Preliminary Results from a Model-Driven Architecture Methodology for Development of an Event-Driven Space Communications Service Concept

Christopher J. Roberts, Graduate Student *Member, IEEE, Robert M. Morgenstern, Member, IEEE,*
David J. Israel, John M. Borky, Senior Life Member, IEEE, and Thomas H. Bradley

Abstract— **NASA's next generation space communications network will involve dynamic and autonomous services analogous to services provided by current terrestrial wireless networks. This architecture concept, known as the Space Mobile Network (SMN), is enabled by several technologies now in development. A pillar of the SMN architecture is the establishment and utilization of a continuous bidirectional control plane space link channel and a new User Initiated Service (UIS) protocol to enable more dynamic and autonomous mission operations concepts, reduced user space communications planning burden, and more efficient and effective provider network resource utilization. This paper provides preliminary results from the application of model-driven architecture methodology to develop UIS. Such an approach is necessary to ensure systematic investigation of several open questions concerning the efficiency, robustness, interoperability, scalability and security of the control plane space link and UIS protocol.**

I. Introduction

The Space Mobile Network (SMN) is NASA's next generation architecture concept for space communications. It will provide more dynamic and autonomous communications services, analogous to those provided by modern wireless terrestrial networks. The SMN architecture consists of a suite of emerging technologies, including high-availability space link channels, delay tolerant networking and other new protocols, optical communications, and advanced position, navigation and timing technologies [1]. Software and data systems engineering play a key role in the transition to the SMN architecture, enabling interoperability among new and legacy elements, virtualization through software defined radios and other network resources, and automation of processes. Infusion, adaptation and extension of terrestrial wireless communications network best practices are also desirable, where appropriate [2].

A rigorous model-driven architecture methodology is necessary to ensure and manage the desired holistic properties of the SMN, given the necessarily piecemeal implementation and infusion of new elements. This methodology utilizes object-oriented modeling, computational process simulation, virtual and physical prototyping, and technology demonstration activities are methods to generate architectural data and experiential learning [3]. It allows architects to characterize predicted and unanticipated emergent behaviors among new and legacy elements, their interactions with environmental factors, and system performance under various conditions, including nominal, stress, and fault scenarios.

Object-oriented modeling emerged from the discipline of software engineering, but is gaining acceptance as a more generalized systems engineering method [4]. More responsive development processes, such as spiral, evolutionary and agile processes, are also being applied to systems engineering efforts [3]. These software-inspired methods and processes leverage design commonality and patterns, accommodate the reality that a complete set of fully specified requirements is often unknowable at the outset of the design effort, and are responsive to changes in user needs or external conditions, which may invalidate some requirements over time. Modern software-enabled information-intensive systems, such as the SMN, have requirements that are somewhat provisional, as they are subject to change based on experiential learning or dynamic externalities, and consist of heterogeneous elements at various degrees of maturity and obsolescence. A model-driven architecture methodology enables and enforces multi-faceted system evolution by providing the architect and other system stakeholders with the perspectives and tools to understand and manage the complexity of these systems over time [3].

The traditional paradigm for space mission operations relies on highly scripted pre-planned processes between people at space communications service providers and user mission operations centers. Communications with user mission platforms are typically intermittent, due to orbital, geographic, and other space link resource constraints. Today, there is limited or no automation on the user mission platform for invoking space communications services in response to dynamic events, or in the provider network for service provisioning [1]. This paper provides preliminary results from the application of a model-driven architecture methodology to develop a more dynamic and autonomous Space Mobile Network communications concept, known as User Initiated Services (UIS).

Christopher J. Roberts, Robert M. Morgenstern and David J. Israel are with the NASA Goddard Space Flight Center, Greenbelt, Maryland 20771 USA (phone: 301-286-7187; e-mail: christopher.j.roberts@nasa.gov). John M. Borky and Thomas H. Bradley are with the Walter Scott, Jr. College of Engineering, Colorado State University, Fort Collins, CO 80523 USA.

The benefits of UIS include enabling new event-driven and collaborative platform mission operations concepts, reduced user burden for space communications service planning, and more efficient and effective provider network resource utilization. UIS is intended to be widely adopted, and available from NASA and worldwide space communications providers [1].

II. UIS Architecture Model

The UIS concept requires the definition and utilization of a highly-available (i.e., continuous or nearly so) bidirectional communications channel through which users may invoke space communications services (i.e., request a resource agnostic service event, typically for higher data rate/lower availability links) and exchange disposition messages with the provider network until a proposed service event is validated. It is also necessary to define the UIS protocol, consisting of the message types, data structures, machine-to-machine behavioral interaction sequences (including nominal, stress and fault scenarios), and any performance timing constraints, dependencies or interactions with other protocols in the stack. A model-driven architecture methodology using a rigorous and standardized object-oriented language, such as the Systems Modeling Language (SysML), implemented in a robust tool, such as MagicDraw, is well-suited for this challenge [3]. This section presents and discusses the top-level structural and behavioral representations of the UIS system architecture.

A. UIS Concept of Operations and Requirements

UIS has as its distinguishing operational characteristic, user invocation of a resource agnostic space communications service event and automated service dispositioning between the user and space communications provider network. These communications occur through a highly available space link channel between user mission platform(s) and the space communications provider resources. Adopting the architectural taxonomy of the software defined networking community, this channel provides a control plane for UIS service request messages and service event dispositioning [5]. The need for high availability of the control plane space link likely drives the solution to space relay resources for near-earth users [1]. For user mission data service provisioning and execution, higher data rate services may be provided by one or more space relay or direct-to-earth link resources. This UIS operational concept is depicted in Figure 1.

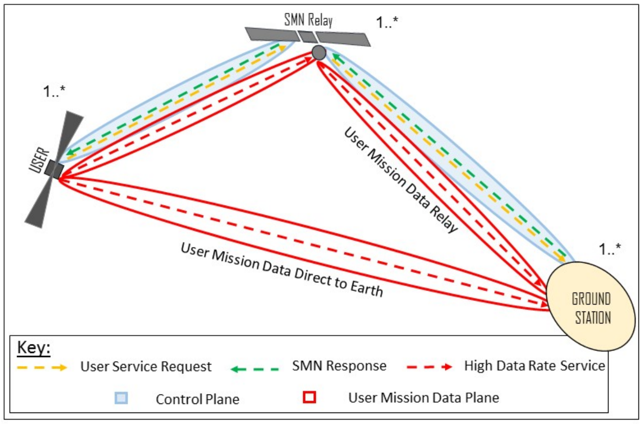

Figure 1: UIS space link concept of operations

Once a user service request message is received by the provider, orchestration of several software services is required to disposition the request. These may include UIS message format translation services, publish and subscribe services to space link resource scheduler(s), publish services to the forward (response) control plane space link resource, UIS thread/instance management services, fault detection and recovery services, cybersecurity services, service accountability reporting and data archiving services. User mission platforms or operations centers will also require several software services to implement UIS. These may include mission-specific UIS request initiation services, UIS message format translation services into platform command sequences, status monitoring and UIS event response services, fault detection and recovery services, and advanced position, navigation and timing services.

This concept of operations has led to a provisional set of UIS capability requirements, capturing stakeholder needs and constraints in Table 1. The purpose of this set of capability requirements is to focus and guide the architectural modeling, simulation, prototyping and demonstration activities. More refined and detailed requirements will emerge based on iterative analysis, synthesis and evaluation of results from these activities.

Table 1: Provisional UIS capability requirements

Id	Name	Type	Text
UIS1	1.1 Event-Based Requests	Functional	The UIS system shall autonomously initiate space communication service requests based on specified mission-specific events of conditions.
UIS2	1.2 Service Request Ingest	Functional	The UIS system shall ingest space communications service requests.
UIS3	1.3 Route Service Requests	Functional	The UIS system shall route user-initiated service requests for disposition processing.
UIS4	1.4 Broker with External Schedulers	Functional	The UIS system shall broker UIS task dispositioning with external scheduling resources.
UIS5	1.5 Configuration Brokering	Functional	The UIS system shall broker configuration of space communications link resources associated with UIS tasks.
UIS6	1.6 Synchronization	Functional	The UIS system shall provide synchronization mechanisms among (internal and external) elements involved in UIS instance collaborations.
UIS7	1.7 Execution Performance	Functional	The UIS system shall execute UIS space communications instances within network and mission-specified performance.
UIS8	1.8 Service Accounting	Functional	The UIS system shall provide accountability reporting.
UIS9	1.9 Monitor Status	Functional	The UIS system shall monitor UIS service instance status.
UIS10	1.10 Report QoS Metrics	Functional	The UIS system shall provide status reporting on Quality of Service metric performance.
UIS11	1.11 Enforce SLAs	Functional	The UIS system shall enforce mission-specific Service Level Agreements.
UIS12	1.12 Notification	Functional	The UIS system shall notify users of conditions affecting UIS services.
UIS13	1.13 Fault Detection and Recovery	Functional	The UIS system shall provide fault detection, isolation and recovery mechanisms.
UIS14	1.14 Archival	Functional	The UIS system shall archive UIS data.

UIS15	1.15 Operational Availability	Performance	The UIS system shall achieve operational availability requirements (mission or deployment specific).
UIS16	1.16 Reliability	Performance	The UIS system shall achieve reliability requirements (mission or deployment specific).
UIS17	1.17 Compatibility	Design Constraint	The UIS system shall achieve compatibility with existing operations, maintenance and logistics organizations, resources and processes.
UIS18	1.18 Enterprise Security	Design Constraint	The UIS system shall conform to enterprise cybersecurity standards, policies and directives.

B. UIS Use Case Model

UIS is an extension of existing space communications services, enabled by the infusion of automation and orchestration technologies across a heterogeneous mix of legacy and new elements.

Figure 2 documents traditional Tier 1 and Tier 2 space communications use cases, and relates UIS as an extension to these use cases. Usage dependencies are illustrated where one element requires another element for its full implementation. The usage dependencies depicted in Figure 2 establish that all user mission operations use cases (i.e., near-earth, deep space and celestial-body surface) have dependencies on space communications services, and that the provisioning of space communications services are, in turn, dependent on network infrastructure support services. The scope of this preliminary model is limited to investigation of the near-earth operational use case because these scenarios impose the most demanding latency requirements on the UIS architecture, have the greatest number of potential users, and provide more readily accessible opportunities for on-orbit demonstrations that are essential to mature and validate UIS technologies. However, the UIS concept is envisioned to be extended to deep space and celestial surface operational use cases. As indicated by the use case diagram, there are no envisioned extensions to the network infrastructure support behaviors to enable UIS.

C. UIS Structural Domain Model

To facilitate widespread infusion of UIS technologies and adoption of UIS, the UIS system must realize open architecture principles, including modularity, loose coupling, interoperability, and scalability. The deployment strategy must accommodate UIS software modules implementing UIS-enabling services in heterogeneous computational runtime-environments, and on elements of both mission user and space communications provider owned resources in space and on the ground.

In a model-driven architecture, the objective is to establish a top-level structural partitioning that supports realization of open architecture principles [3]. This is achieved by grouping related system information and activities into domains, which collaborate to perform major system behaviors. These domains also establish the basis for allocating requirements to system resources. Interfaces between the domains are often opportunities for standardization. Figure 3 illustrates the domain partitioning of the UIS architecture in a block definition diagram. Each domain element includes the operations of the domain and associated values, which define the system functions and information associated with the element, respectively.

UIS enables the transition from a “human-in-the-loop” paradigm to a more dynamic “human-on-the-loop,” or fully autonomous operations. As a result, the user roles for the UIS system, indicated in the use case diagram and domain

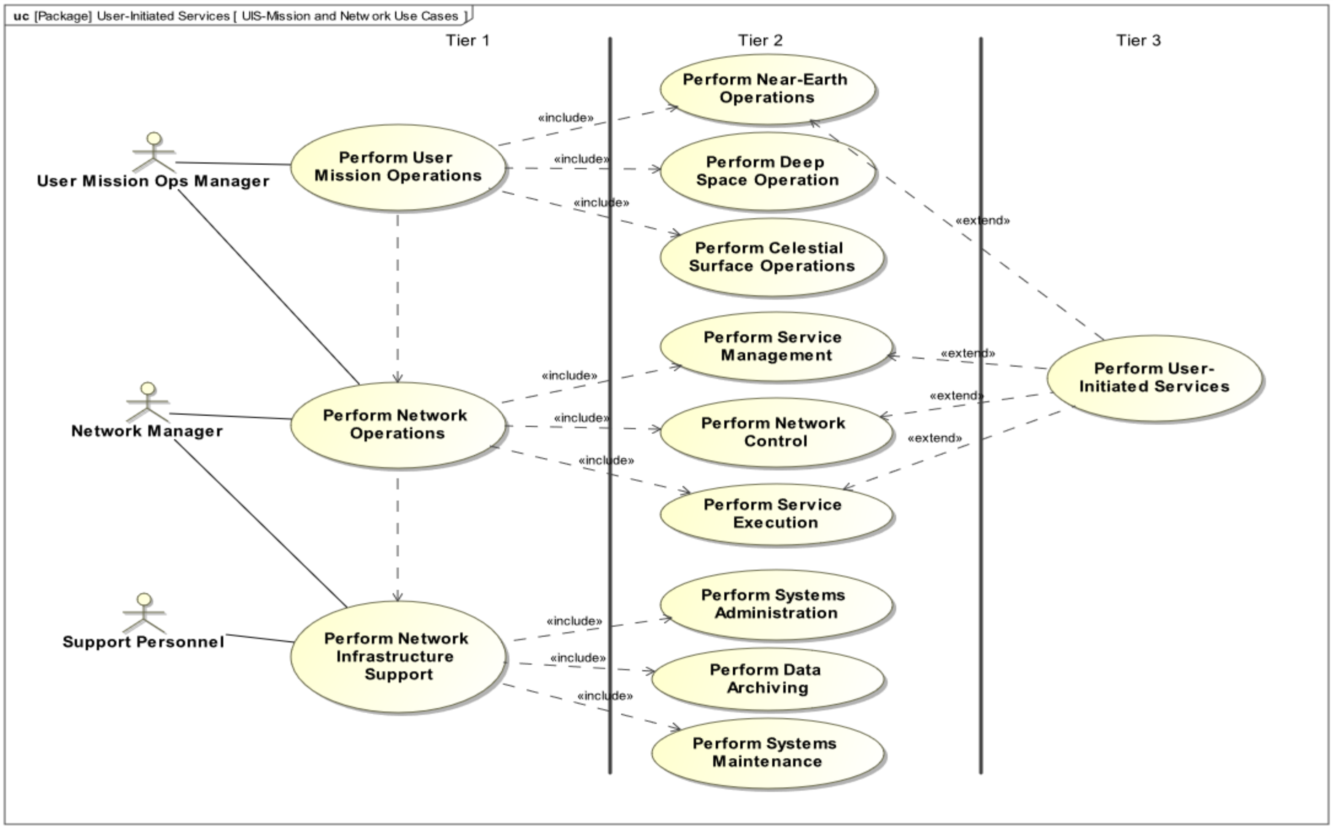
uc [Package] User-Initiated Services [UIS-Mission and Network Use Cases]
Tier 1
Tier 2
Tier 3
User Mission Ops Manager
Network Manager
Support Personnel
Perform User Mission Operations
Perform Network Operations
Perform Network Infrastructure Support
Perform Near-Earth Operations
Perform Deep Space Operation
Perform Celestial Surface Operations
Perform Service Management
Perform Network Control
Perform Service Execution
Perform Systems Administration
Perform Data Archiving
Perform Systems Maintenance
Perform User-Initiated Services
«include»
«extend»

Figure 2: User Initiated Services extend traditional space communications use cases

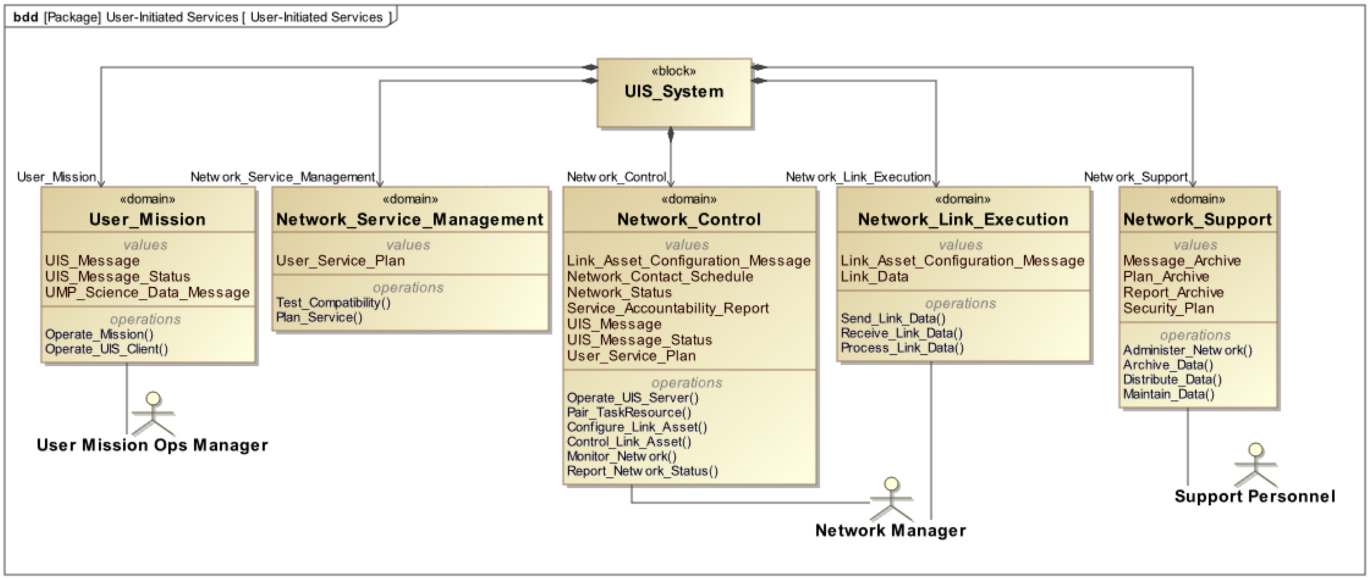

Figure 3: Domain partitioning of the UIS architecture block definition diagram

block definition diagram, are rather limited. User Mission Operations Manager, Network Manager and Support Personnel roles are defined. External system resources also take the form of user roles in SysML, but none have been identified for UIS at this time.

D. UIS Activity Model

In a model-driven architecture, system structure and behavior are rigorously defined. Refinements to ensure completeness, correctness and consistency are made as system understanding is matured through stakeholder feedback, new information and ideas, and results from modeling, simulation, prototyping and demonstrations at various levels of the system hierarchy. Definition of capability requirements and use cases are typically followed by structural partitioning into domains, which forms the basis for requirements allocation to system resources. An activity diagram furthers the architectural specification, and illustrates a thread of activities across the previously defined structural domains and user roles to perform a major system

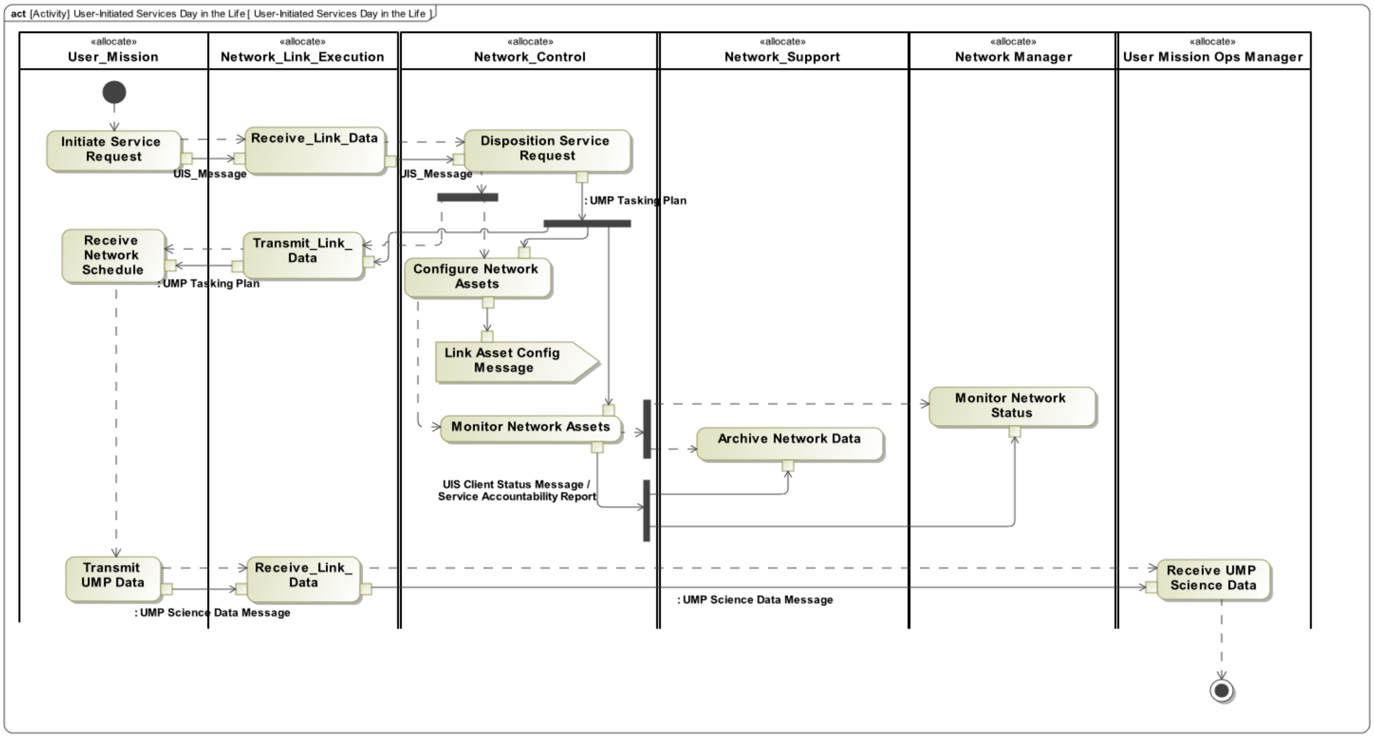

Figure 4: Nominal UIS service instance activity flow diagram

behavior [3]. The construction of an activity diagram provides an early test of the correctness, completeness and consistency of the architecture modeling performed to this point. An activity diagram showing a nominal instance of a user service request, disposition and service event execution is provided in Figure 4. The activity diagram establishes control flows and object flows across the structural domains and user roles. Space link control plane interactions enabling user service request and dispositioning behaviors, as well as behavioral interactions within the user mission data plane, occur in the Network Link Execution domain. Further decomposition of this domain is necessary to address questions associated with the allocation of these functions to space link resources. This is accomplished through standardized object-oriented SysML artifacts, including structural representations using block definition diagrams and internal block diagrams, and behavioral representations using state machine and sequence diagrams. As discussed previously, the high availability requirement of the control plane space link will likely drive solutions to a space relay resource capable of supporting multiple simultaneous UIS service instances, while the user mission data plane space link may be instantiated by space relay and direct-to-earth link resources with more limited availability. The architectural model provides the structural and behavioral framework within which these physical system tradeoffs occur, enabling rigor and specificity on trade criteria and ensuring traceability throughout the system hierarchy.

E. UIS Message Data Modeling

Data modeling is a critically important aspect of architecting information intensive systems, and often begins through the iterative process of developing an activity diagram [3]. Conceptual data objects represent the information content of the architecture, and are exchanged among the domains in the object flow of the activity diagram. Using the inheritance properties of object-oriented languages, these conceptual data objects may be grouped and abstracted into foundational classes that describe broad categories of data content. Common foundation classes in information-intensive systems include plans, tasks, reports and messages [3]. The logical data model is a level of abstraction below the conceptual level, and describes the structure of conceptual data objects in accordance with the inheritance, associations and other relationships defined previously. The user service request message provides an illustrative example of this concept.

A user service request message must specify the attributes of the requested space communications services. This message includes a header and a payload. The header information includes data about the message type and transaction. The payload information includes transaction setup and service request data. Figure 5 illustrates a preliminary user request message logical data structure block definition diagram for the request of a space link resource agnostic telemetry service event for downlinking a user specified data volume within a user specified deadline.

This logical data structure accommodates future protocol versions, fault management, error correction, authentication, confidentiality, integrity and non-repudiation mechanisms, identity information supporting thread management, archiving and retrieval, and quality of service information such as user mission data volume, urgency and the user specified deadline.

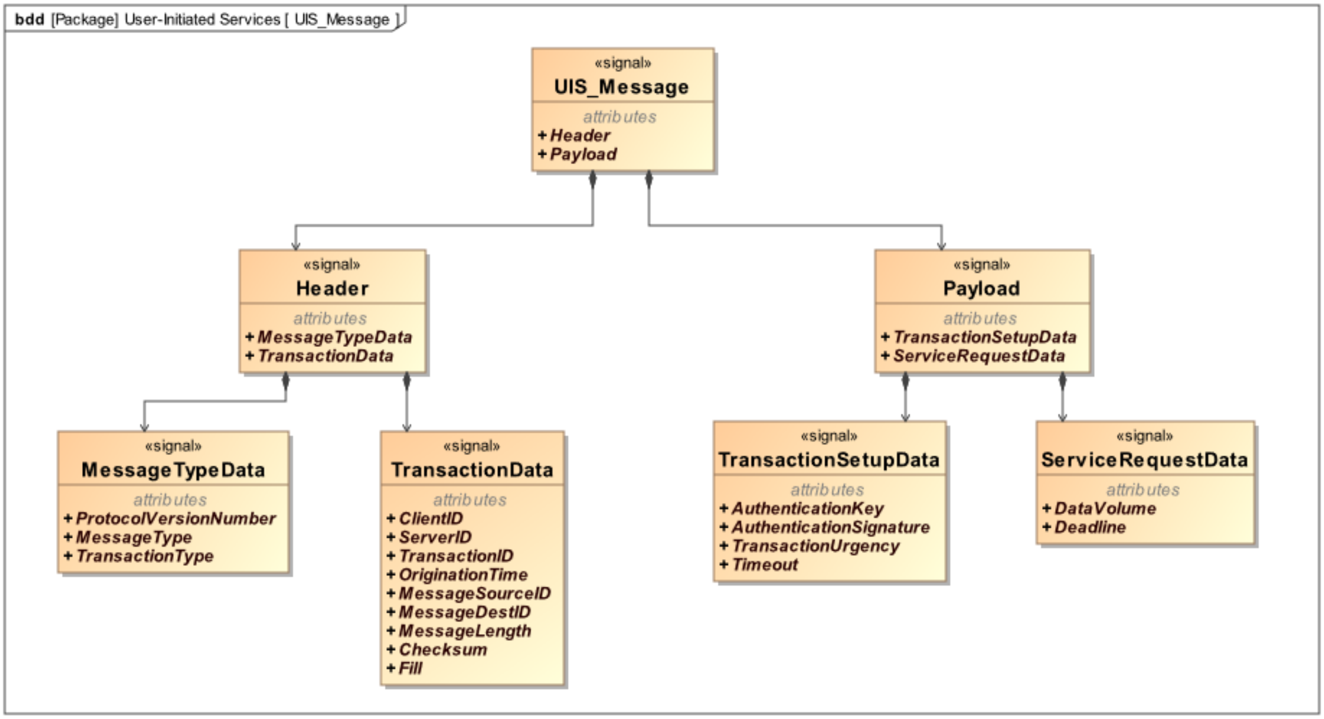

Figure 5: Logical data model of a UIS user service request message

III. Discussion

As documented in this paper, the use of a model-driven architecture methodology provides several advantages over the more common ad-hoc systems architecting approach. Standardized object-oriented languages originating in the software engineering discipline have been extended to better meet the needs of systems architecting. SysML artifacts, including requirements, structural and behavioral diagrams facilitate traceability, consistency and rigorous analysis, synthesis and evaluation at all levels of the system hierarchy.

UIS provisional capability requirements, relationships to traditional mission operations and space communications use cases, and the top-level UIS structural, behavioral and data models have been defined. Further UIS architectural specification, in accordance with the model-driven architecture methodology, is necessary to understand and evaluate alternative UIS protocol and space link control plane design trades.

In addition to the static object-oriented modeling presented in this paper, computationally executable architectural simulations are possible in SysML. In conjunction with more traditional engineering simulation packages, these executable models may be calibrated with experimental results from prototyping and demonstrations. This powerful approach allows induction of architectural performance and other characteristics, informing key trade-offs at various levels of the system hierarchy.

For UIS, key performance metrics and trade-offs may concern the minimum, maximum and expected user wait time elapsed between service invocation and execution under various conditions, the physical data volume of alternative UIS message data structures, the quantity of interactions to invoke, disposition and execute UIS service instances, as well as the control plane bandwidth and simultaneous service instance capacity.

Computational simulation methods may also be used to explore enterprise-level performance and emergent behaviors, such as those pertaining to end-to-end service disposition dynamics, including space-terrestrial network traffic analysis, space-terrestrial resource scheduling policy, and multi-criteria optimization trade-offs. Additional investigations may pertain to the impacts of federating resource event scheduling to multiple providers (e.g., U.S. government, international, and commercial), and the impacts of dynamic entry and exit of both users and provider resources due to weather, asset maintenance or other service constraints or enablers.

At each stage of development, insights from simulations, prototypes and demonstrations will be fed back into the UIS architecture model, resulting in an iterative, evidence-based, consistent, cohesive and traceable system evolution. This process is expected to maximize the likelihood of infusing UIS enabling technologies, and ultimately widespread adoption of user initiated services.

IV. Conclusion

NASA’s Space Mobile Network will involve dynamic and autonomous services analogous to services provided by current terrestrial wireless networks. A new service concept under this paradigm, known as User Initiated Services, extends traditional highly scripted mission operations and space communications use cases through infusion of software-enabled services across the user and space communications provider domains. UIS will enable more event-driven and collaborative mission operations concepts, reduced user space communications planning burden, and more efficient and effective provider network resource utilization.

To realize the UIS concept, a continuously available space link control plane must be defined and established to support bidirectional communications between the user and provider network. In addition, a new protocol must be defined specifying message data structures, machine-to-machine behavioral sequences and rules, all within a heterogeneous, distributed, and evolving system and environmental context.

This paper documents preliminary results from the application of model-driven architecture process to develop UIS, including provisional capability requirements, use cases, and the top-level system structural, behavioral and data models. Such an approach is necessary to ensure systematic investigation of several open questions concerning the efficiency, robustness, interoperability, scalability and security of the control plane space link and UIS protocol. Future work will focus on continued specification of the UIS architecture model, including iterative refinements based on analysis, synthesis and evaluation of results from computational simulation, virtual and physical prototyping, and demonstration activities.

Acknowledgment

C.R. thanks Jacob Burke and Mark Sinkiat for their contributions to the UIS logical data model, and the Cognitive Communications Project at NASA Glenn Research Center for their constructive review of concepts presented in this manuscript. C.R. thanks Seema Vithlani and Carolyn Crichton for technical editing and graphics support in preparation of this manuscript.

References

[1] D. J. Israel, G. W. Heckler, and R. J. Menrad, *Space Mobile Network: A Near Earth Communications and Navigation Architecture*, presented at the IEEE Aerospace Conference, Big Sky, MT, United States, 2016.

[2] T. Newton, C. Roberts, G. Fletcher, and D. Rossiter, *An Evaluation of Terrestrial Wireless Network Modeling Approaches for the Space Mobile Network*, presented at the International Telemetering Conference, Las Vegas, NV, United States, 2017.

[3] J. M. Borky, *Architecting Information-Intensive Aerospace Systems.* Colorado State University, Ft. Collins, CO, September 2016.

[4] R. Hartmann ed., *A World in Motion – Systems Engineering Vision 2025*. International Council on Systems Engineering, October 2014.

[5] E. Haleplidis, K. Pentikousis, S. Denazis, et al., *Software-Defined Networking (SDN): Layers and architecture Terminology,* Internet Research Task Force (IRTF) Request for Comments: 7426, ISSN: 2070-1721, January 2015.