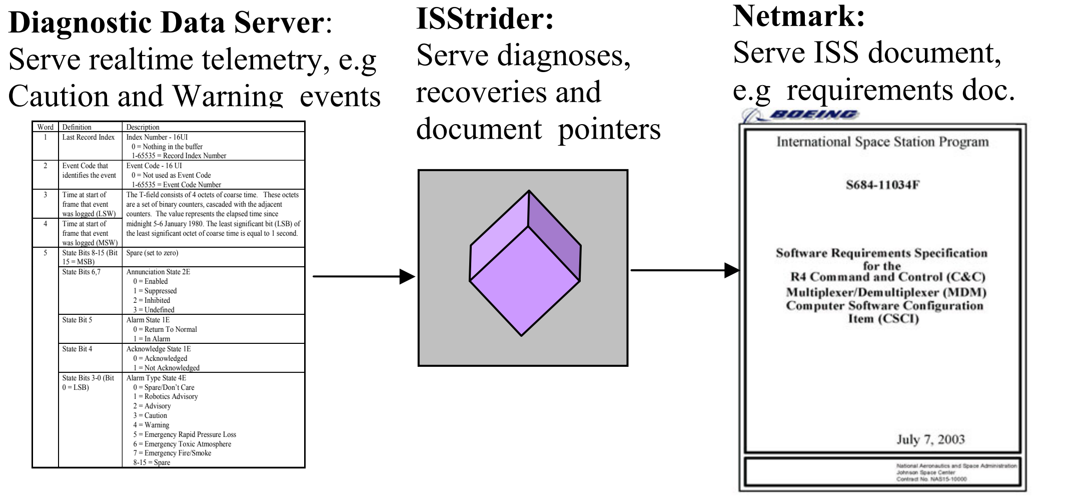
Realtme Knowledge Management (RKM) –
From an International Space Station (ISS) Point of View

Peter I. Robinson (NASA Ames/QSS Group) probinson@mail.arc.nasa.gov
William McDermott (NASA Ames) B.McDermott@nasa.gov
Rick Alena (NASA Ames) Richard.L.Alena@nasa.gov

Introduction

When problems occur on spacecraft, onboard and ground-based teams begin to analyze the data. Timely access to relevant domain knowledge can mean the difference between a nominal, degraded or failed mission. For a spacecraft as large as the ISS, the documentation is well over 1 million pages not including reference materials in the form of spreadsheets, diagrams, databases, source code, test results and other domain-specific forms of knowledge. This is due in large part to the fact that ISS is being developed in stages, over many years, with many international collaborators. We believe that tools which can provide relevant and timely access to these documents and reference materials will decrease anomaly resolution time with concurrent increases in safety.


Diagnostic Data Server:
Serve realtime telemetry, e.g Caution and Warning events
ISStrider:
Serve diagnoses, recoveries and document pointers
Netmark:
Serve ISS document, e.g requirements doc.
BOEING

Word	Definition	Description
1	Last Record Index	Index Number - 16UI 0 = Nothing in the buffer 1-65535 = Record Index Number
2	Event Code that identifies the event	Event Code - 16 UI 0 = Not used as Event Code 1-65535 = Event Code Number
3	Time at start of frame that event was logged (LSW)	The T-field consists of 4 octets of coarse time. These octets are a set of binary counters, cascaded with the adjacent counters. The value represents the elapsed time since midnight 5-6 January 1980. The least significant bit (LSB) of the least significant octet of coarse time is equal to 1 second.
4	Time at start of frame that event was logged (MSW)	
5	State Bits 8-15 (Bit 15 = MSB)	Spare (set to zero)
	State Bits 6,7	Annunciation State 2E 0 = Enabled 1 = Suppressed 2 = Inhibited 3 = Undefined
	State Bit 5	Alarm State 1E 0 = Return To Normal 1 = In Alarm
	State Bit 4	Acknowledge State 1E 0 = Acknowledged 1 = Not Acknowledged
	State Bits 3-0 (Bit 0 = LSB)	Alarm Type State 4E 0 = Spare/Don't Care 1 = Robotics Advisory 2 = Advisory 3 = Caution 4 = Warning 5 = Emergency Rapid Pressure Loss 6 = Emergency Toxic Atmosphere 7 = Emergency Fire/Smoke 8-15 = Spare

International Space Station Program

S684-11034F

Software Requirements Specification
for the
R4 Command and Control (C&C)
Multiplexer/Demultiplexer (MDM)
Computer Software Configuration
Item (CSCI)

July 7, 2003

National Aeronautics and Space Administration
Johnson Space Center
Contract No. NAS15-10000

Figure 1: Realtime Knowledge Management Example Realtime telemetry determines diagnostic state which is mapped to documents, e.g. map Caution and Warning event [10] to ISS Requirements documentation.

We are developing automated methods to provide realtime access to spacecraft domain knowledge relevant to current spacecraft operational state. The method is based upon analyzing state-transition signatures in the telemetry stream. A key insight is that documentation relevant to a specific failure mode or operational state is related to the structure and function of spacecraft systems. This means that diagnostic dependency and state models can provide a roadmap for effective documentation navigation and presentation.

Diagnostic models consume the telemetry and derive a high-level state description of the spacecraft. Each potential spacecraft state description is matched against the predictions of models which were developed from information found in the pages and sections in the relevant ISS documentation and reference materials. By annotating each model fragment with the appropriate domain knowledge sources, from which it was derived, we can develop a system which automatically selects those documents representing the domain knowledge encapsulated by the models which compute the current spacecraft state. In this manner, when the spacecraft state changes, the relevant documentation context and presentation will change as well.

Architecture

The Realtime Knowledge Management (RKM) tool is developed as an integration of three existing software tools: 1) the Diagnostic Data Server [1], a telemetry server which provides a temporally organized set of telemetry, logs and data-dumps of the ISS over a selected time window 2) the Netmark [2] document database which indexes documents both on context (document token) and content (ISS token); and 3) the ISStrider [3] model-based diagnostic tool [3] developed using L2 [6]which models both the hardware and software aspects of the ISS Command and Data Handling (C&DH) system. These three technologies are part of a set of engineering support tools which will be deployed at NASA JSC in the next two years.

The flow of information for the RKM architecture follows the numbers in Figure 1: 1) Telemetry queries are defined by the diagnostic tool ISStrider (or by the human operator), 2) telemetry is consumed by ISStrider which produces 3) diagnoses (in the future recoveries as well) and 4) document queries. The document queries to Netmark produce 5) relevant documents. The telemetry, diagnoses and relevant documents are all integrated in user-defined GUIs for rapid access by ISS flight engineers.

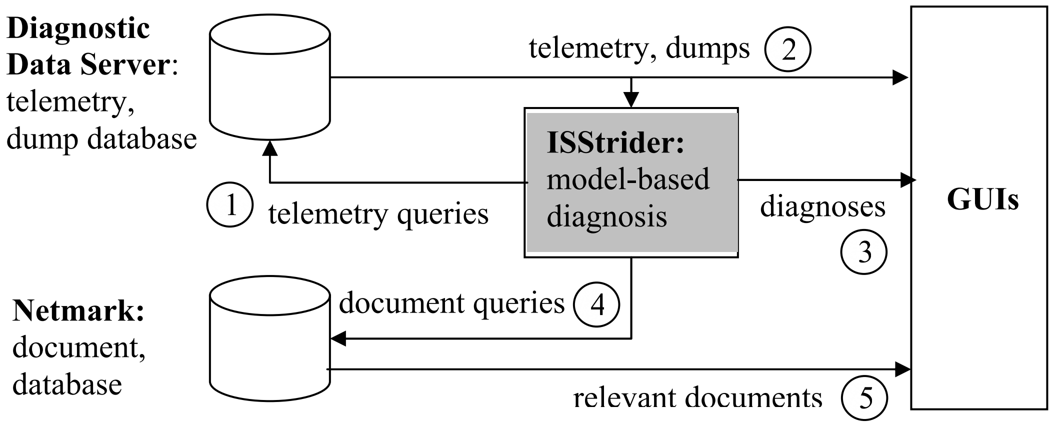

Diagnostic Data Server: telemetry, dump database
telemetry, dumps 2
ISStrider: model-based diagnosis
1 telemetry queries
diagnoses 3
GUIs
Netmark: document, database
document queries 4
relevant documents 5

Figure 2: Realtime Knowledge Management (RKM) architecture which integrates DDS [1], Netmark [2] and ISStrider[3].

Example

We demonstrate an example of the RKM architecture in Figure 1., on an example from the ISS C&DH domain. The DDS serves a realtime Caution and Warning event to ISStrider which maps this event to a portion of the ISS software requirements specification (SRS) [4]. This example defines a direct mapping between telemetry and corresponding documentation (see Implementation Section for details).

In general the mapping from a caution and warning event to domain documentation is non-trivial and could require methods which reason about hidden state. For example, the documents relevant to interpret a bus caution and warning event (e.g CW #5392), include a hardware schematic, Bus Address Assignments Table, and the bus profile. Or for example the documents relevant to interpret a computer (MDM) failure caution and warning event (e.g. CW #5104) will require all document related to both the failed execution of the hardware as well as the software. This would include documents for both the hardware and software schematics, requirements specifications, documents which define the Built-in Tests (BIT) and Power-On Self-Tests (POST) tests, documents which define the rate monotonic scheduler and its tasks, and of course the source code itself [8,9,10,11,12,13,14,15,16,17].

We are developing additional model-based methods based on internal state variables to allow for indexing documents based upon the internal structure and topology of the subsystem domain. Below we define the grammar required to implement the RKM system. It will also apply to state-less dependency models such as TEAMS as well. For TEAMS additional elements will be added to define tests and the source knowledge required to model each test.

Implementation

The implementation is partitioned into three areas, 1) the grammar to describe the ISS document dependencies and the interface into the Netmark document store/database 2) the grammar to describe the ISStrider model-based diagnosis models and its links to the document dependencies. Once the document dependencies are defined, we can annotate our models with the source documents. Our models are based upon the ISStrider system, which utilizes the L2[6] model-based diagnosis system. Our approach is also applicable for many other diagnostic systems including the state-less structural dependency TEAMS[7] tool. For TEAMS focus would be on defining the tests, while for L2 focus is on defining the states of the models and its transitions. This is accomplished in L2 through the use of a model development interface which allows the user to annotate each model fragment with documentation.

Document dependency grammar. To develop a system which can capture ISS specific document types, we define a document dependency grammar. The grammar serves to map the ISS document types to the Netmark tags for both context and content. **Context** is defined as the sections of documents which are important to index to. We have identified a preliminary set of ISS document contexts: 1) document name 2) section 3) page-slide 4) figure 5) table 6) web- link 7) workbook 8) database 9)) source DDS dataset. **Content** is defined with respect to domain specific features of the ISS domains. We have identified a preliminary set of ISS document contents: 1) requirement number 2) PUI 3) CSCI (software program identifier) 4) part number 5) procedure number.

<doc_dependency>::= <context>+<content>

<context>::=<document_name>|<section> | <page-slide> | <figure> |<table>|
<web- link>|<workbook> |<database-table> | <DDS dataset>|…
<content>::= <requirement number>$^{+}$ |<PUI> $^{+|}$<CSCI>$^{+}$ |<Part Number>$^{+}$ |
<procedure number>$^{+}$|<CSCI>$_{+}$

Model-based diagnosis grammar. An ISStrider Livingstone model is made up of a set components which are connected via a set of connections. At each level, including the model level, we can annotate with documentation dependencies which source documents to model constructs.

<model>::=<component>+ (<connection>)* <doc_dependency>*

Each component is defined at its boundaries by a finite set of ports, each of domain specific types. Internal to each component, there is a finite-state machine which itself is made up of a finite set of states. Directed and undirected transitions between the states are defined. Executing the transitions in the forward direction is used to perform state estimation, while executing the state transitions in the reverse direction is used to perform regulation (recovery).

<component>::=<port>+ <finite_ state_machine> <doc_dependency>*
<finite_state_machine>::=<state>+ <transition>* <doc_dependency>*
<transition>::=<guard> <cost> <state> <state> <transition_type><doc_dependency>*
<state>::=<logical expression> <doc_dependency>*

Each connection between components are defined with a type, port-name and connection specific documentation dependencies. Each port is defined by a type which identifies the port as an observation, command or internal port as well the possible values which the port can take. Each value is an element of the set of all possible values: $\{..v_i,v_j..\}$.

<connection> ::=<type> <port> <doc_dependency>*.
<port>::=<port-type> <value-type> <doc_dependency>
<port-type>::=[observation | command | internal] <doc_dependency>*
<**value-type>**::=[(discrete|continuous) <value>+] <doc_dependency>*
<value> ::= $\{..v_i,v_j..\}$ <dependency>*

Discussion

At any point in time, there exists a ranked set of diagnoses from the ISStrider system. Each of these diagnoses provide a state vector over the state of each component in the system. By identifying the active model fragments, the RKM system, given the current diagnosis state, can automatically present the user the documentation entailed by the diagnosis.

Since the documentation is being driven by the telemetry, often it will occur that many sets of documents are available. For example, our system will be able to automaticallyt find the intersection of two requirements (raised by some event): requirement(x.x.x.x) and requirement(y.y.y.y). First by a **context search** on requirement(x.x.x.x) in Netmark. Netmark we return a set of pages over a range [$lb_1..ub_1$]. Then a **content search** on requirement(y.y.y.y) over the range of pages [$lb_1..ub_1$]. This will return a more refined set of pages [lb_2 .. ub_2] such that $lb_1 <= lb_2$ & $lb_2 <= ub_1$.

References

[1] Fletcher, D. P., Alena, R. "A Scalable, Out-of-Band Diagnostics Architecture for International Space Station Systems Support", IEEE Aero 2003
[2] Maluf, David A. and Tran, Peter B., "NETMARK: Adding Hierarchical Object to Relational Databases", Intelligent Systems Design and Applications (ISDA) 2003, Tulsa, Oklahoma, Conference Proceedings, 2003.
[3] P. Robinson, M. Shirley, D. Fletcher, R. Alena, D. Duncavage, C. Lee, "Applying Model-Based Reasoning to the FDIR of the Command & Data Handling Subsystem of the International Space Station," iSAIRAS 03 2003
[4] S684-11034F Boeing ISS Program "Software Requirements Specification for the R4 Command and Control (C&C) Multiplexor/Dimultiplexors (MDM) Computer Software Configuration Item (CSCI)", July 2003
[5] Schumann, J. Robinson P., [] or SUCCESS is Not Enough: Current Technology and Future Directions in Proof Presentation , Workshop on "Future Directions in Deduction", International Joint Conference of Automated Reasoning IJCAR 2001
[6] Kurien, J. , Nayak. P.P Back to the Future for Consistency-based Trajectory Tracking. AAAI-97
[7] Deb, S., Domagala, C.,Ghosal, S.,Patterson-Hine A., Alena , R. Remote Diagnosis of the International Space Station utilizing Telemetry Data SPIE April 2001
[8] D684-10500-04B CDH ADD Vols. 1-4.
[9] C&W Fault Trees: http://hsi.jsc.nasa.gov/Cwad/
[10] Owens, D. , Dempsey, R .Caution and Warning Systems Brief NASA JSC DF25 CDH 06/02
[11] S684-11032 ISS Software Requirements: R2,, Command and Control (C&C) MDM (CSCI) 11/01
[12]D684-11106-01 ISS Command and Control Software , Software User's Manual R2 10/01
[13] D684-10056-01K ISS Prime Contractor Software Standards and Procedures Specification 12/00
[14] D684-10177-01F ISS Mission Build Facility Standard Output Definition
[15]NAS15-10000 Software Rqmts Spec for the MDM Boot and Diagnostics Firmware of C&DH
[16]D684-11111-01 ISS Command and Control Software, Software Top Level Design Document R3
[17] JSC 28721 User's Guide for the Portable Computer System (PCS) ISS MOD 5/2001
[18] D684-11379-01 MADE Design Document 3/02