswas@mailhost2.vuse.vanderbilt.edu;
john.s.kinnebrew@vanderbilt.edu
615-343-8197 wotte@dre.vanderbilt.edu; nshankar@dre.vanderbilt.edu

Abstract—Remote sensing missions for Earth Science contribute greatly to the understanding of the dynamics of our planet. Conventional approaches however, impede the scientific community's ability to (1) generate and refine models of complex phenomena, such as, extended weather forecasting, (2) detect and rapidly respond to critical transient events (e.g., disasters, such as hurricanes and floods). This paper describes a more effective approach based on intelligent, networked sensor webs that incorporate seamless dynamic connectivity between spacecraft, aircraft, and *in situ* terrestrial sensors, employs reactive and proactive strategies for improved temporal, spectral, and spatial coverage of the earth and its atmosphere, and uses enhanced dynamic decision-making for rapid responses to changing situations. MACRO, an extension of our earlier work on a multi-agent framework for heterogeneous spacecraft constellations, will provide interoperability and autonomy to achieve the needs for smart sensing in NASA's proposed sensor web. The system capability will be demonstrated via a simulated but salient disaster management scenario on an existing hardware testbed at the Lockheed Martin Advanced Technology Center.

TABLE OF CONTENTS

1. INTRODUCTION

Remote sensing missions for Earth Science provide a wealth of information to help scientists understand the dynamics of our planet. Conventional approaches use a stovepipe operational model with a single spacecraft commanded by and transmitting data to dedicated ground operations centers. These approaches, however, introduce untenable latencies in developing data products that hinder model building and refinement, as well as generating timely responses for hazard mitigation. To address these limitations, future Earth Science missions will therefore operate as part of a sensor web comprised of interlinked platforms with onboard information processing systems capable of orchestrating real-time collaborative operations with other platforms and ground stations [1], as shown in Figure 1.

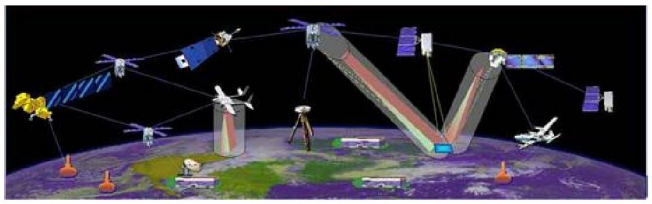


Figure 1 - Using a Sensor Web as a set of interlinked sensor platforms to perform collaborative observations. *Graphic Credit: NASA/GSFC: 2000 Survey of Distributed Spacecraft Technologies and Architectures for NASA's Earth Science Enterprise in the 2010-2025*

To support the needs of future Earth Science Missions, we are developing a *Multi-agent Architecture for Coordinated, Responsive Observations* (MACRO) that provides a natural computational infrastructure for enabling the deployment and operation of a sensor web. MACRO extends our earlier work on the *Adaptive Network Architecture* (ANA) [2] which is a software framework of multiple distributed agents developed using Lockheed Martin Space System Company's (LMSSC) R&D funds and an earlier NASA Earth-Sun Science Technology Office/Advanced Information Systems Technology program (ESTO/AIST) contract to provide localized autonomy on distributed science missions. The scope of this autonomy encompasses dynamic instrument re-configuration and distributed data processing on science missions comprised of spacecraft constellations, where each spacecraft hosts multiple heterogeneous instruments.

An example of autonomy for space based autonomy is the Global Precipitation Measurement (GPM) project [3], a constellation of low earth-orbiting satellites each carrying a variety of passive and active microwave measuring instru-

1

[1] 1-4244-0525-4/07/$20.00 ©2007 IEEE
[2] IEEEAC paper #1198, Version 2, Nov 17, 2006

ments. GPM will require one core and approximately eight support spacecraft to be launched incrementally in the 2009-2014 timeframe. The Core satellite will be the central rain measuring observatory and will serve as the calibration reference system to enable an integrated measuring system. Each support satellite will carry one or more precipitation sensing instruments, e.g., some type of passive microwave (PMW) radiometer measuring at several rain frequencies. Thus, the satellite instruments used together on the constellation will provide a sensor network in space that provides global, bias-free precipitation estimates.

The MACRO extensions described in this paper help overcome current mission limitations by facilitating real-time, reactive data acquisition, analysis, fusion, and distribution, i.e., a Smart Sensing capability in the sensor web context. Design and development of MACRO has been selected for an Advanced Information Systems Technology (AIST) program award by the NASA Earth-Sun Science Technology Office (ESTO). Specifically, MACRO will leverage the earlier ANA technologies to

- Incorporate self-describing sensor, processing, and measurement models, i.e., the Sensor Markup Language (Sensor ML) defined by the Open Geospatial Consortium; and
- Enable collaborative observations between agents via distributed planning, scheduling, and resource management schemes.

MACRO provides many of the key characteristics needed for smart sensing, by integrating standardized middleware, software agent technology, novel planning, scheduling, and resource management algorithms. The integration of these core building block technologies provides additional software development benefits that will be useful in the deployment and operation of future Earth Science Missions. MACRO's use of modern software technologies reduces the risk and development costs as well as operational concerns by enabling the deployment of distributed software components that can be seamlessly replaced and/or upgraded at run-time without affecting the operation of the remaining system.

2. MACRO COMPONENTS

Our prior work on the ANA provides the foundation for MACRO. ANA is a quality of service (QoS)-enabled component middleware framework containing a set of intelligent agents that operate a constellation of spacecraft and behave as an ensemble to ensure that objectives of a remote sensing mission are met, e.g., planning for science operations, science data acquisition, processing, and dissemination. The design of the agents is based on a combination of mature terrestrial standards defined by the Object Management Group (OMG) (www.omg.org) and the Foundation for Intelligent Physical Agents (FIPA) (www.fipa.org) and the implementation of the agents is based on a state-of-the-art component middleware implementation of the Common Object Request Broker Architecture (CORBA) Component Model (CCM) to ensure interoperability across heterogeneous computing platforms (i.e., different processor, OS, and language), reduce development costs, and improve the software's robustness and scalability.

CORBA Component Model

Object-oriented (OO) programming simplified software development through higher level abstractions and patterns, thereby easing the transition to the next step of creating robust distributed systems through the use of distributed object computing middleware, e.g., CORBA. Thus, the application is shielded from dependencies that are generated by the heterogeneous nature of the underlying system, e.g., hardware, OS, and protocol specific details.

Until the advent of the CORBA Component Model (CORBA 3), CORBA 2 programming did not provide a way to logically bundle interfaces into service families leaving that to the developer. Nor did it specify the configuration and deployment of objects as complete applications. This resulted in implementations that were (1) brittle and non-scalable (2) hard to adapt and maintain, and (3) late to reach market. The component model was introduced as a solution where components encapsulate application "business" logic and interact via well defined ports. Standard container mechanisms provide an execution environment for components with common operating requirements and a reusable/standard infrastructure configures and deploys components throughout a distributed system.

The Component Integrated ACE ORB (CIAO) and the Deployment and Configuration Engine (DAnCE) are open source implementations of the OMG's *Lightweight CORBA CCM* and *Deployment and Configuration (D&C)* [5] specifications. CIAO and DAnCE are built atop *The ACE ORB* (TAO). TAO is a highly configurable, open-source real-time CORBA Object Request Broker (ORB) that implements key patterns to meet the demanding QoS requirements of Distributed Real-Time Embedded (DRE) systems. CIAO extends TAO by abstracting key QoS concerns (such as priority models, thread-to-connection bindings, and timing properties) into elements that can be configured declaratively via metadata (such as standards for specifying, implementing, packaging, assembling, and deploying components). Promoting these QoS concerns as metadata disentangles code for controlling these non-function concerns from code that implements the application logic, thus making space system development more flexible and productive. DAnCE extends TAO by allowing application deployers to specify how existing components should be packaged, assembled, and customized into reusable services.

The hierarchical software architecture is composed of two types of logical elements: (i) Sensor Nets and (ii) Science Agents. Figure 2 illustrates the proposed architecture. A Sensor Net is an assembly of software components consisting of Sensor Nodes, data collection and analysis routines, and a single Sensor Net Agent. Each Sensor Node is a component that provides an interface to a particular physical device. The Sensor Net Agent is the manager of the Sensor Nodes, and contains a planner to dynamically generate operational strings that define the set of components that need to be active, and their interactions to satisfy pre-specified goal and task requests. Operational strings are linked chains of tasks that generate specific data products, where the tasks are individual parameterized components that implement a specific input-output mapping.

These tasks are component implementations for operations such as data analysis, selection, and processing algorithms, as well as components that act as messaging proxies to other Sensor Nets or Agents. As a whole, a Sensor Net Agent can be viewed as a reconfigurable data product generator that brings together a number of Sensor Nodes in a configuration to acquire the desired measurements.

Science Agents are proxies for users that request, coordinate, and analyze the data products generated by Sensor Nets to achieve a particular science mission goal, such as collecting magnetic data around fault zones, or monitoring wind and temperature profiles around an evolving hurricane system. Each Science Agent contains a planner to decompose its higher-level goal into sub-goals that can be achieved by individual Sensor Nets. These sub-goals are then allocated to individual Nets via a negotiation process.

Standards for Sensors and Processing Models

The Open Geospatial Consortium (OGC) [6] has made great strides toward specifying interoperable standards for enabling the development of the Sensor Web. For MACRO, this work is extremely beneficial because (1) it provides a common language for the agents to describe tasks, sensors, and measurements, and (2) it supports interoperability with other external tools and systems. These standards and the current implementations have focused on using Web Services for self-describing and interconnecting elements of the Sensor Web.

While Web Services provide an extremely flexible architecture, direct support of the OGC Sensor Web Enablement (SWE) schema in CIAO [7] would allow fine-grained QoS support in DRE systems, where computing resources are severely limited. We are therefore, extending CIAO to create a domain-specific modeling language for the OGC XML Schema that supports import/export/conversion of SensorML descriptions of sensors, processes, etc., into native CIAO components. These additions allow the software developer to target specific portions of the data processing chain for efficient real-time implementation, without sacrificing the standardization and benefits of the OGC SWE activities.

MACRO's Support for Collaborative Observations

MACRO will support collaborative observations via the design and implementation of distributed coordination, plan-

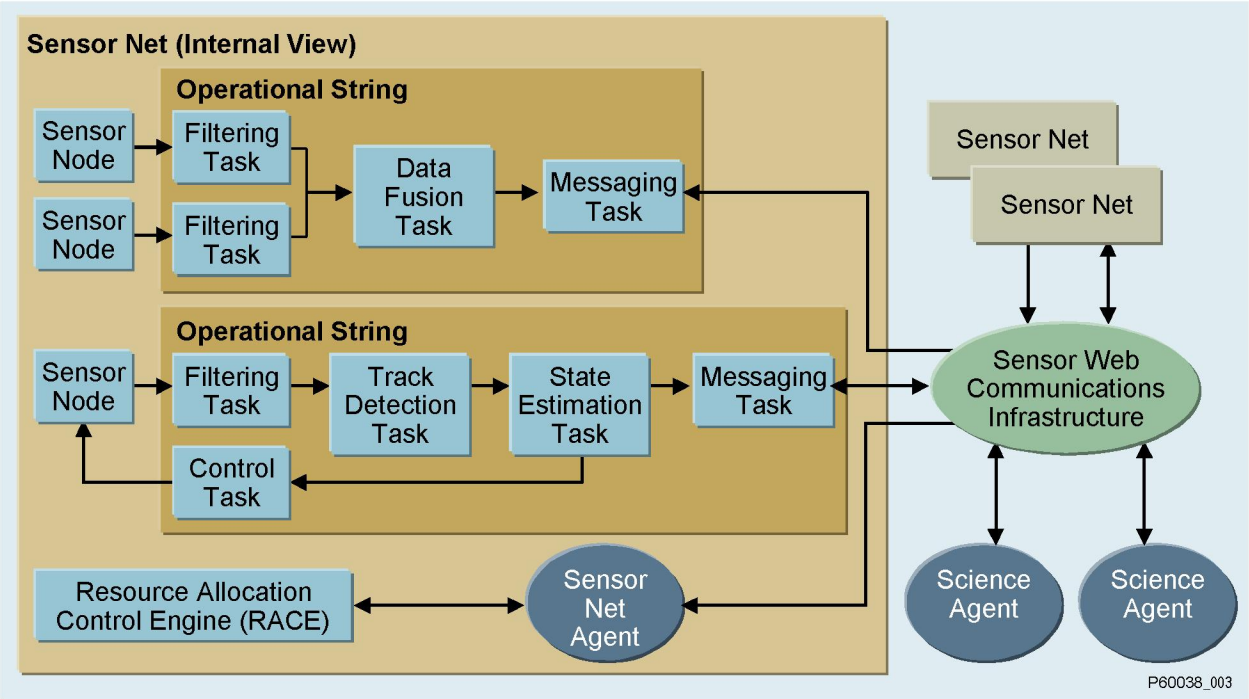


Figure 2 - The Smart Sensing Architecture is defined by a set of Sensor Nets and Science Agents interacting to provide collaborative science observations.

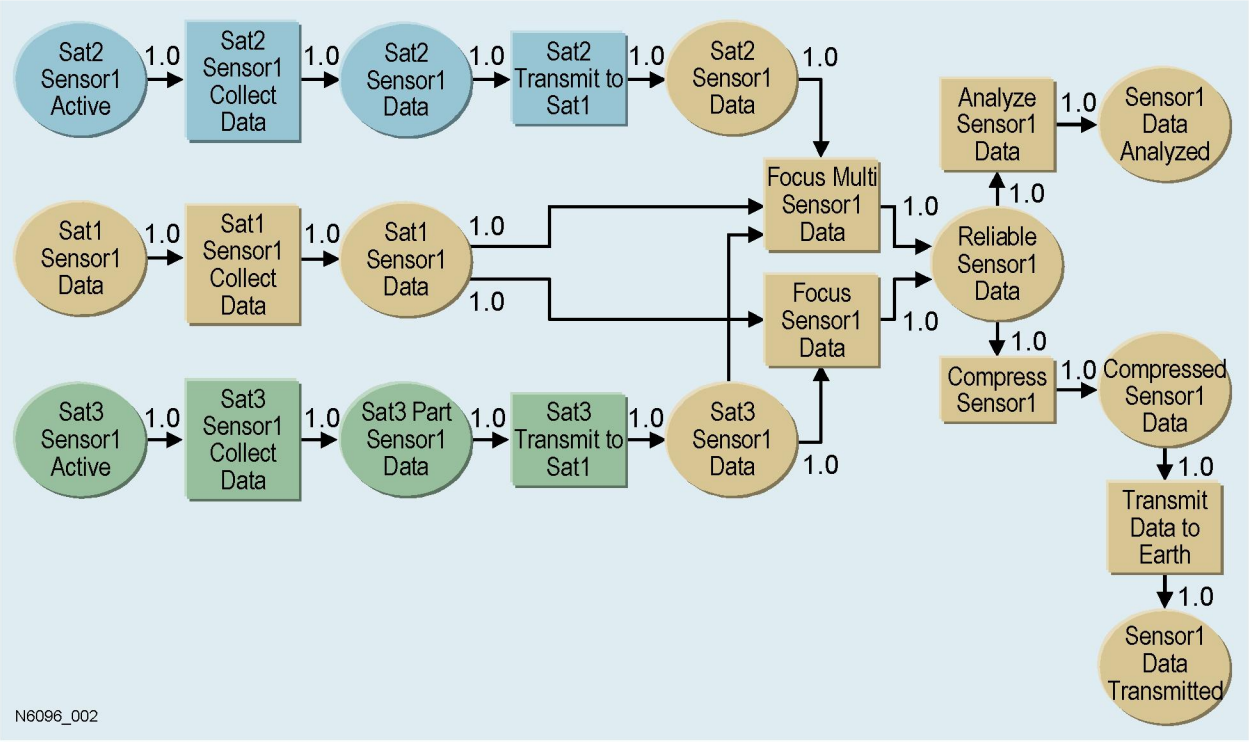


Figure 3 - SA-POP Planner

ning, and scheduling between the Sensor Nets and Science Agents. In the Sensor Web, each Sensor Net Agent requires the capability to plan and schedule observations for the Sensor Nodes that it manages in response to variable priority requests from other elements of the Sensor Web. In addition, Science Agents negotiate with the Sensor Nets and potentially each other to achieve their science objectives without explicit direction from an end user or central authority.

Moreover, system resources are to be (1) allocated to components that perform the desired operations, and (2) managed to ensure desired system performance is met throughout the lifetime of the system. These collaborative observations are realized through the use of:

- A Contract-net (C-Net) based negotiating mechanism that implements a bidding scheme directed by a Science Agent to allocate sensing and processing tasks to the distributed Sensor Net Agents in the Sensor Web.
- A Spreading Activation Partial Order Planner (SA-POP) that forms the core of the Sensor Net and Science Agents.
- A Resource Allocation and Control Engine (RACE), that integrates multiple resource management algorithms for (re)deploying and (re)configuring application components in DRE systems.

The remainder of this section describes each of these technologies and explains how they support collaboration in Sensor Webs.

Contract-net (C-Net)-based Negotiating Mechanism

The C-Net based scheme is designed to provide a cooperative framework for collaboration among autonomous agents to meet overall science mission goals. In this case, the individual science agents are designed to negotiate with each other using a C-Net based protocol to take on subgoals [8] and tasks that contribute to solving the overall mission goals. This approach has the advantage of responding dynamically to changing mission goals based on input by scientists and policy makers, predictive modeling or further data analysis. The bidding policy is based on a combination of the part and machine centered mechanisms that have been designed for scheduling batch-oriented manufacturing operations [9].

Spreading Activation Partial Order Planner (SA-POP)

SA-POP forms the core of the Sensor Net and Science Agents. As shown in Figure 3, the novel, computationally efficient SA-POP in the Sensor Net Agents starts with the initial goals allocated to the individual systems by the Science Agents and generates partial-order task sequences i.e., Operational Strings, for achieving specified goals so that the expected utility (the product of benefit with likelihood of success) is maximized [10, 11, and 12]. This is achieved by using a spreading activation mechanism that is applied to a task network, where the desired goal nodes are assigned utilities in proportion to their importance, and the likelihood of success of tasks can be computed given the current state of the system and the environment. Individual tasks in the Operational Strings are then mapped to available executable software components that satisfy existing pre- and post-

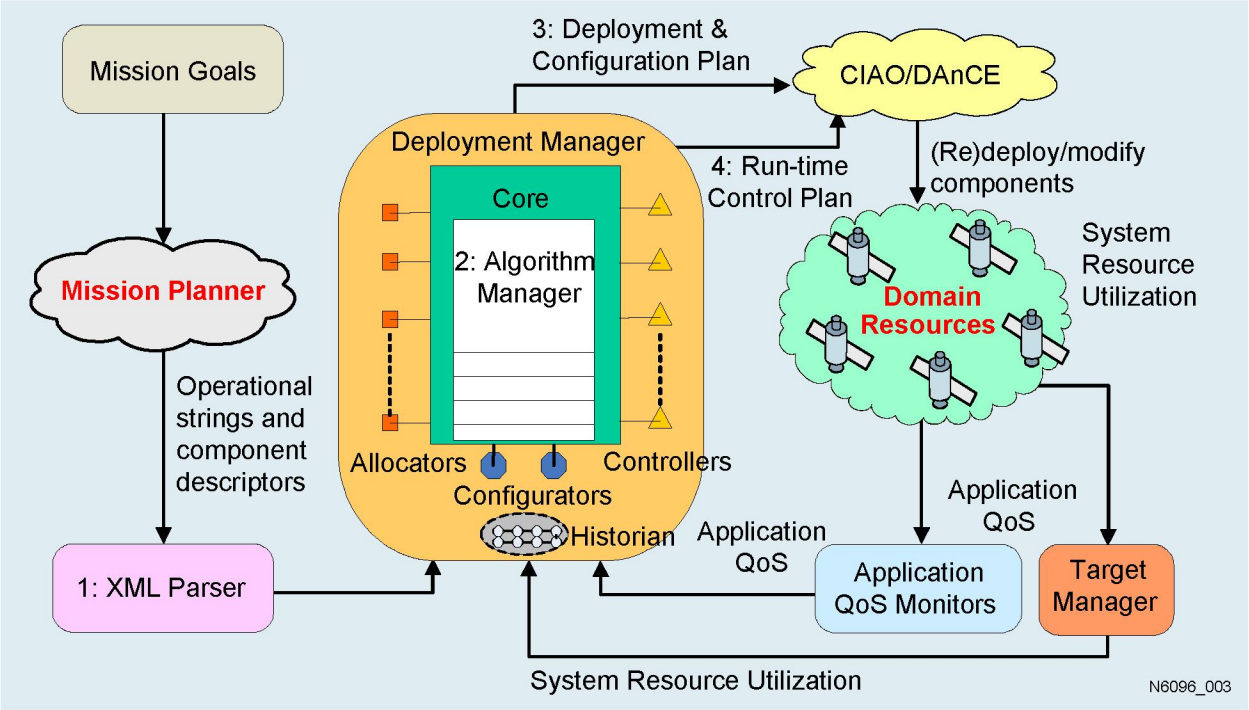


Figure 4 – RACE Architecture

conditions, and meet specific resource constraints, e.g., the planner may pick a data compression task and then select an appropriate component implementation for a chosen compression algorithm. Operational strings are finally given as input to RACE, which provides reusable algorithms for (re)deploying components onto nodes and managing application performance. RACE allocates resources to application components based on their resource requirements and QoS characteristics. RACE can also redeploy and/or reconfigure application components using the implementation options available in the shared task map to ensure the desired end-to-end QoS requirements of operational strings are not violated [13].

Resource Allocation and Control Engine (RACE)

RACE (Figure 4) is a reusable framework that separates resource allocation and control algorithms from the underlying middleware deployment, configuration, and control mechanisms so that different algorithms can reuse these common middleware mechanisms to (re)deploy components onto nodes and manage the node's resources among competing applications. RACE provides a range of resource allocation and control algorithms that use the middleware deployment and configuration mechanisms of the OMG D&C specification to allocate resources to operational strings and control system performance after operational strings have been deployed.

RACE's algorithms determine how to deploy and redeploy operational strings of application components at system initialization and during runtime. Its allocation algorithms determine the initial component deployment using a bin packing algorithm that maps these components to the appropriate target nodes based on available system resources. For example, an allocation algorithm could apportion CPU resources to components in such a way that avoids saturating these resources.

In addition, RACE's control algorithms adapt the execution of an operational string's components at runtime in response to changing environmental conditions and variations in resource availability and/or demand. For example, a control algorithm could (1) modify an application's current operating mode, (2) dynamically update component implementations, and/or (3) redeploy all or part of an operational string's components to other target nodes to meet end-to-end QoS requirements.

The RACE architecture consists of the entities shown in Figure TBS. These entities are implemented as CCM components using CIAO and are deployed via DAnCE. The key entities in RACE are described below:

- **Application QoS Monitors** are CCM components that track the performance of application components by observing QoS properties, such as throughput and latency. One or more Application QoS Monitors are associated with each type of application component.
- The **Target Manager** is a CCM component defined in the OMG D&C specification that receives periodic resource utilization updates from resource monitors within a domain. It uses these updates to track resource usage of all resources within the domain. The Target Manager provides a standard interface for retrieving information pertaining to resource consumption of each component and

an assembly in the domain, as well as the domain's overall resource utilization. The Target Manager provides information on resource utilization component ports in operational strings.

- The **Deployment Manager** is an assembly of CCM components that encapsulates and coordinates one or more allocation and control algorithms. This manager deploys assemblies by dynamically allocating resources to individual components in an assembly. After assemblies are deployed, the Deployment Manager manages the performance of (1) operational strings and (2) domain resource utilization. This manager ensures desired performance of the operational strings by performing the following actions to the components that make up the operational strings: (1) dynamically (re)allocating resources to the component, (2) dynamically modifying component parameters, such as execution mode, and/or (3) dynamically replacing the component implementations

Combining C-Nets, SA-POP and RACE

The Sensor Net and Science Agents use the C-Net negotiation mechanism for the collaborative use of resources and task allocation to meet the larger mission goals. The details of this mechanism will be the focus of the effort for the duration of the project. At the individual Agent level, the SA-POP generates refined operational strings that are given as input to RACE, which provides reusable algorithms for (re)deploying components onto nodes and managing application performance. RACE allocates resources to application components based on their resource requirements and QoS characteristics. RACE can also redeploy and/or reconfigure application components using the implementation options available in the shared task map to ensure the desired end-to-end QoS requirements of operational strings are not violated [13].

3. EXPERIMENTAL DEMONSTRATION

To demonstrate the flexibility and functionality of MACRO, a hardware-in-the-loop prototype implementation is created based on a disaster management scenario. The future of detecting, tracking, and managing potential disaster scenarios is fundamentally dependent on acquiring heterogeneous complementary data from spatially distributed sources in or near real-time, and processing this data with high-fidelity prognostic weather, climate, and solid Earth models to make predictions of evolving weather patterns and related situations. These prediction models will require the fusion of data from remote and *in situ* observations so that rapid responses to potential natural disasters, such as floods and landslides, caused by mid-latitude weather systems or severe weather events, such as hurricanes and tornados can be initiated [14]. Furthermore, timely coordination between the sensor platforms is required to continue monitoring the targeted regions that need special attention. The monitoring timescales can range from the several minutes to hours depending on the severity of the event.

An example scenario shown in Figure 5 involves the autonomous detection of an impending flood or landslide by combining data gathered from *in situ* soil moisture sensors, airborne sensor platforms, and remote sensors monitoring regional weather patterns. Initial analysis of the *in situ* sensor data by a Sensor Net resident on the sensor pods detects high moisture content in the surrounding soil, and sends an alert to notify other agents throughout the Sensor Web. A Science Agent tailored for detecting landslides recognizes the alert as a precondition and initiates a chain of negotiations to update model predictions that validate the potential event via coordinated observations among other sensing elements, such as the GPM spacecraft constellation. A Sensor Net on the GPM Core satellite could coordinate with its support spacecraft to acquire higher spatial and temporal resolution sampling of rainfall intensity in the localized area surrounding the *in situ* sensors. In addition to higher-fidelity monitoring of the weather patterns, additional solid earth information would be required for validation and detection of the actual landslide event. An In-SAR constellation could respond in parallel to the Science Agent's requests for

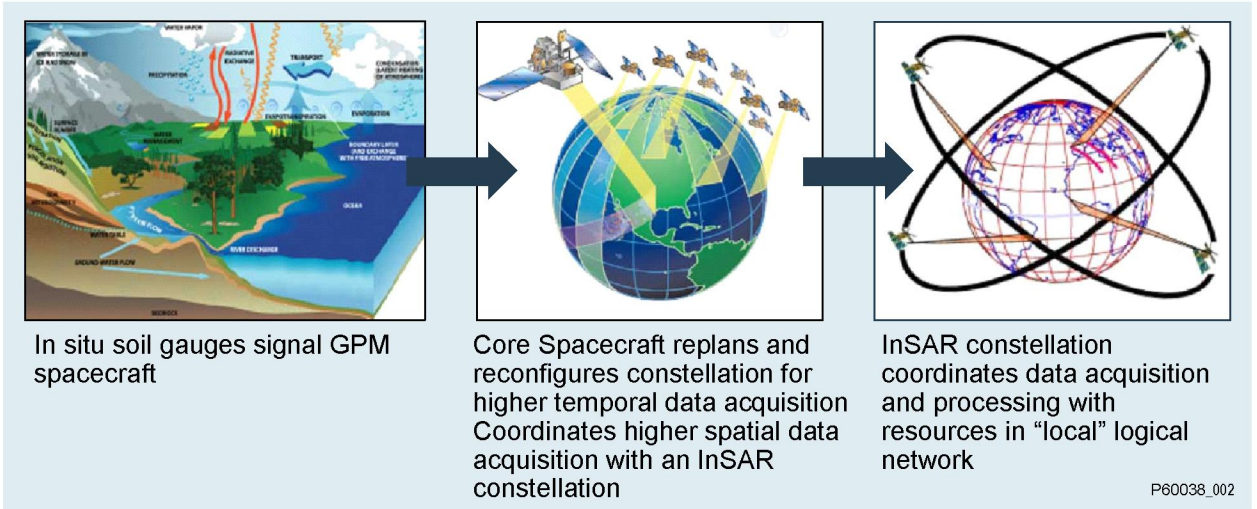


Figure 5 – A Sensor Web Operational Concept for Earth Science involving in situ and remote sensor platforms

higher temporal resolution data acquisition of surface movement and density changes in the same region. Unmanned aerial vehicles with LiDAR instruments operating near the region could also respond to the Science Agent, offering additional higher spatial resolution data to improve estimates of the landslide's location and severity.

Our demonstration will use the Lockheed Martin Advanced Technology Center's (ATC) Distributed Systems Lab (DSL) to emulate several spacecraft driven by a weather model groundstation (Figure 6). The DSL testbed consists of two classes of robots, and the requisite infrastructure for emulating multiple spacecraft in a two dimensional plane. Figure 7 shows the air-bearing robots floating on a 10'x12' granite table using an on-board cold gas system for propulsion and flotation, while reaction wheels are used for attitude control. Each robot has at least one on-board processor and an 802.11b wireless link for command and telemetry interfacing via a Windows ground station. The larger class of robots, called the Microbot, has two heterogeneous computing assets representing the Spacecraft Bus and Payload processors; a PC/104 form factor PowerPC running Linux and a PC/104 form factor Intel Pentium III running VxWorks. The smaller Picobots contain a single Intel xScale processor running Linux.

These Microbots and Picobots simulate the operation of the GPM and InSAR formations, acquiring simulated data representative of the proposed missions. Evaluation on the impact of high temporal data on numerical weather prediction (NWP) could be simulated using the Fifth Generation Mesoscale Model (MM5) (Penn State/NCAR) or WRF (Weather Research and Forecast Model – also NCAR) using observation nudging – four dimensional data assimilation (FDDA). One method would involve simulations using real data that would have been collected during an actual weather event. This data, taken at multiple temporal resolutions, would be used as input to MM5 at different time frequencies to evaluate the impact on the forecast model. Interaction between the weather model and a set of Science Agents developed to discover landslides and precursor weather conditions drive the operating mode, formation spacing, and location of the robotic elements of the testbed.

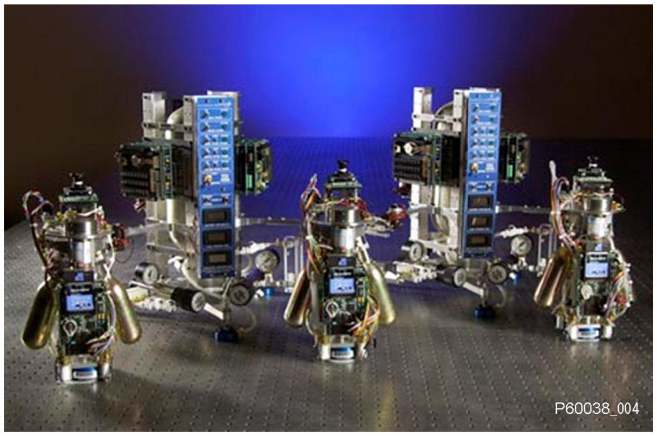


Figure 7 - The ATC Microbots and Picobots provide 2-D hardware simulation of spacecraft

4. CONCLUSIONS

In a large-scale system like the sensor web, the sheer number of available components poses a combinatorially large planning, scheduling, and resource allocation problem for identifying and executing task (component) sequences to achieve specified goals in a dynamic and uncertain environment. The MACRO's use of a decision theoretic partial-order planner and scheduler coupled with an intelligent resource manager, the Resource Allocation and Control Engine (RACE), enables efficient implementation for autonomy.

The use of Agent technology adds to the cumulative benefits of MACRO since correctly and efficiently engineering complex software architectures/systems that have many dynamically interacting components, and complex coordination protocols, is hard. Agent technology is a tool for managing the systems engineering complexity such as specification, design, implementation, and verification of systems with these characteristics.

In conclusion, MACRO provides many of the key characteristics needed for smart sensing, as shown in Table 1.

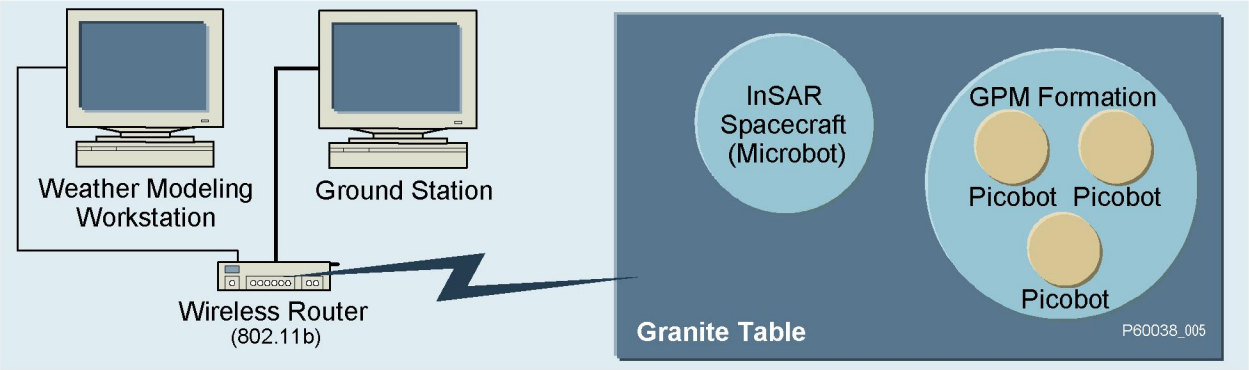


Figure 6 – A schematic of the disaster management demonstration configuration using the Lockheed Martin DSL hardware testbed being driven by an MM5 weather model.

Table 1 – MACRO supports Smart Sensing in the Sensor Web

Sensor Web Characteristics [15]	Supporting MACRO Elements
Ability to accommodate disparate remote sensing & in situ measurement platforms	CIAO middleware
Node aggregation, replacement, and upgrades with new hardware & software	MACRO Agents in conjunction with RACE
Scalability without adversely affecting throughput and/or response time	CIAO middleware
Dynamic deployment and configuration into varying topologies & node relationships	CIAO middleware and RACE
Ability to handle different command/control mechanisms flexibly	MACRO Agents
Dynamic re-deployment & reconfigurability to combine/merge assets into multiple logical sensor nets	MACRO Agents in conjunction with RACE
Seamless exchange of disparate types of data between platforms	Self-describing sensor, processing, and measurement models (SensorML)
Integrated data analysis, data fusion, event detection, & model building capabilities	MACRO Agents

REFERENCES

[1] NASA Earth Science Strategic Vision

[2] Suri, Howell, Schmidt, Biswas et al., "Onboard Processing using the Adaptive Network Architecture", Proceedings of the Earth-Sun Science Technology Conference, College Park, Md. 2006

[3] Smith et al, "International Global Precipitation Measurement (GPM) Program and Mission: An Overview", Measuring Precipitation from Space: EURAINSAT and the Future, 2004

[4] Object Management Group. Light Weight CORBA Component Model Revised Submission, OMG Document realtime/03-05-05 edition, May 2003.

[5] Object Management Group. Deployment and Configuration Adopted Submission, OMG Document ptc/03-07-08 edition, July 2003.

[6] http://www.opengeospatial.org

[7] http://www.dre.vanderbiltcs.wustl.edu/~schmidt/CIAO.html.

[8] Parunak, H. V. D., "Characterizing the manufacturing scheduling problem", Journal of Manufacturing Systems, Vol. 10, No. 3, pp. 241-259, 1991.

[9] Biswas et al. Proceedings of SPIE, "Architectures, Networks, and Intelligent Systems for Manufacturing Integration", Pittsburgh, Pennsylvania, 15-16 October 1997, pp. 108-115.

[10] G. Biswas et al., Task Planning under Uncertainty using a Spreading Activation Network, IEEE Trans. on Systems, Man, and Cybernetics, vol. 30, no. 6, pp. 639-650, 2000.

[11] G. Biswas, D. Schmidt et al., A Decision-Theoretic Planner with Dynamic Component Reconfiguration for Distributed Real-Time Applications, in review National Conference on AAAI, Pittsburgh, PA, 2006.

[12] P. Laborie, Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results, Artificial Intelligence Journal, vol. 143, pp. 151-188, 2003.

[13] D. Schmidt, G. Biswas, et al. A Framework for (Re)Deploying Components in Distributed Realtime and Embedded Systems, poster paper at the Dependable and Adaptive Distributed Systems, Track of the 21st ACM Symposium on Applied Computing, April 23-27, 2006, Bourgogne University, Dijon, France.

[14] Solid Earth Science Working Group Report, "Living on a Restless Planet", 2002.

[15] Advanced Weather Prediction Technologies: NASA's Contribution to the Operational Agencies.

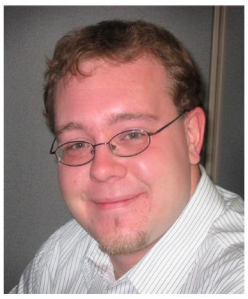
BIOGRAPHY

Dipa Suri *is a staff software engineer at the Lockheed Martin Space Systems Company's Advanced Technology Center with 25 years experience in software development on DoD, commercial, and NASA space missions such as MILSTAR, Iridium, Gravity Probe-B, and HIRDLS. She was the Principal Investigator for the NASA Advanced Information Systems Technology (AIST) contract for a Formation Computing Environment (FCE). She has a B.S., Biology from the University of Santa Clara and an M.S. in Computer Science from San Jose State University.*

Adam Howell *is a Senior Research Scientist at LMSSC ATC and has implemented, and tested an agent-based software architecture for collaborative science operations of multiple spacecraft missions. Prior to joining LMSSC, he implemented an embedded software architecture for real-time coordinated control of formations of unmanned aerial vehicles (UAV) as a Visiting Postdoctoral Scholar, at the Center for Collaborative Control of Unmanned Vehicles (C3UV), University of California, Berkeley. He has a B.S. in Aerospace Engineering from the University of Maryland, and a PhD in Mechanical Engineering from the University of California, Berkeley.*

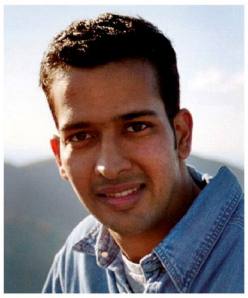
Doug Schmidt *is Full Professor of Electrical Engineering and Computer Science, and Associate Chairman of Computer Science at Vanderbilt University. Previously he was Associate Professor of Computer Science at Washington University and has also been Deputy Director and Program Manager of the Information Technology Office (ITO), DARPA. He was the principal developer of The ADAPTIVE Communication Environment (ACE) an OO toolkit for high-performance and real-time networked systems. He oversees the development of both The ACE ORB (TAO) - a widely used high-performance, real-time ORB endsystem for distributed systems with end-to-end quality of service (QoS) requirements; as well as the Component Integrated ACE ORB CIAO that extends work on TAO via component-based abstractions as defined by the CORBA Component Model (CCM) and Deployment and Configuration specifications. He has an M.S. in Sociology from the College of William and Mary, and a PhD in Computer Science from the University of California, Irvine.*

Gautam Biswas *is Full Professor of Electrical Engineering and Computer Science, at Vanderbilt University. Previously he was Assistant Professor, at the University of South Carolina, Columbia. His primary research areas are in modeling and analysis of complex hybrid systems with applications to diagnosis and fault adaptive control, and the design of intelligent systems for complex problem solving. He has developed innovative algorithms for model-based diagnosis and fault-adaptive control for aircraft and spacecraft systems. He has also developed planning and scheduling systems for robotic and manufacturing applications He has developed a new and innovative learning environment called Teachable Agents Systems developed using this paradigm: Betty's Brain and Billy that is being used for Math and Science education in middle school classrooms. He has a B.Tech, Electrical Engineering, Indian Institute of Technology, Bombay, India, and PhD in Computer Science from Michigan State University.*



John Kinnebrew *is a graduate student in computer science at Vanderbilt University. He received his B.A. in computer science from Harvard University in 2001. His current research focuses on artificial intelligence planning and scheduling techniques in autonomous systems.*

William R. Otte *is a Ph.D. student in the Department of Electrical Engineering and Computer Science (EECS) at Vanderbilt University. His research focuses on middleware for distributed real-time and embedded (DRE) systems. He is currently involved in several aspects of developing a Deployment and Configuration Engine (DAnCE) for CORBA Components. This work involves investigation of techniques for run- time planning and adaptation for component based applications as well as specification and enforcement of application quality of service and fault tolerance requirements. Before joining as a graduate student, William worked for a year as a staff engineer at the Institute for Software Integrated Systems after graduating with a B.S. in Computer Science from Vanderbilt University, Nashville TN in 2005.*



Nishanth Shankaran *is currently a PhD student and a graduate student researcher at the Institute for Software Integrated Systems (ISIS) at Vanderbilt University. He received a master's degree in Computer Science and Engineering from University of California Irvine in 2004. His research focuses on adaptive resource management middleware for distributed real-time embedded systems*