Requirements Structure in a
System of Systems / Product-line Architecture

Geri Schneider Winters, July 1, 2007
# Table of Contents

Requirements for Projects Described by a System of Systems and Product Line Combined Architecture ................................................................................................................. 3

- System of Systems ........................................................................................................ 4
  - Documenting the System of Systems ........................................................................ 6
    - Handling Requirements and Interfaces .................................................................. 6
    - Focus on One Level of Detail .................................................................................. 7
    - Contain Changes ................................................................................................... 7

- Product Lines .................................................................................................................. 7

- Documenting a Product Line ......................................................................................... 8
  - Example Product Line Structure - RWGS .................................................................. 9

- Tracing requirements .................................................................................................... 12
  - Change scenarios and the trace model ....................................................................... 14

- Identification and Structure of the Use Cases ............................................................... 15

- Requirement relationships for system of systems combined with product line ............ 16

- Example: Requirements and Analysis Model for Combined System of Systems and Product Line ........................................................................................................................................... 17
  - RWGS Level Requirements ...................................................................................... 18
    - RWGS Activity Diagram .......................................................................................... 18
    - RWGS RD .............................................................................................................. 18
    - RWGS Use Case Realization: Run Training Exercise .............................................. 20

- Ground Vehicle Level Requirements .......................................................................... 21
  - Ground Vehicle Activity Diagram ............................................................................ 21
  - Ground Vehicle IRD .................................................................................................. 22
  - Ground Vehicle RD .................................................................................................. 22
  - Ground Vehicle Use Case Realization: Shut Down Vehicle ...................................... 24

- Command Interpreter Level Requirements .................................................................. 25
  - Command Interpreter Activity Diagram ..................................................................... 25
  - Command Interpreter IRD .......................................................................................... 25
  - Command Interpreter RD .......................................................................................... 26
  - Command Interpreter Use Case Realization: Report Heartbeat ................................. 28

- Final Thoughts ................................................................................................................ 28

- References and Websites .............................................................................................. 29
  - Geri’s websites .......................................................................................................... 29
Requirements for Projects Described by a System of Systems and Product Line Combined Architecture

Some projects are best described as a combination of a system of systems and product line architecture. This has implications for how the requirements are structured and traced.

This document describes how to handle requirements in a project described by a combination of a system of systems and product line architecture, using Robotics for an example. Simpler projects can use the same kind of techniques by suppressing some of the levels of systems or by removing the product line structures.

Some of you may be thinking this is just an academic paper. That is far from the truth. This paper was originally written for a specific client to use in a current project, and going forward, for all future projects in their line of business. My client gave me permission to share this information with others, but I have taken the precaution of changing the industry and the names of the elements in the examples, verifying my updated work with people in the Robotics field.

This is a very complex system that I am describing, and it is something that will evolve over time. The paper represents a way of structuring requirements in a systematic way that matches the structure of the system being described. One of the main reasons for designing the system this way is exactly so that it can be developed over time using small, well-defined, independent modules. I am not advocating writing all of the requirements for this system at the beginning (you can not do that anyway because no one knows how this will evolve). Rather, I am showing a way to develop the requirements over time, maintain relationships between the requirements of the different parts, and take advantage of natural commonalities in the system.
Many large projects can be described as a system of systems. This means that the parts of a system are themselves systems. Robotic War Games System (RWGS) is an excellent example of this structure. Please note: Figure 1 is an abstract of some of the major functionality of RSGS and is not meant to be a complete description of an actual system.
Figure 1 shows the overall system, and shows that a vehicle (in this case a ground vehicle) is also a system. The systems inside the ground vehicle can in turn be described by systems, as for example the sensor and propulsion systems, which in an actual system would be far more complex than indicated in this diagram.

In theory, you could pull out an entity such as Propulsion and replace it by a different propulsion unit, as long as it responds to the same commands as the original propulsion unit. (In real life, there are likely further subdivisions of entities such as Propulsion - for example power source, engine, and treads - which are the actual FRU – field replaceable units).

Another advantage of this structure is that by defining generic ground vehicle requirements, you could create a whole new ground vehicle that conforms to the interfaces defined for battlefield management and status, and put that vehicle in the field without changing any other software (or possibly just a change to a table in battlefield management for that new type of vehicle). This kind of system of systems defines plug-and-play capability at all levels of the RWGS.

The RWGS system as a whole is composed of a number of parts, such as Air Vehicles, Water Vehicles, Ground Vehicles, and Battlefield Management. There may be requirements at the level of RWGS which describe features that must be supplied by all parts of the system. Here you will often find overall performance requirements, security requirements, or error handling requirements. You may find overall software requirements such as exceptions that must be handled, patterns to be used, and specific formats of messages. You may find overall hardware requirements such as fail-over or emergency stop. Architecture at this level would describe the interactions between the major parts of the system.

Each major part of the RWGS system is in turn a system. You can consider Ground Vehicle to be a whole system. Ground Vehicle will be bound by the requirements at the RWGS level, including interfaces with other systems such as Battlefield Management. In addition, there are requirements that apply to all ground vehicles. You may find here requirements for self-testing, reporting, status, and exceptions to be handled. The Ground Vehicle system is in turn composed of parts such as Command Interpreter Software, Path Analysis Software, Clock, Propulsion, and Sensors. Architecture at the level of Ground Vehicle would describe the interactions between these major parts of the system.

Each major part of the Ground Vehicle system is in turn a system. You can consider that Propulsion is a whole system. Propulsion will be bound by the requirements at the Ground Vehicle level, including interfaces with other systems such as Path Analysis Software and Command Interpreter Software. In addition, there are requirements that apply to all of propulsion. You may find here requirements for response times, emergency stop, self-test, and fail-over. The Propulsion system is in turn composed of parts such as Propulsion Hardware and Propulsion Software. Architecture at the level of Propulsion would describe the interactions between these major parts of the system.
You may decide at this point to call the Propulsion parts (Propulsion Hardware, Propulsion Software) subsystems. Each subsystem has its own architecture and design, but is treated as a “black box” from the point of view of the rest of the system. As long as the subsystem conforms to its interface and any requirements imposed by Propulsion, Ground Vehicles, and RWGS, nothing at any level of the RWGS system should be concerned with how the subsystem is implemented.

Notice that this hierarchy can be easily expanded or collapsed depending on the needs of a particular project or program. For example, you might want to further decompose Propulsion into lower level systems such as power, engine, and treads. Then perhaps power has power hardware and power software. In another project, you might only be working on the Battlefield Management software and only have to be concerned with requirements at the level of RWGS and requirements for the Battlefield Management software.

**Documenting the System of Systems**

Proper use of encapsulation will prevent large overall changes when maintaining requirements. In general, changes will only affect one system. For example, a requirements change for Propulsion should not affect any other system in RWGS.

**Handling Requirements and Interfaces**

At the RWGS level, requirements documents describe the overall RWGS requirements and the interfaces to the parts of RWGS (Ground Vehicles, Air Vehicles, Water Vehicles, and Battlefield Management). From the point of view of RWGS, nothing needs to be known of the parts of RWGS, just the interfaces to those parts.

At the level of a part of RWGS (Ground Vehicles, Air Vehicles, Water Vehicles, and Battlefield Management), requirements documents describe the overall requirements for that part (such as the overall requirements for any Ground Vehicle) and the interfaces to the parts of the system (Command Interpreter Software, Path Analysis Software, Clock, Propulsion, and Sensors). From the point of view of Ground Vehicles in general, nothing needs to be known of the parts of a Ground Vehicle, just the interfaces to those parts.

At the level of a part of a Ground Vehicle (Command Interpreter Software, Path Analysis Software, Clock, Propulsion, and Sensors) requirements documents describe the overall requirements for that part (such as the overall requirements for the Path Analysis Software of a ground vehicle) and the interfaces to the parts of that system (such as Command Interpreter Software, Path Analysis Software, Clock, Propulsion, and Sensors). From the point of view of Command Interpreter Software in general, nothing needs to be known of the parts of the Command Interpreter Software, just the interfaces to those parts.

At the level of a part of the Propulsion (Propulsion Software, Propulsion Hardware), requirements documents describe the overall requirements for that part (such as the overall requirements for Propulsion Software) and the interfaces to the parts of that system (the internal
components of Propulsion Software). From the point of view of Propulsion Software in general, nothing needs to be known of the internal components of Propulsion Software, just the interfaces to those components.

**Focus on One Level of Detail**

What this will mean for someone working on the project, is that the person will focus on just their level of concern and not worry about anything else in the system. Someone at the RWGS level does not need to know anything about how Ground Vehicles, Air Vehicles, Water Vehicles, or Battlefield Management works, that person just needs to know how to interact with those parts. That information is described in the scenarios of RWGS and the interfaces to its parts.

Similarly, someone working at the Ground Vehicle level does not need to know how Command Interpreter Software, Path Analysis Software, Clock, Propulsion, or Sensors work or how they are structured. That person just needs to know how to interact with the Command Interpreter Software, Path Analysis Software, Clock, Propulsion, and Sensors. That information is described in the scenarios of Ground Vehicle and the interfaces to its parts.

**Contain Changes**

By using this approach, changes to requirements are contained within a small part of the system and confined to a person with expertise in that area. Now high level changes may flow down into lower level parts, but that flow down is well understood, and the actual change at the lower level is made by someone who is an expert on that part of the system.

Assume that a change is made to a requirement at the RWGS level. That change may cause a change to the scenarios of RWGS or the interfaces to the parts or both. The implementation of an RWGS scenario is described by a series of interactions with the interfaces of the parts, so the requirements change will flow down to the parts because of changes to their interfaces.

Inside a part, say Ground Vehicle, the change in the interface will be reviewed by someone who is responsible for the Ground Vehicle requirements. That person, who is an expert in that area, will determine what changes need to be made to the Ground Vehicle scenarios and the interfaces to the parts of Ground Vehicle.

The same will be true at each level of detail. So the change appears as a change to an interface, and a person responsible for the part connected to the interface will determine what that change means for the requirements of that part.

From the other direction, changes inside of Propulsion for example, will be contained within Propulsion and will not affect anything else throughout RWGS.

**Product Lines**

By separating out the ideas of system of systems and product line, we can focus on a particular set of needs and ignore other parts of the overall system. From the point of view of RWGS, the
fact that a robot travels on the ground is more important than the particular kind of robot it is. Now this is not completely true, and so we will incorporate the knowledge of types of ground robots into (for example) the Battlefield Management software. But for many of the requirements at the RWGS level, it really does not matter what kind of ground robot we have, it just matters if the robot is on the ground, in the water, or in the air.

For the similarities and differences between the robots, we use product line architecture. In RWGS we have systems such as Ground Vehicle or Water Vehicle identified. Actually, these are placeholders for a whole set of products. Ground Vehicle represents any kind of ground robot, Water Vehicle represents any kind of water based robot, and Air Vehicle represents any kind of flying robot. Ground Vehicle, Water Vehicle, and Air Vehicle each represent not one robot, but rather a product line of related robots.

**Documenting a Product Line**

When writing use cases for a product line, you will focus more on scenarios than whole use cases. Remember that a use case represents a collection of related scenarios. Each individual scenario is one requirement. In this section, I only discuss Use Case requirements, though the project will also include non-functional requirements for things that cannot be expressed as use cases (such as performance requirements).

The common requirements of the product line are implemented in every product in the product line. So for example, you can expect to see a startup use case as a requirement for all ground vehicles. These requirements could be implemented one time, and the same source code used in all ground vehicles. These common requirements are described by use cases that include a main scenario and common alternative scenarios.

Some common requirements may be implemented somewhat differently in different products in the product line. While every product has an implementation of the common requirement, the source code may be somewhat different in the different vehicles. This describes variants of the common requirements. So for example, while each ground vehicle has to be started, the startup code might be somewhat different in each type of vehicle. These variant requirements are described by adding alternative scenarios to the common use cases.

Requirements that are specific to one vehicle type will be described by use cases that include a main scenario and alternative scenarios. The implementation of these use case scenarios will appear in source code in just this one vehicle type.
Common requirements – these are use case scenarios or non-functional requirements that apply to all products in a product line, for example, requirements that apply to all ground vehicles in RWGS.

Vehicle Variant requirements – these are use case scenarios or non-functional requirements that are variations of common requirements which apply to a specific kind of product in the product line, for example, Mobile Platform specific variations of common requirements.

Vehicle Specific requirements – these are use case scenarios or non-functional requirements that apply to a specific kind of product in the product line, for example, requirements that apply only to Mobile Platform robots and not other types of vehicles.

A particular vehicle, such as Mobile Platform, will implement one or more scenarios of each common use case, plus any new use cases specific to that vehicle. In addition, the vehicle will
implement other requirements defined for the system, such as interfaces, architectural patterns, security, and performance.

You see in Figure 2 that Mobile Platform depends on Mobile Platform Vehicle Variant Requirements which inherits from Ground Vehicle Common Requirements. This means that Mobile Platform will implement all of the Ground Vehicle Common Requirements, either as they are written in Ground Vehicle Common Requirements or as they are modified by Mobile Platform Vehicle Variant Requirements. In addition, Mobile Platform will implement the requirements in Mobile Platform Vehicle Specific Requirements, Architecture and general project-wide requirements, and Interfaces.

The Ground Vehicle Common Requirements can be divided into separate systems, so the diagram can be expanded to show all the systems that apply to a particular vehicle, as shown in Figure 3.
To make life even more exciting, you can extend the requirements for a vehicle, treating the vehicle requirements as we did common requirements. See Figure 4 for example.

Figure 4 indicates that Heavy Duty Platform uses the same requirements as Mobil Platform, but may have some variations specific to Heavy Duty Platform (such as requirements for armoring and a chassis that handles more weight). And for those who take their games seriously, we extend the Heavy Duty Platform with a Paint Ball Turret so we can have our robot shoot other robots with paint.
Tracing requirements

IRD = interface requirements document
RD = requirements document
Entity = the thing you are describing at any level (RWGS, Ground Vehicle, Propulsion, Sensor Hardware, Sensor software, etc.)
Supplementary requirements = any non-use case requirement

Figure 5 shows the traces for Ground Vehicle Software Requirements from requirements documents to UML models to test cases. Note that component in Figure 5 can represent any independent design element, such as a class, dll, java file, .exe, process, or thread. Some
companies like to put the interface requirements in a separate document from the other requirements, so I have shown the two types of documents on this diagram.

Some supplementary requirements can be traced to specific functionality (scenarios) in the system and can be tested in the context of a relatively small number of scenarios. Others are more global, affecting large numbers of scenarios. These supplementary requirements should have their own test cases that apply to the system as a whole.

A specific supplementary requirement might be something like “A sensor must return its status within 0.1 ms of the request.” This requirement affects a relatively small part of the system. A global supplementary requirement might be something like “The Mean Time Between Failure (MTBF) of the RWGS shall be 9 months or greater.” This requirement affects the whole RWGS system and must be tested separately from a specific scenario or set of scenarios.

Here are some types of supplementary (non-Use Case) requirements that you may need to consider for a project.

<table>
<thead>
<tr>
<th>Supplementary Requirement Type</th>
<th>Subtypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Suitability, Accuracy, Interoperability,</td>
</tr>
<tr>
<td></td>
<td>Compliance, Security</td>
</tr>
<tr>
<td>Usability</td>
<td>Learnability, Understandability, Operability</td>
</tr>
<tr>
<td>Reliability</td>
<td>Maturity, Recoverability, Fault tolerance</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Stability, Analyzability, Changeability,</td>
</tr>
<tr>
<td></td>
<td>Testability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Time behavior (performance), Resource behavior</td>
</tr>
<tr>
<td>Portability</td>
<td>Installability Conformance, Replaceability,</td>
</tr>
<tr>
<td></td>
<td>Adaptability, Scalability</td>
</tr>
</tbody>
</table>
Change scenarios and the trace model

So what happens with this whole, complex requirements model when changes need to be made? Following are some example scenarios describing what has to happen when making changes to various parts of the requirements or analysis model. These scenarios do not describe HOW to make the changes; just WHAT to do when change happens. These scenarios would be modified for a particular project to match how the work is assigned to the various teams.

- Change a use case
  1. Look at the related scenarios in the requirements database to determine if they need changes.
  2. Find the scenario realizations to see if they need changes. This includes a review of the interfaces, the associated components, and any code already written in the associated components.
  3. Verify that any interface requirements traced to the changed interfaces have not changed or do the work needed to change the interface requirements.
  4. Review the test cases associated with the changed scenarios to determine if the test cases need changes.
  5. Run the test cases to find any results not consistent with the changed use case.

- Change a scenario
  1. Look at the related use case and all of its scenarios in the requirements database to determine if they need changes.
  2. Follow “Change a use case above”, starting with step 2 and going to the end.

- Change an interface requirement
  1. Follow the trace to the associated design interfaces to see if they need changes.
  2. Review the associated components and any code already written to see if they need changes.
  3. Review the scenario realizations in which the interfaces appear to see if the realizations need changes.
  4. Run the test cases to find any results not consistent with the changed interface.

- Change a specific supplementary requirement
  1. Follow the trace to the scenario to see if it needs any changes.
  2. Follow the steps in change a scenario above.

- Change a global supplementary requirement
  1. Follow the trace to the test case to see if it needs any changes.
  2. Run the test case to find any results not consistent with the changed requirement.
  3. Use the results of the test case to analyze the code for changes.
• Change a design interface
  1. Review the components associated with the interface and their code to see if they need any changes.
  2. Find the associated interface requirement to see if it needs to be changed.
  3. Find any other interfaces associated with the interface requirement to see if they need to be changed.
  4. Find the scenarios that use those interfaces.
  5. Run the test cases for those scenarios to find any errors that may have resulted from changing the interface.

• Change a component
  1. Find the interfaces associated with the component.
  2. Find the scenarios that use those interfaces.
  3. Run the test cases for those scenarios to find any errors that may have resulted from changing the component.

**Identification and Structure of the Use Cases**

Each use case will be structured as a set of scenarios. A suggested process for writing the use case is this:

1. Identify the use cases and actors. Create a use case diagram showing the system, use cases, and actors.
2. For each use case, write a brief description and the preconditions of the use case.
3. Based on the brief description, identify the various scenarios of the use case.
4. For each scenario, write the postconditions of the scenario.
5. For each scenario, write a descriptive statement that is composed of the use case preconditions and the postconditions of the scenario. This is useful for companies who test a scenario as a black box, using only the preconditions and the postconditions for the test.

This describes the basic use cases and scenarios of your system, at a level appropriate for black box testing of the functionality. In general, this is not enough information to design the system.

Next, describe the functionality of the use case by adding to each scenario a list of steps that describe how to go from the preconditions of the use case to the post conditions of the scenario. The brief description of the use case describes the basic functionality of the use case, which is detailed in the steps of the scenarios.

In the steps of the scenarios, identify what thing is doing each action. The words you will use to describe the thing depend on the level of the use case and the kind of project. For example, if you have a single system, you can identify the system as “the system”. When working in a system of systems, you need to further identify which system is being discussed. For example, consider the RWGS project, which is a very large system of systems. For requirements that apply to all of RWGS, the scenarios describe “the system” or “RWGS”. For requirements that apply to a system
such as Battlefield Management, Ground Vehicles, or Air Vehicles, the scenarios describe “the Battlefield Management system”, or “the Ground Vehicles system”, or “the Air Vehicles system”. For requirements that apply to a system such as Propulsion or Path Analysis, the scenarios describe “the Propulsion System” or “the Path Analysis system”. Finally, for requirements that apply to a system such as Propulsion software or Propulsion hardware, the scenarios describe “the Propulsion Software system” or “the Propulsion Hardware system”. In this manner, by looking at any requirement in any context, you always know exactly what the requirement applies to.

When describing message passing, you will often not need to explicitly identify the sender of the message. This makes it easier to maintain the requirements in the future. So for example, “Vehicle receives a command to initialize”. If the sender is important then state, “Vehicle receives a command to initialize from Battlefield Management.” You can also reference senders or receivers by using other documents or tables in this manner, “Vehicle sends a stop command to all of the vehicle components (see table xx for a list of all vehicle components).”

**Requirement relationships for system of systems combined with product line**

![Figure 6](image-url)
Figure 6 shows the relationship between different levels of requirements through the different systems. This is showing the common requirements of the system, not the vehicle variant or vehicle specific requirements.

Vehicle variant and vehicle specific requirements can be handled by creating new IRD and/or RD at any level. So you could add a new ground vehicle with new Command Interpreter, Path Analysis, Sensor, Propulsion, and new Propulsion Hardware and Propulsion Software as needed to show the new requirements for the new vehicle.

Handle the vehicle variant use case requirements by adding scenarios to the common use cases. Handle the vehicle variant supplementary requirements by adding them to the appropriate RD and tracing the original requirement to the variant. Add vehicle specific requirements of all kinds to the appropriate RD.

Compare Figure 6 to Figure 3. Figure 3 is describing common, vehicle variant, and vehicle specific requirements for a specific vehicle type called Mobile Platform. In Figure 3, the box marked “Interfaces to/from battlefield management” is a general description of what to do. This box becomes the Ground Vehicle IRD in Figure 6. In Figure 3, the box marked “Architecture and general project-wide requirements” becomes the RWGS RD and Ground Vehicle RD in Figure 6, and any architecture requirements that apply to ground vehicles. In Figure 3, the box marked “Propulsion: Common Requirements” becomes the Propulsion IRD and RD, and the IRDs and RDs for Propulsion Hardware and Propulsion Software in Figure 6.

Be sure requirements at a particular level really do apply to the systems at that level. For example, requirements for RWGS apply to Ground Vehicles, Water Vehicles, Air Vehicles, and Battlefield Management. At the level of a Ground Vehicle, that RWGS requirement may apply to all the parts of the Ground Vehicle (such as Command Interpreter, Path Analysis, Propulsion, Sensors) or only some of the parts.

**Example: Requirements and Analysis Model for Combined System of Systems and Product Line**

Below I show an example of a small part of the RWGS system. I am not trying to show complete, formal documentation, but rather to give an example of the level of detail appropriate to the different parts of the system. For each system, you will find one example use case in bold, and several supplementary requirements. Some of the diagrams are quite compressed, and will be easier to read online by setting the document zoom to 150%.

In creating the example, I started with an activity diagram for a system. The activities on the activity diagram become use cases for that system, and the events on the transitions become interfaces for that system. Supplementary requirements do not appear on activity diagrams.

Next, I wrote the use cases. The events on the transitions that trigger the activities appear as preconditions of the use cases. Results of the activities are postconditions of the use cases.
Then I created sequence diagrams for the use cases. The objects on the sequence diagram are the parts of the system. Events that trigger activities on the activity diagram (that are preconditions of the use case) appear in the sequence diagram as the first message or trigger for the use case.

**RWGS Level Requirements**

**RWGS Activity Diagram**

### RWGS RD

**Use Case: Run Training Exercise**
1. Design training plan
2. Determine available equipment
3. Configure Training Plan
4. Execute Training Plan
5. End Training Exercise
6. Review Results

*Use Case: Manage Battlefield*
*Use Case: Maintain Vehicles*
*Use Case: Upgrade Software*
*Use Case: Run Reports*
<table>
<thead>
<tr>
<th>Shall Requirement Type</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality – Suitability</td>
<td></td>
</tr>
<tr>
<td>Functionality – Accuracy</td>
<td></td>
</tr>
<tr>
<td>Functionality – Interoperability</td>
<td></td>
</tr>
<tr>
<td>Functionality – Compliance</td>
<td>All electrical equipment in RWGS shall comply with electronics testing</td>
</tr>
<tr>
<td></td>
<td>requirements set by United Laboratories (UL)</td>
</tr>
<tr>
<td>Functionality – Security</td>
<td></td>
</tr>
<tr>
<td>Usability – Learnability</td>
<td></td>
</tr>
<tr>
<td>Usability – Understandability</td>
<td></td>
</tr>
<tr>
<td>Usability – Operability</td>
<td>RWGS shall be operable by any person with 8 years of schooling (junior high</td>
</tr>
<tr>
<td></td>
<td>school in the USA).</td>
</tr>
<tr>
<td>Reliability – Maturity</td>
<td></td>
</tr>
<tr>
<td>Reliability – Recoverability</td>
<td></td>
</tr>
<tr>
<td>Reliability – Fault tolerance</td>
<td>RWGS shall provide at least one level of redundancy for all systems</td>
</tr>
<tr>
<td>Maintainability – Stability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Analyzability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Changeability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Testability</td>
<td></td>
</tr>
<tr>
<td>Efficiency – Time behavior (performance)</td>
<td></td>
</tr>
<tr>
<td>Efficiency – Resource behavior</td>
<td></td>
</tr>
<tr>
<td>Portability – Installability</td>
<td></td>
</tr>
<tr>
<td>Portability – Conformance</td>
<td></td>
</tr>
<tr>
<td>Portability – Replaceability</td>
<td></td>
</tr>
<tr>
<td>Portability – Adaptability</td>
<td></td>
</tr>
<tr>
<td>Portability – Scalability</td>
<td></td>
</tr>
</tbody>
</table>
RWGS Use Case Realization: Run Training Exercise

1: Design Training Exercise
   2: What kind of vehicle are you
   3: vehicle type
   4: What equipment do you have
   5: equipment list
   6: What kind of vehicle are you
   7: vehicle type
   8: What equipment do you have
   9: equipment list
   10: What kind of vehicle are you
   11: vehicle type
   12: What equipment do you have
   13: equipment list

14: Configure Training Exercise - TrainingPlan
   15: Configure for Training (TrainingPlan)
   16: Training configuration status
   17: Configure for Training (TrainingPlan)
   18: Training configuration status
   19: Configure for Training (TrainingPlan)
   20: Training configuration status

16: Configure for Training (TrainingPlan)
   16: Training configuration status

21: Execute Training Plan
   22: status reports
   23: Execute Training Plan
   24: status reports
   25: Execute Training Plan
   26: status reports

27: Return to home
   28: I am home
   29: Power off

30: Return to home
   31: I am home
   32: Power off

33: Return to home
   34: I am home
   35: Power off

36: Review Results
Ground Vehicle Level Requirements

Ground Vehicle Activity Diagram

- Start up Vehicle
- Power turned on
- Determine Vehicle Type
- Determine Equipment
- Run Training Exercise
- Configure Training Plan
- Execute Training Plan
- Configure for Training (Training Plan)
- Configure for standard operations
- What is your current status
- Emergency Stop
- Report Status
- Change to Operational Mode
- Power off
- Emergency Stop
- Shut down vehicle
Ground Vehicle IRD
Interface: Power Turned On
Interface: What kind of vehicle are you, return vehicle type
Interface: What kind of equipment do you have, return equipment list
Interface: Configure for Training (TrainingPlan), return training configuration status
Interface: Execute Training Plan, return status reports
Interface: Configure for Standard Operations
Interface: What is your current status, return vehicle status
Interface: Emergency Stop
Interface: Power Off

Ground Vehicle RD
Use Case: Start up Vehicle
Precondition: Power turned on
Use Case: Determine Vehicle Type
Precondition: The vehicle has received a “What kind of vehicle are you” command
Postcondition: The vehicle has returned its type to the requester
Use Case: Determine Equipment
Precondition: The vehicle has received a “What kind of equipment do you have” command
Postcondition: The vehicle has returned an equipment list to the requester
Use Case: Run Training Exercise
Precondition: The vehicle has received an “Execute Training Command” command
Postcondition: The vehicle has returned status reports to the requester
Use Case: Configure Training Plan
Precondition: The vehicle has received a “Configure for Training (TrainingPlan)” command
Postcondition: The vehicle has returned its training configuration status to the requester
Use Case: Change to Operational Mode
Precondition: The vehicle has received a “Configure for Standard Operations” command
Use Case: Report Status
Precondition: The vehicle has received a “What is your current status” command
Postcondition: The vehicle has returned vehicle status to the requester
Use Case: Emergency Stop
Precondition: The vehicle has received an “Emergency Stop” command
Use Case: Shut down Vehicle
Precondition: The vehicle has received a “Power Off” command.
1. Check and hold for shutdown of vehicle
2. Prepare to shutdown vehicle
3. Shutdown vehicle
<table>
<thead>
<tr>
<th>Shall Requirement Type</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality – Suitability</td>
<td></td>
</tr>
<tr>
<td>Functionality – Accuracy</td>
<td></td>
</tr>
<tr>
<td>Functionality – Interoperability</td>
<td></td>
</tr>
<tr>
<td>Functionality – Compliance</td>
<td></td>
</tr>
<tr>
<td>Functionality – Security</td>
<td>Upon receiving an emergency stop command, Ground Vehicle shall immediately stop the vehicle</td>
</tr>
<tr>
<td>Usability – Learnability</td>
<td></td>
</tr>
<tr>
<td>Usability – Understandability</td>
<td></td>
</tr>
<tr>
<td>Usability – Operability</td>
<td></td>
</tr>
<tr>
<td>Reliability – Maturity</td>
<td></td>
</tr>
<tr>
<td>Reliability – Recoverability</td>
<td></td>
</tr>
<tr>
<td>Reliability – Fault tolerance</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Stability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Analyzability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Changeability</td>
<td>When a field replaceable unit is replaced, Ground Vehicle shall detect and record the new vehicle configuration</td>
</tr>
<tr>
<td>Maintainability – Testability</td>
<td></td>
</tr>
<tr>
<td>Efficiency – Time behavior (performance)</td>
<td></td>
</tr>
<tr>
<td>Efficiency – Resource behavior</td>
<td></td>
</tr>
<tr>
<td>Portability – Installability</td>
<td></td>
</tr>
<tr>
<td>Portability – Conformance</td>
<td></td>
</tr>
<tr>
<td>Portability – Replaceability</td>
<td></td>
</tr>
<tr>
<td>Portability – Adaptability</td>
<td></td>
</tr>
<tr>
<td>Portability – Scalability</td>
<td></td>
</tr>
</tbody>
</table>
Ground Vehicle Use Case Realization: Shut Down Vehicle

1: Power Off
2: Check and hold for shutdown of vehicle
3: Check and Hold (shutdown)
   check and hold complete
5: Check and Hold (shutdown)
6: check and hold complete
7: Prepare to shutdown vehicle
8: Prepare for mode change (shutdown)
9: preparation complete
10: Prepare for mode change (shutdown)
11: preparation complete
12: Shutdown vehicle
13: Complete mode change (shutdown)
14: shutdown propulsion
15: Complete mode change (shutdown)
16: shutdown sensors
17: Shutdown software
18: terminate
Command Interpreter Level Requirements

Command Interpreter Activity Diagram

Command Interpreter IRD
Interface: Vehicle Power-up
Interface: Initialize
Interface: Start
Interface: Change Mode (mode)
Interface: Prepare for Mode Change (mode)
Interface: Configure Vehicle Systems (operational mode)
Interface: Complete Mode Change (mode)
Interface: Emergency Stop Received
Interface: Training Exercise Platform Configuration (TrainingPlan)
Interface: Prepare for Training
Interface: Configure for Start
Interface: Begin
Interface: Interrupt
Interface: Initiate De-allocation

**Command Interpreter RD**

**Use Case: Launch Command Interpreter**
Precondition: The vehicle is powered up

**Use Case: Initialize Vehicle Systems**
Precondition: The vehicle has received an “Initialize” command

**Use Case: Configure Vehicle Systems**
Precondition: The vehicle has received a “Configure Vehicle Systems (operational mode)” command

**Use Case: Determine Health of Vehicle Systems**
Precondition: None.

**Use Case: Start Vehicle Systems**
Precondition: The vehicle has received a “Start” command

**Use Case: Report Overall Health of Vehicle Systems**
Precondition: None

**Use Case: Report Heartbeat**
Precondition: A clock tick has occurred
1. The Command Interpreter verifies that all hardware is on and responsive.
2. If any hardware is not on or not responsive, the Command Interpreter sets the heartbeat to failure.
3. Otherwise, the Command Interpreter sets the heartbeat to operating.

**Use Case: Monitor System Health**
Precondition: None.

**Use Case: Detect Faults**
Precondition: None.

**Use Case: Vehicle Mode Change Check and Hold**
Precondition: The vehicle has received a “Change Mode (mode)” command

**Use Case: Prepare for Vehicle Mode Change**
Precondition: The vehicle has received a “Prepare for mode change (mode)” command

**Use Case: Complete Vehicle Mode Change**
Precondition: The vehicle has received a “Complete mode change (mode)” command

**Use Case: Shutdown Systems**
Precondition: The vehicle has received a “Complete mode change (shutdown)” command

**Use Case: Execute Emergency Stop**
Precondition: The vehicle has received an “Emergency Stop” command

**Use Case: Provide Configuration for Training Exercise**
Precondition: The vehicle has received a “Training Exercise Platform Configuration (TrainingPlan)” command
Use Case: Prepare for Training Exercise
  Precondition: The vehicle has received a “Prepare for Training” command
Use Case: Configure to Exercise Start Condition
  Precondition: The vehicle has received a “Configure for Start” command
Use Case: Run Training Exercise
  Precondition: The vehicle has received a “Begin” command
Use Case: Interrupt Training Exercise
  Precondition: The vehicle has received a “Interrupt” command
Use Case: De-allocate System Resources
  Precondition: The vehicle has received a “Initiate De-allocation” command

<table>
<thead>
<tr>
<th>Shall Requirement Type</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality – Suitability</td>
<td></td>
</tr>
<tr>
<td>Functionality – Accuracy</td>
<td></td>
</tr>
<tr>
<td>Functionality – Interoperability</td>
<td></td>
</tr>
<tr>
<td>Functionality – Compliance</td>
<td></td>
</tr>
<tr>
<td>Functionality – Security</td>
<td></td>
</tr>
<tr>
<td>Usability – Learnability</td>
<td></td>
</tr>
<tr>
<td>Usability – Understandability</td>
<td></td>
</tr>
<tr>
<td>Usability – Operability</td>
<td></td>
</tr>
<tr>
<td>Reliability – Maturity</td>
<td></td>
</tr>
<tr>
<td>Reliability – Recoverability</td>
<td></td>
</tr>
<tr>
<td>Reliability – Fault tolerance</td>
<td>The command interpreter software must recover from all failures and continue</td>
</tr>
<tr>
<td></td>
<td>operating, except in the case of power failure.</td>
</tr>
<tr>
<td>Maintainability – Stability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Analyzability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Changeability</td>
<td></td>
</tr>
<tr>
<td>Maintainability – Testability</td>
<td></td>
</tr>
<tr>
<td>Efficiency – Time behavior (performance)</td>
<td></td>
</tr>
<tr>
<td>Efficiency – Resource behavior</td>
<td></td>
</tr>
<tr>
<td>Portability – Installability</td>
<td></td>
</tr>
<tr>
<td>Portability – Conformance</td>
<td></td>
</tr>
<tr>
<td>Portability – Replaceability</td>
<td></td>
</tr>
<tr>
<td>Portability – Adaptability</td>
<td></td>
</tr>
<tr>
<td>Portability – Scalability</td>
<td></td>
</tr>
</tbody>
</table>
**Command Interpreter Use Case Realization: Report Heartbeat**

The command interpreter verifies that all hardware is on and responsive. The command interpreter sets the heartbeat to operating.

**Final Thoughts**

I encourage anyone working on this kind of complex system to take advantage of tools to help you maintain the information. At a minimum, you will need a good requirements management tool based on a database that can be shared among project teams.

The original client had many teams of people working on this system, who were in different geographic areas, and who had minimal communication with each other. Each team worked as its own independent project. In addition, the whole system is being developed over a period of many years, as different products in the lines of business are added over time. This document was developed to outline an approach, and was used to put together the tools and databases that would be required to manage all of this information over time.

I am always happy to get feedback on my articles. Please write and let me know what you think!

Best regards –

**Geri**
Geri Schneider Winters
information@wyyzzk.com
References and Websites
Geri Schneider Winters has been a mentor and consultant to companies from Fortune 50 to startups. A software engineer for over 20 years, she assists companies in describing, documenting, and creating their business processes and software systems. A popular author, speaker, and trainer, Geri has worked throughout the USA, Canada, and the UK. Geri is the primary author of the popular book “Applying Use Cases: A Practical Guide”, from Addison-Wesley publishers.

Geri’s websites
Resources for Business Analysts http://www.writingusecases.com
Resources for Owner/Managers of Internet businesses http://realizeyourbusiness.com
10th Anniversary Celebration of Applying Use Cases http://www.applyingusecases.com
3D Requirements Relationship Modeling Tool http://www.3dvizz.com