Architectural Analysis of Complex Evolving Systems of Systems

SARP 2007 - 2009 Initiative -- End-of-year Summary of Accomplishments

Third and Final Year

Principal Investigator (PI): Dr. Mikael Lindvall, FC-MD

NASA POC: Sally Godfrey, GSFC

Team members: William C. Stratton[1], Deane E. Sibol[1], Dr. Arnab Ray[2], Chris Ackermann[2], Lyly Yonkwa[2], Dharma Ganesan[2]

December 2009

1. Johns Hopkins University Applied Physics Laboratory Space Department Ground Applications Group (APL)

2. Fraunhofer Center for Experimental Software Engineering Maryland (FC-MD)

REVISION HISTORY

Date	Version	Status	Reason for Change
12/12/2009	0.9	Initial version	New document
12/19/2009	1.0	Submitted	Reviewed, updated

TABLE OF CONTENTS

Table of Contents

1.0 Summary and Introduction

1.1 Summary Statement

The **Architectural Analysis of Complex Evolving Systems of Systems** research project was a three year long research initiative originally scheduled to start in January 2007, but was delayed until May 2007 due to contracting constraints. In the summer of 2007, the project ramped up to the planned level of staffing, and continued as planned up through when the project ended 12/19/2009. This report represents the research conducted during year three, the final year.

The goal of this collaborative project between FC-MD, APL, and GSFC and supported by NASA IV&V Software Assurance Research Program (SARP), was to develop a tool, Dynamic SAVE, or Dyn-SAVE for short, for analyzing architectures of systems of systems.

The project team was comprised of the principal investigator (PI) from FC-MD and four other FC-MD scientists (part time) and several FC-MD students (full time), as well as, two APL software architects (part time), and one NASA POC (part time).

The PI and FC-MD scientists together with APL architects were responsible for requirements analysis, and for applying and evaluating the Dyn-SAVE tool and method. The PI and a group of FC-MD scientists were responsible for improving the method and conducting outreach activities, while another group of FC-MD scientists were responsible for development and improvement of the tool. Oversight and reporting was conducted by the PI and NASA POC.

The project team produced many results including several prototypes of the Dyn-SAVE tool and method, several case studies documenting how the tool and method was applied to APL's software systems, and several published papers in highly respected conferences and journals.

Dyn-SAVE as developed and enhanced throughout this research period, is a software tool intended for software developers and architects, software integration testers, and persons who need to analyze software systems from the point of view of how it communicates with other systems. Using the tool, the user specifies the planned communication behavior of the system modeled as a sequence diagram. The user then captures and imports the actual communication behavior of the system, which is then converted and visualized as a sequence diagram by Dyn-SAVE. After mapping the planned to the actual and specifying parameter and timing constraints, Dyn-SAVE detects and highlights deviations between the planned and the actual behavior.

Requirements based on the need to analyze two inter-system communication protocols that are representative of protocols used in the Aerospace industry have been specified. The protocols are related: APL's Common Ground System (CGS) as used in the MErcury Surface, Space ENvironment, GEochemistry, and Ranging (MESSENGER) and the Radiation Belt Space Probes (RBSP) missions. The analyzed communications were implementations of the Telemetry protocol and the CCSDS – File Delivery Protocol (CFDP) protocol. Based on these requirements, three prototypes of Dyn-SAVE were developed and applied to these protocols. The application of Dyn-SAVE to these protocols resulted in the detection of several issues. Dyn-SAVE was also applied to several Testbeds that have previously been used for experimentation earlier on this project, as well as, to other protocols and logs for testing its broader applicability. For example, Dyn-SAVE was used to analyze 1) the communication pattern between a web browser and a web server, 2) the system log of a computer in order to detect off-nominal computer shut-down behavior, and 3) the actual test cases of NASA Goddard's Core Flight System (CFS) and automatically generated test cases in order to determine the overlap between the two sets of test cases. In all cases, Dyn-SAVE assisted in providing insightful conclusions about each of the cases identified above.

In addition to the applied research and transfer work conducted at FC-MD, Chris Ackermann, a FC-MD supported PhD student from the University of Maryland's (UMD) Computer Science (CS) department is using this project as a key component of his PhD thesis. This has allowed him to conduct research on fundamental algorithms that have been used in this project. Thus we have been able to link university research (UMD) with applied research and tech transfer (FC-MD), to Government and Industry using feedback (NASA and APL) in a very successful and efficient way that has been a win-win for everyone involved.

In the future, as an extension of this research initiative, there is a plan to apply Dyn-SAVE to several additional NASA systems (CFS, GSFC Mission Services Evolution Center (GMSEC), and the Mars Science Laboratory- MSL) through a new SARP project, "Architecture Analysis of Dynamically Configurable Systems" that started in October 2009.

1.1.1 Evolution of Prototypes

The first prototype, developed in 2007, had some basic capabilities for comparing planned sequence diagrams with traces captured from running software systems and could detect some additional, missing, and incorrectly ordered messages. The user interface of this first prototype wasn't as polished, which is not uncommon for an early prototype, and needed to be tried and tested by knowledgeable and friendly users. This type of feedback was gathered to improve and enhance the Dyn-SAVE tool throughout the research initiative. The first prototype was applied to our TSAFE testbed, as well as, to the Telemetry protocol used in CGS.

The second prototype, developed in 2008, is based on specifications of more advanced dynamic behavior and algorithms for their evaluation. The second prototype had capabilities for comparing planned sequence diagrams, specifying order of messages, parameters and values, as well as, timing constraints, with traces captured from a running software system. The user interface was still unpolished and the tool still needed to be used and tried by a knowledgeable and friendly users. The second prototype was applied to the Telemetry protocol used in CGS. An even more advanced version of Dyn-SAVE, but still an early version of the prototype was applied to the communication based on CFDP between satellites and the Mission Operation Center (MOC); CGS is a component of the MOC.

The third prototype, a more refined and user friendly version, was developed over the past year. This prototype version can model and analyze more advanced protocols with less effort. For example, it has an advanced search mechanism that allows the user to search for the most important pieces of information within the systems that were studied. This is important because typical traces are long and are typically difficult to navigate. The latest prototype also has features that facilitate the analysis, such as automatic name-based mapping, and a connection to APL's Mission Operation Center (MOC) facilitating the analysis of data from their missions. We also created a set of videos that guide a novice user on how to use Dyn-SAVE.

Figure 1~~Figure 1~~ illustrates how Dyn-SAVE was applied in the context of the MOC for MESSENGER, which is currently orbiting Mercury.

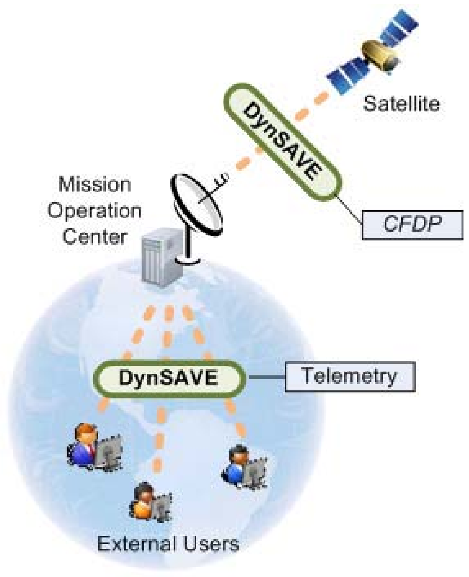


Figure 1. Dyn-SAVE in perspective. The satellite communicates with the Mission Operation Center (MOC) to transfer data using CCSDS – File Delivery Protocol (CFDP) (e.g. captured images) into a permanent storage on earth. External users can connect to the MOC via client software and access the data using the Telemetry protocol.

1.2 Project Goal

"The goal of the SARP project *Architectural Analysis of Complex Evolving Systems of Systems* is to research and develop a tool that will allow for architecture analysis of dynamic data captured from a software system during run-time. The new tool, *Dyn-SAVE*, will build upon and extend the already existing Fraunhofer *Software Architecture Visualization and Evaluation (SAVE)* tool, which allows for static analysis of a software system."

1.2.1 Project Objectives

The original objectives by year were:

- ❖ "Objective Year 1: Develop a first prototype version of the new Dyn-SAVE tool and apply it to APL's Common Ground System (CGS)."
- ❖ "Objective Year 2: Develop a second prototype version of the new Dyn-SAVE tool that handles problems related to Flight Systems. Apply the tool to APL's Flight System Software."
- ❖ "Objective Year 3: Develop a third prototype version of the new tool that handles larger systems of systems (SoS); apply it to CGS and Flight System Software together. Utilize the experience gained from over the three year period to improve the tool and package it so that it can be used by a broader set of users."

Comment: The objective for year 1 has been met. The objectives for year 2 and 3 have been modified to accommodate APL's actual needs and availability of APL testbeds.

The objective for year 2 was changed to:

- ❖ Objective Year 2: Develop a prototype of Dyn-SAVE and apply it to more complex scenarios of communication between the MOC and its various Clients, as well as, to the communication link between Satellites and the MOC.

The objective for year 3 was changed to:

- ❖ Objective Year 3: Develop a third prototype version of the new tool that handles larger SoS and can be applied to live data captured in the MOC. Utilize the experience gained from over the three year period to improve the tool and package it so that it can be used by a broader set of users.

The modified objectives for year 2 and 3 have been met.

1.2.2 Success Measures

The original success measures by year were:

- ❖ "End of year 1: The tool can successfully compare and identify deviations between planned and actual dynamic profiles, as well as, between two actual dynamic profiles; the tool can identify deviations between dynamic profiles based on test cases."

- ❖ "End of year 2: The tool can successfully compare and identify deviations between the dynamic profile of a system modeled in Simulink and the dynamic profile of that code that is automatically generated from that model."

- ❖ "End of year 3: The tool can successfully compare and identify deviations between planned and actual, as well as, between two actual dynamic profiles, in a combined fashion involving SoS developed in Simulink, generated code, and/or C/C++."

"The overarching criteria for this research initiative are that in the end Dyn-SAVE will make it easier and faster to detect problems related to dynamic profiles than conducting the analysis by conventional testing and analysis techniques. Detection of these problems today occurs usually during integration or acceptance testing or worse - the end-user reports an anomaly. Detecting such problems today is very difficult and time-consuming."

"The following measures will be used to evaluate the successfulness of the final version of Dyn-SAVE:"

- ❖ "Reduced time to detect problems related to dynamic profiles when using the tool instead of conventional testing and analysis techniques"

- ❖ "Reduced number of related problems related to dynamic profiles when using the tool instead of conventional testing and analysis techniques"

- ❖ "Ability to detect problems related to dynamic profile when using the tool that would be impossible to detect with conventional testing and analysis techniques"

Comments:

- After discussions with APL in 2007 regarding the most important problems related to dynamic architectures, the criteria was changed slightly for year 2 and 3. Instead of focusing on Simulink, we decided to focus on other aspects that are important to systems of systems such as modeling and evaluating timing issues, as well as, more advanced architecture rules and more complex communications.

- The applications of Dyn-SAVE to problems at APL have so far showed promising results. Analyzing communications' traces from inter-system communication using Dyn-SAVE is easier and takes less time than before. In addition, issues and previously undetected problems were detected using Dyn-SAVE. For example, our analysis showed that some configurations of the CFDP implementation deviate from the specified behavior in such a way that valuable and costly space network bandwidth would otherwise be wasted.
- New search features in Dyn-SAVE even more facilitate finding defects thus reducing the time to inspect evaluated results.

1.3 Working Approach

In order to research and develop a successful tool, it is important to have direct access to potential users, records of historical problems, and, in this case, a real System of Systems (SoS) to understand the needs, apply, and test the technology. For this project, Fraunhofer partnered with APL to ensure such access. APL's software systems are used as test-beds and APL software architects represent users of the technology. Fraunhofer scientists analyze the requirements, conduct related research, and develop and apply the software tools.

In order to experiment with and evaluate the new technology, we also used Fraunhofer's software test-bed called TSAFE (Tactical Separation Assisted Flight Environment).

More specifically, we followed the process steps enumerated below and iterated multiple times (at least three):

1. Collected concrete examples from APL
2. Modeled planned behavior
3. Used specification from ICD
4. Captured actual traces
5. Manually generated client requests and observed Server responses
6. Allowed "real" clients to perform certain tasks (requests), observed clients responses to server responses, and detected unintended use of server services
7. Identified common patterns

Now that the new technology is reasonably mature and as part of our outreach for this research initiative, we started contacting other NASA projects to help us further evaluate the new technology by sharing a demo of the tool and soliciting their feedback. We also solicited projects to participate in a research infusion project.

The NASA projects/centers that we have contacted are:

- The Core Flight System (GSFC) – has been supported by this project and is supported as part of the new SARP project
- The (GSFC) Mission Services Evolution Center (GMSEC) – is supported as part of the new SARP project

- The Mars Science Laboratory, JPL – plan to support the new SARP project; first visit to JPL was 12/7-9, 2009.
- CLARREO, LARC – initial contact and presentation made summer 2009, future support TBD.
- Safety analysis project, LARC – initial contact and presentation made December 2009, future support unclear.

In addition to an applied research project involving future technology users, this research initiative was done in close collaboration with the University of Maryland, in particular with Mr. Chris Ackermann, who is doing his PhD on this topic. This collaboration has allowed us to conduct very advanced research on how to use machine learning and related technologies for analyzing software runtime information. These results were published at prestigious conferences such as the International Symposium for Software Reliability Engineering (ISSRE). The publications are detailed below in Section 2.2.2.

2.0 Summary of Activities Performed

During the final year of this research initiative, we conducted the following activities:

2.1 Research and Development:

- Identified goals to be addressed by this research project during 2009
- Developed the third version of Dynamic SAVE in order to improve usability and add functionality that had been identified via user feedback
- Re-applied third version of Dynamic SAVE to Telemetry and CFDP protocols in order to verify that usability and functionality of the new version had improved. Examples are provided below.

2.2 Outreach and Reporting of Results:

2.2.1 Presentations

- Bill Stratton presented a paper at the 2009 IEEE Aerospace conference [1]
- Mikael Lindvall and Dharma Ganesan presented the approach and results (several times) to the GMSEC team at NASA Goddard
- Dharma Ganesan presented the approach and results (several times) to the CFS team at NASA Goddard
- Mikael Lindvall presented the approach to the CLARREO team at NASA Langley p(LARC)
- Dharma Ganesan presented a paper at the 2009 Software Product Line Conference (SPLC) [2]
- Mikael Lindvall presented results at NASA IV&V’s 2009 Software Assurance Symposium (SAS)
- Chris Ackermann presented a paper at the 2009 Working Conference on Reverse Engineering (WCRE) [3]
- Chris Ackermann presented a paper at the 2009 IEEE ISSRE conference [4]
- Dharma Ganesan presented at the Third IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT) Workshop: Software Reliability for Space Missions 2009 [5]
- Mikael Lindvall presented the approach to the MSL team at NASA JPL
- Mikael Lindvall presented the approach to a Safety analysis project at NASA LARC

2.2.2 Papers

[1] W. C. Stratton, D. E. Sibol, M. Lindvall, C. Ackermann, and S. Godfrey, "Developing an Approach for Analyzing and Verifying System Communication," The Aerospace conference, 2009.

[2] D. Ganesan, M. Lindvall, C. Ackermann, D. McComas, and M. Bartholomew, "Verifying Architectural Design Rules of the Flight Software Product Line," in *13th International Software Product Line Conference (SPLC)* 2009.

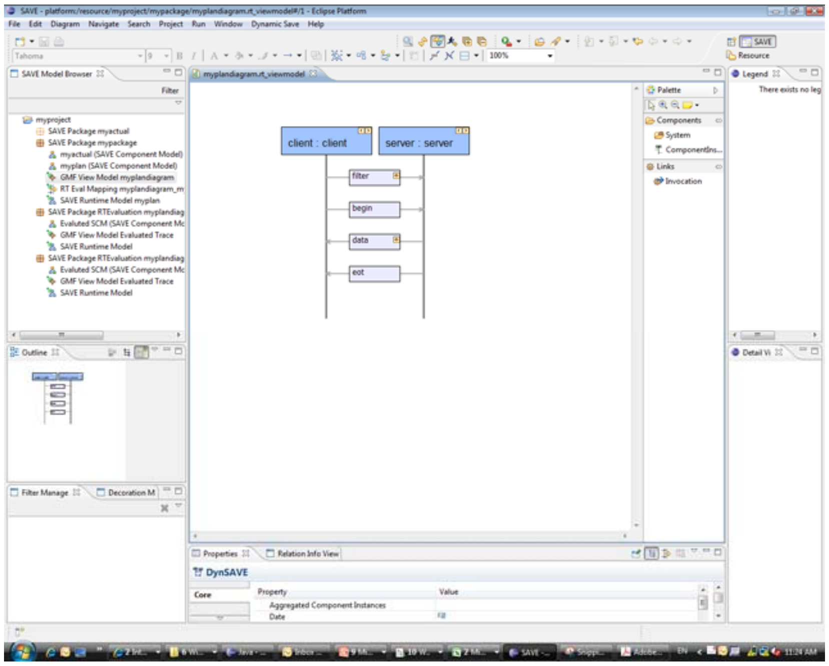
[3] C. Ackermann, M. Lindvall, and R. Cleaveland, "Recovering Views of Inter-System Interaction Behaviors," in *The Working Conference on Reverse Engineering (WCRE)* 2009.

[4] C. Ackermann, M. Lindvall, and R. Cleaveland, "Towards Behavioral Reflexion Models," in *The 20th annual International Symposium on Software Reliability Engineering (ISSRE 2009)* 2009.

[5] D. Ganesan, M. Lindvall, D. McComas, M. Bartholomew, M. Blau, and G. Cammarata, "Architecture-Centric Reliability Analysis of the NASA cFE/CFS," in The Third IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT) Workshop: Software Reliability for Space Missions 2009, M. Lindvall, W. C. Stratton, D. E. Sibol, C. Ackermann, M. Reid, D. Ganesan, D.
[6] McComas, M. Bartholomew, and S. Godfrey, "Connecting Research and Practice: An Experience Report on Research Infusion with Software Architecture Visualization and Evaluation (SAVE) - to appear in,"*NASA's journal on Innovations in Systems and Software Engineering*, 2009.

3.0 Using Dynamic SAVE – A Short Description

Dyn-SAVE is a software tool intended for software developers and architects, software integration testers, and persons who need to analyze software systems from the point of view of how it communicates with other systems. Dyn-SAVE is illustrated below using a series of screenshots.



SAVE - platform:/resource/myproject/mypackage/myplandiagram.rt_viewmodel#/1 - Eclipse Platform
File Edit Diagram Navigate Search Project Run Window Dynamic Save Help
SAVE Model Browser
myproject
SAVE Package myactual
SAVE Package mypackage
myactual (SAVE Component Model)
myplan (SAVE Component Model)
GMF View Model myplandiagram
RT Eval Mapping myplandiagram_m
SAVE Runtime Model myplan
SAVE Package RTEvaluation myplandiag
Evaluated SCM (SAVE Component Mc
GMF View Model Evaluated Trace
SAVE Runtime Model
myplandiagram.rt_viewmodel
client : client
server : server
filter
begin
data
eot
Palette
Components
System
ComponentIns...
Links
Invocation
Legend
There exists no leg
Outline
Detail Vi
Filter Manage
Decoration M
Properties
Relation Info View
DynSAVE
Core
Property
Value
Aggregated Component Instances
Date

Figure 2: A planned sequence diagram in Dyn-SAVE

The Dyn-SAVE user specifies the planned communication behavior of the system using a sequence diagram, see Figure 2~~Figure 2~~. The planned sequence diagram specifies that a set of Filters sent by the client to the server are followed by one BeginPlayback message. The server responds by sending a set of Data messages that are followed by one end-of-transaction (EOT) message. The implication of the sequences of messages as depicted in the planned sequence diagram is that that we expect all messages to appear in the specified order. In addition, we do not expect any other messages to appear in such a communication between a client and a server. We will use this planned sequence diagram in the examples below to illustrate how Dyn-SAVE can detect deviations between the planned sequence diagram and an actual sequence diagram.

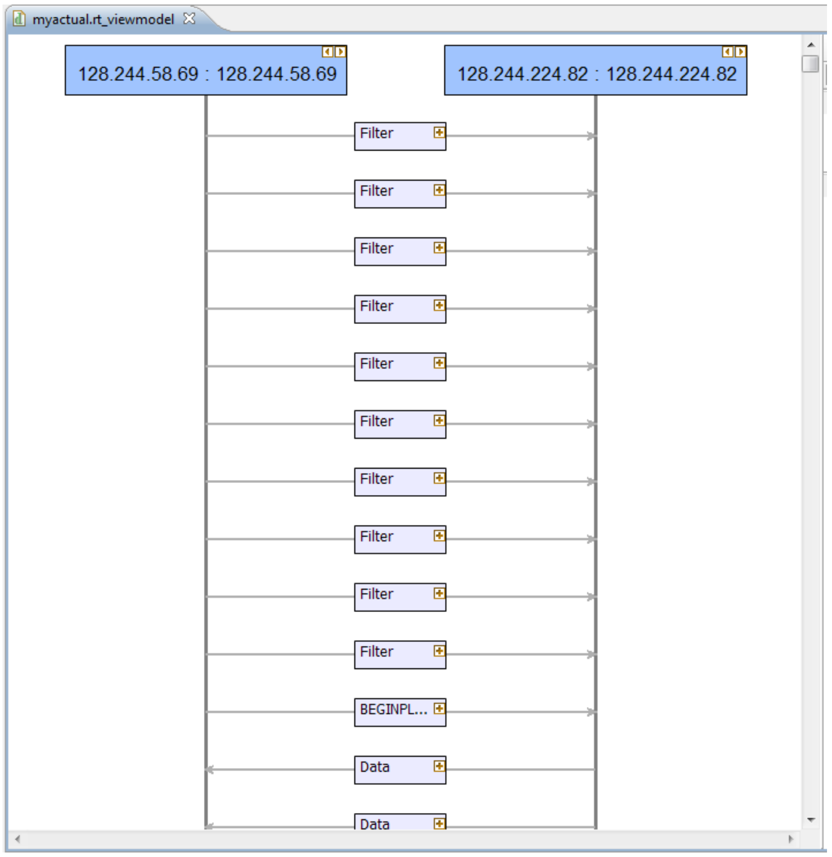


Figure 3: An actual communication behavior that has been imported into Dyn-SAVE and visualized as a sequence diagram.

The user then captures and imports the actual communication behavior of the system, which is also visualized as a sequence diagram, see **Figure 3**~~**Figure 3**~~. In this case, the communication behavior was captured using a network sniffer and stored as a comma separated file (CSV). Thereafter it was imported to Dyn-SAVE and the sequence diagram in **Figure 3**~~**Figure 3**~~ was automatically generated. The sequence diagram has 10 Filter messages and 1777 Data messages and is quite long. The numbers 128.244.58.69 (client) and 128.244.58.82 (server) are the IP-addresses of the client and the server.

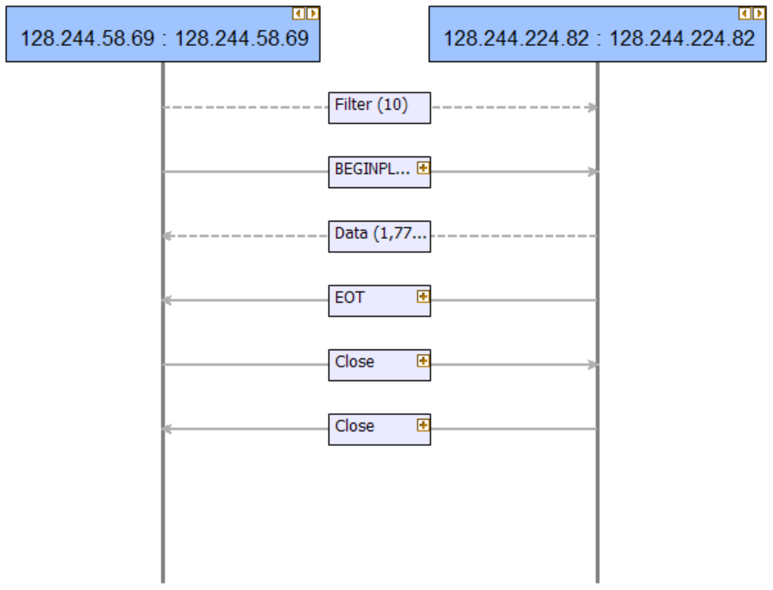


Figure 4: Aggregating messages to identify patterns

In order to facilitate identifying message patterns, the user can aggregate messages by, for example, the message names, see Figure 4~~Figure 4~~. In this diagram, we can see that the actual basically follows the same pattern as the planned described in Figure 2~~Figure 2~~., i.e. the communication starts with a number of Filter messages followed by one BeginPlayBack, from the client to the server. The server responds by sending a number of Data messages followed by one EOT message to the client. Finally, the communication ends with the client sending a close message and the server responding with another close message. Thus the only difference in this simple case is that that the two close messages in the actual are missing from the planned.

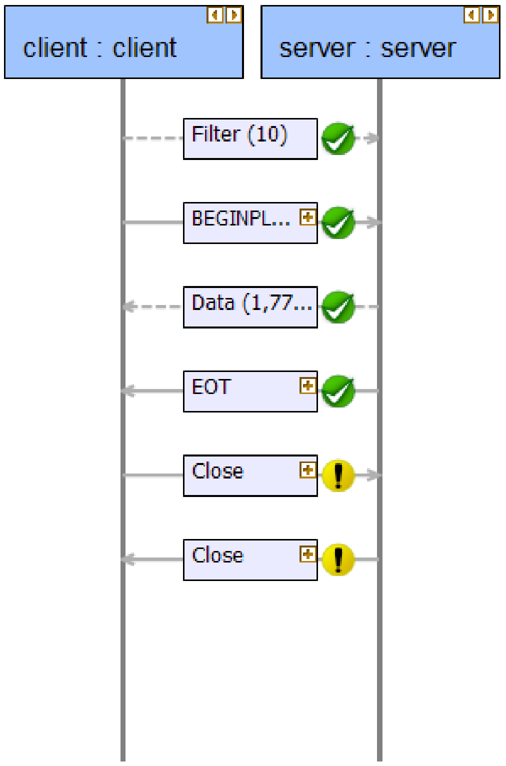


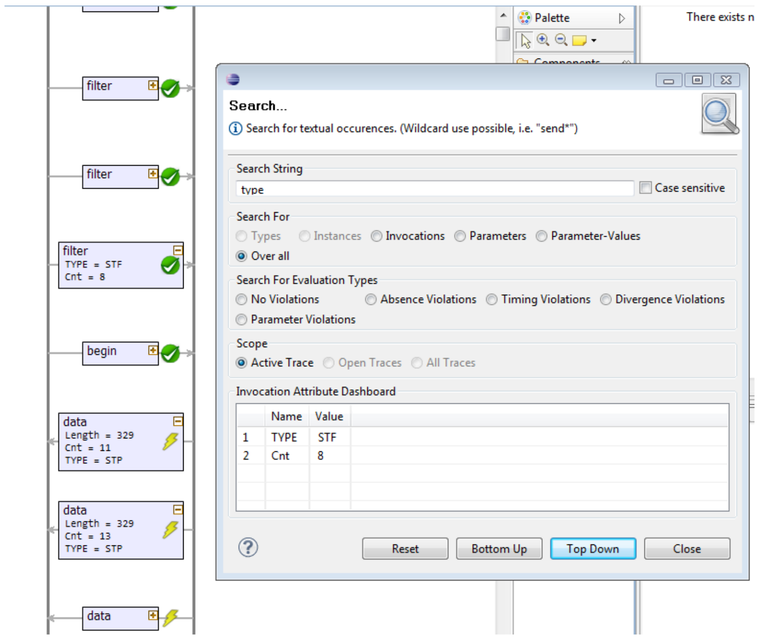
Figure 5: The evaluated diagram. Check marks indicate messages in the actual (Figure 3~~Figure 3~~ and Figure 4~~Figure 4~~) that correctly matches messages in the planned (Figure 2~~Figure 2~~). Exclamation marks indicate messages that occur in the actual but are missing from the planned.

Formatted: Font: 9 pt

Formatted: Font: 9 pt

Formatted: Font: 9 pt

After mapping the planned in **Figure 2~~Figure 2~~**.to the actual depicted in **Figure 3~~Figure 3~~** and aggregated in **Figure 4~~Figure 4~~**., In this example we show how Dyn-SAVE can detect and highlight deviations between the planned and the actual behavior, in this case the close messages. The result is stored as an evaluated diagram, see **Figure 5~~Figure 5~~**. This diagram can be exported and imported to, for example, PowerPoint or Word and be used in presentations and reports to illustrate critical deviations between the planned communication behavior and the actual.



Search...

Search for textual occurences. (Wildcard use possible, i.e. "send*")

Search String: type — [ ] Case sensitive

Search For: Types, Instances, Invocations, Parameters, Parameter-Values, [✓ Over all]

Search For Evaluation Types: No Violations, Absence Violations, Timing Violations, Divergence Violations, Parameter Violations

Scope: [✓ Active Trace], Open Traces, All Traces

Invocation Attribute Dashboard

	Name	Value
1	TYPE	STF
2	Cnt	8

Figure 6: Violation because Data.Type = "STP" and Filter.Type = "STF".

More advanced constraints can be defined in order to detect more subtle deviations than in the example above. For example, one can specify in the planned that a certain message parameter value has to match another certain parameter value. In this case (Figure 6~~Figure 6~~), we specified that there is a constraint between Filter message that has a parameter named “Type,” and Data messages that also have a parameter named “Type.” The constraint specifies that the value of the “Type” parameter carried by the Data message must be identical to the value of the “Type” parameter carried by the Filter message. Thus Filter.Type = Data.Type. If they are not identical then the Data message has deviated from the planned. In this example, **Figure 6**, a violation has been reported because Data.Type = "STP" and Filter.Type = "STF". Timing constraints can be defined in a similar way. For example, one can specify that the time between any to Data messages has to be less than a certain time, e.g. 1 second (not illustrated in the diagram). Such parameter and timing constraints can be used to detect deviations that are very difficult to detect.

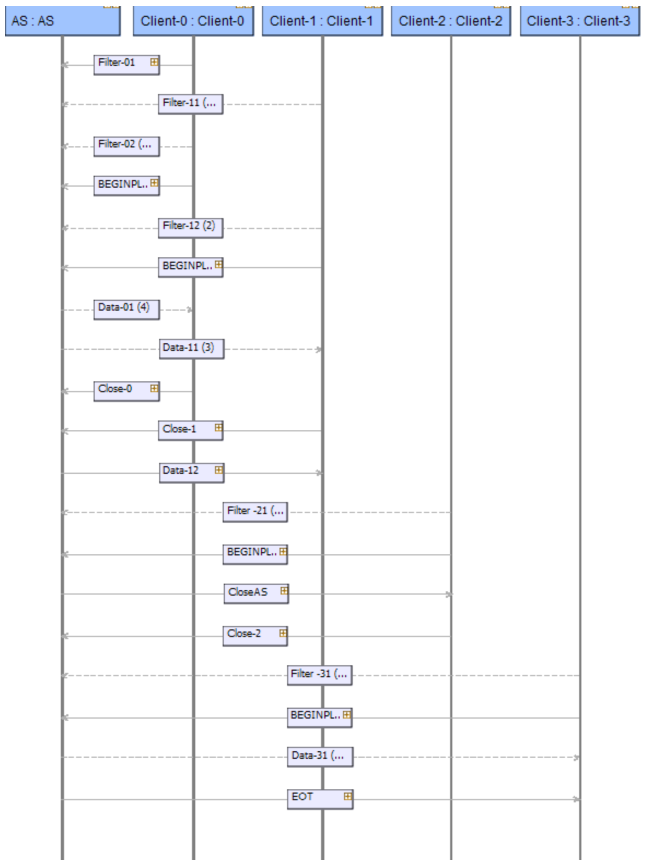


Figure 7: Several clients communicating with one server

Some support is also provided for handling several concurrent communication behaviors such as the communication denoted in Figure 7~~Figure 7~~. This example depicts four different clients communicating with the same server, each following the same pattern as the one specified in the planned in Figure 2~~Figure 2~~. One can see that each client first sends a series of Filter messages followed by one BeginPlaback. Three of the four clients receive a set of Data messages and then Close the connection while client-2 receives no Data messages but a Close message. Such a diagram can, for example, explain why the server seems to be unresponsive to one particular client. The reason for the slow response could be that the server is busy serving other clients that requested large amounts of data.

4.0 Conclusions and Outlook

4.1 Conclusions

The results from our studies show that problems in the communication between two systems can be detected by using sequence diagrams to model the planned or expected communication and by comparing the planned sequence to the actual sequence. The results also show that there are different kinds of problems and that they can be addressed by modeling the planned sequence using different levels of details. Sequencing problems, that is, messages that occur unexpectedly or out of order, can be detected by using high-level sequence diagrams, if there are not too many details. Content problems, problems which are related to the content of messages rather than to the order of messages, require a more detailed modeling approach. This detailed approach, which is based on parameter and timing constraints in combination with sequence diagrams, seems to be a feasible approach for addressing this type of problem.

4.2 Outlook

During this project, we have had the opportunity to present our research to a large number of practitioners and researchers. We have presented our work at several conferences devoted to Aerospace engineering with aerospace software engineers attending. We have visited several NASA centers and talked to several NASA projects and asked for their feedback and comments. We have presented our work at prestigious academic conferences in order to keep a close connection with the state of the art in research and to get feedback from the university-based research community. We have also applied the new technology to other problems (e.g. web server, computer event logs, test cases) that are outside of the immediate target problem in order to test its applicability to a broader set of problems.

Based on this varied feedback and our overall experience from this project, we have a very positive outlook for the Dyn-SAVE tool and related technologies. The problems Dyn-SAVE addresses are problems experienced by a large number of software engineers that develop complex systems. The systems they develop are quickly becoming too large and complex for anyone to fully understand their emerging behavior as the system evolves. In addition, these systems are often built in a way that they promote flexibility, but what is often gained in flexibility, is lost in the ability to easily analyze them. We have also experienced that the technology is generally applicable to situations where there is a need to check that a sequence of messages or events have certain properties.

Even though a number of very powerful verification technologies have been developed through university research, they are often very limited, especially in terms of usability. That is, they are typically very powerful, but there is a steep learning curve to be able to use them effectively. In addition, they do not often explicitly address the problems that software engineers are currently experiencing.

Our approach is to work closely with the practitioners in the field in order to closely understand their issues and problems. We base our research and development technologies on these insights to better ensure that these issues and problems are addressed. By applying and iterating technology throughout the development and research cycle, often repetitively, to real systems and continuously improving the technology based on the results, we are able to research and develop powerful and usable technology.

To end our outlook section, we want to mention that we are excited about the possible application of Dyn-SAVE to one of the most advanced current NASA projects. In December of 2009, we were invited to present our approach to the Mars Science Laboratory (MSL) team at NASA JPL. In a three-day workshop with the team, we defined two tasks to determine the feasibility of Dyn-SAVE to MSL. This feasibility analysis will be carried out as part of the new SARP project and will lead to new exciting adventures.

4.3 Acknowledgements

Lisa Montgomery and the SARP team at the NASA IV&V Center, Sally Godfrey GSFC, Dr. William Hartman Fraunhofer USA for supporting this work, and Myrna Regardie FC-MD for proof reading and editing this report.

Appendix A: Acronyms

APL	Applied Physics Laboratory
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS – File Delivery Protocol
CFS	Core Flight System
CGS	Common Ground System
COTS	Commercial Off the Shelf Software
CPU	Computer Processing Units
CSE	Column-Sum Edit
DNA	Deoxyribonucleic acid
Dyn-SAVE	Dynamic SAVE tool
EOT	End of Transmission
FC-MD	Fraunhofer Center for Experimental Software Engineering, Maryland
FSM	Finite State Machine
GMSEC	(GSFC) Mission Services Evolution Center
GSFC	Goddard Space Flight Center
ICD	Interface Control Documents
JPL	Jet Propulsion Laboratory
MIT	Massachusetts Institute of Technology
MESSENGER	MErcury Surface, Space ENvironment, GEochemistry, and Ranging
MSL	Mars Science Laboratory
NACK	Negative Acknowledgement
NASA	National Aeronautic and Space Administration
RBSP	Radiation Belt Space Probes
SARP	Software Assurance Research Program
SAVE	Software Architecture Visualization and Evaluation tool
SoS	System of Systems
STP	Supplemental Telemetry Packet
TP	Telemetry Packet
TPTP	Test and Performance Tools Platform
TSAFE	Tactical Separation Assisted Flight Environment
UI	User Interface
UML	Unified Modeling Language
V&V	Verification and Validation