Australian Government
Department of Defence
Defence Science and
Technology Organisation

The PARTI Architecture Assurance

B Mahony
A Cant

Command, Control, Communications and Intelligence Division

Defence Science and Technology Organisation

DSTO–GD–0557

ABSTRACT

Safety Critical Systems are those with the potential to cause death or injury as a result of accidents arising from unintended system behaviour. The arguments for safety, along with the body of supporting evidence, make up what is called the *Safety Case*. Requirements and guidance for Safety Cases are given in Def (Aust) 5679 Issue 2 [2]; in this standard the key stages of the Safety Case are: *Hazard Analysis*, *Safety Architecture* and *Design Assurance*. The process is driven by the identification of *System Safety Requirements*. The standard requires an argument be made that the Safety Architecture meets the System Safety Requirements. In the most serious cases, this argument is required to be made in a formal language and supported by formal reasoning tools. In this paper, we demonstrate the feasibility of such formal argument through the presentation of a formal verification argument for a simplified case study in Defence safety engineering.

APPROVED FOR PUBLIC RELEASE

Published by

DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: *(08) 8259 5555*
Facsimile: *(08) 8259 6567*

© *Commonwealth of Australia 2009*
AR No. AR 014-350
October, 2008

APPROVED FOR PUBLIC RELEASE

The PARTI Architecture Assurance
Executive Summary

The Australian Standard for safety-critical systems development, DEF(AUST)5679, was first released in 1998. As part of the release of Issue 2 [21] of the STANDARD, guidance material has been prepared to assist those who need to apply the STANDARD. The guidance is made up of two main parts.

1. A Case Study that demonstrates how the STANDARD can be applied to an example safety critical system,
2. Issues Guidance Papers (IGPs) that further explain key concepts or requirements of the STANDARD.

The guidance material case study demonstrates the application of the STANDARD to an example system known as the PARTI (Phased Array Radar and Target Illumination). The PARTI system is a fictional upgrade of a class of Navy frigates with the installation of a self defensive capability. It scans, detects, discriminates, selects and illuminates airborne threats. The system is intended to work in conjunction with the existing Evolved Sea Sparrow Missile system, in that the illumination of targets by the PARTI provides a target fix for the ESSM. The system incorporates an electronically-scanned phased array radar and associated control functionality, a sub-system for detecting, discriminating and selecting airborne threats, and a laser designator and associated control functionality, for high speed self-defence missile target illumination. The system also incorporates the PARTI operator.

The HAZARD ANALYSIS of the PARTI system [7] determined a number of potentially serious hazards and established a SYSTEM DANGER LEVEL of D_5. In this case, DEF(AUST)5679/ISSUE 2 requires that SAFETY ARCHITECTURE analysis must include a FORMAL model of the SYSTEM SAFETY REQUIREMENTS. The COMPONENT SAFETY REQUIREMENTS are also formally specified and a FORMAL proof is used to show that the SYSTEM SAFETY REQUIREMENTS follow from the COMPONENT SAFETY REQUIREMENTS.

This paper has been prepared in conjunction with and in response to the SAFETY ARCHITECTURE REPORT prepared for the guidance material case study [8]. The SAFETY ARCHITECTURE REPORT from the case study uses an Event-B modelling approach [1], supported by the Rodin tool [25], to carry out the required architecture modelling and verification. In its application to the PARTI architectural model, Event-B proved less satisfactory in some aspects and this paper demonstrates an alternative SAFETY ARCHITECTURE VERIFICATION approach using the Isabelle theorem prover [23]. This approach offers both a lower development overhead to the SUPPLIER and also produces a model and associated arguments that are more accessible to a wider audience.

Authors

Dr Brendan Mahony
Defence Science and Technology Organisation

Dr Mahony was admitted as a PhD by the University of Queensland in 1991. After post-doctoral research in high-assurance real-time systems, he joined the DSTO in 1995. With DSTO he has continued his research into high-assurance design methods and has guided the development of the high-assurance tools DOVE and the HiVe.

Dr Tony Cant
Defence Science and Technology Organisation

Tony Cant obtained a BSc (Hons) in 1974 and a PhD in Mathematical Physics in 1979, both from the University of Adelaide. He also obtained a Grad Dip in Computer Science from the ANU in 1991. Before joining DSTO in 1990, Tony did postdoctoral research in mathematical phsyics and worked in science policy for the Australian Government. With DSTO Tony has led work on high assurance methods and tools for critical systems, and has provided advice to Defence on Safety-Critical Systems. Tony is the editor-in-chief of the Australian Defence Standard Def (Aust) 5679, which is currently undergoing revision and is entitled “Safety Engineering for Defence Systems”.

Contents

1 Introduction

1.1 Background

Safety Critical Systems are those with the potential to cause death or injury as a result of accidents arising from unintended system behaviour. An effective safety engineering process must be carried out in order to ensure that the system is safe to use in its intended operating environment. The arguments for safety, along with the body of supporting evidence, make up the *Safety Case*: this is presented to safety Evaluators and Certifiers for assessment.

Requirements and guidance for Safety Cases are given in Def (Aust) 5679 Issue 2 [2], which has recently been prepared by the Defence Science and Technology Organisation (DSTO), following an extensive revision process supported by Invensys Rail Systems Australia and sponsored by the Defence Materiel Organisation (DMO).

1.2 Def (Aust) 5679: The Safety Case

Def (Aust) 5679 provides detailed requirements and guidance on the structure of the *Safety Case*. In particular, it prescribes certain *assurance* activities: these are system development and analysis activities that provide evidence that the system meets its safety requirements. The key stages of the Safety Case are as follows:

- An analysis of the danger that is potentially presented by the System. This involves an assessment of the System Hazards, and the ways that Accidents could occur and how serious they would be. This is called *Hazard Analysis*.
- A System Design that provides safety features, i.e. a *Safety Architecture*.
- Arguments that System Components have been built in such a way that provides assurance of safety, called *Design Assurance*.
- An overall narrative (or high-level argument) that is convincing to a third-party and pulls all the above together.

1.3 Hazard Analysis

The aim of *Hazard Analysis* is to describe the *System*, its *Operational Context* and identify all possible *Accident Scenarios* (and their associated *Danger Levels*) that may be caused by a combination of the states of the *System*, environmental conditions and external events.

An *Accident* is an external event that could directly lead to death or injury. The *Severity* of an Accident is a measure of the degree of its seriousness in terms of the extent of injury or death resulting from the accident. The possible severities are *Catastrophic*, *Fatal*, *Severe* and *Minor*. *System Hazards*

Table 1: DANGER LEVELS

		EXTERNAL MITIGATION		
ACCIDENT SEVERITY	Default	LOW	MEDIUM	HIGH
CATASTROPHIC	D_6	D_6	D_5	D_4
FATAL	D_5	D_5	D_4	D_3
SEVERE	D_4	D_4	D_3	D_2
MINOR	D_3	D_3	D_2	D_1

are top-level states or events of the system from which an Accident, arising from a further chain of events external to the System, could conceivably result. *Accident Scenarios* describe a causally related collection of System Hazards and *Coeffectors* that lead to a defined Accident.

The Operational Context should be referenced to determine the strength of *External Mitigation* present in each Accident Scenario. An External Mitigation is an external factor that serves to suppress or reduce the occurrence of Coeffectors. For each Accident Scenario the strength of External Mitigation must be assessed as one of *Low*, *Medium* or *High*.

There are six Danger Levels labelled from D_1 to D_6, and these are assigned in accordance with the following requirements:

- For each Accident Scenario, a *default* Danger Level is assigned based on the Accident Severity using Table 1.
- If no External Mitigations are present in the Accident Scenario, the Danger Level remains at the default value for that severity.
- If, for a given Accident Scenario, a strength of External Mitigation can be assigned, then the Danger Level is reduced from its default value according to Table 1.

1.4 Safety Architecture

The aim of the *Safety Architecture* phase is to describe the *System Architecture* in terms of *Components*; to describe *Safety Features* present in the architecture; to detail *System Safety Requirements* (SSRs) and how these requirements flow down to *Component Safety Requirements* (CSRs); and to provide arguments (at an appropriate level of formality) that satisfaction of the CSRs entails satisfaction of the SSRs.

Corresponding to each System Hazard is a System Safety Requirement demanding that the hazard does *not* occur. The SSRs must be specified to the same level of formality, depending on the System Danger Level, as given in Table 2.

When the System Danger Level is D_1, the SSRs may be expressed as English language statements; if it is D_2, the SSRs require some form of Semiformal specification, which may involve the use of structured

Table 2: Safety Architecture Assurance Attributes

	System Danger Level					
	D_1	D_2	D_3	D_4	D_5	D_6
System Safety Requirements	Informal	Semi-formal	Formal	Formal	Formal	Formal
Component Safety Requirements	Informal	Semi-formal	Formal	Formal	Formal	Formal
Safety Architecture Verification	Informal	Semi-formal	Semi-formal	Formal	Formal	Formal
Safety Architecture Testing	Low	Medium	Medium	High	High	High

English language statements, possibly augmented with diagrams and mathematical notation; if it is D_3 or higher, the SSRs are expressed in a formal specification language.

The System Architecture must be expressed in terms of Components that combine to carry out the system functions: a block diagram showing interfaces between Components, as well as inflows from (kinetic energy, thermal energy, radiation, chemical, data, etc.) and outflows to the environment, is the most suitable for this purpose. See Figure 8 for an example of such a block diagram.

The Architecture will be informed by safety considerations, exhibiting specific *Safety Features*: these are either:

- *Protective Measures*, i.e. Components introduced into the Architecture with a specific safety purpose; or
- *Partitioning* the design to isolate – both functionally and physically – Components that have safety requirements.

We then derive a collection of Component Safety Requirements (CSRs), expressed in terms of Component interfaces. The CSRs must be expressed to the level of formality given in Table 2.

Safety Architecture Verification provides arguments that satisfaction of the CSRs entails satisfaction of the SSRs. This is done at a level of formality, depending on the System Danger Level, given by Table 2. When the System Danger Level is D_1, it is sufficient that such proofs be Informal, i.e. expressed using clear and logical English arguments; if it is D_2 or D_3 some form of Semi-formal proof is required. Such proofs are mathematically convincing but may omit some of the detailed justification; if it is D_4 or higher, formal proofs are needed, i.e. proofs expressed using a formal mathematical logic.

1.5 Tool Support

Readers may wonder why such apparently high levels of formality are required for the assurance of the Safety Architecture. The justification is that it is important to have high confidence that the architecture is appropriate before proceeding with further system and safety case development; issues and errors not dealt with at this stage will have far-reaching and expensive consequences further in the system lifecycle.

We note that, for the highest Danger Levels, the use of formal methods is mandated for: the specification of component safety requirements; the component model; and the verification that the model meets these requirements. There are a limited number of tools that support this process. In this paper, we show how the Isabelle theorem prover can be used to provide this assurance.

1.6 Def (Aust) 5679: Guidance Material

Def (Aust) 5679 Issue 2 has been enhanced by the production by NOVA Defence of extensive Guidance Material, under contract to the DMO. This Guidance Material is based upon a fully developed Case Study — called the Phased Array Radar Target Illuminator (PARTI) system — illustrating how the standard can be applied to a realistic exemplar Defence system. The Case Study deliverables include the following technical Safety Case documents:

- Hazard Analysis Report [7];
- Safety Architecture Report [8];
- Design Assurance Report [9, 10, 11, 12, 13];
- Safety Case Summary Reports [14, 15, 16]; and
- Evaluation reports on the above [17, 18, 19].

The lessons learned from the development of this Case Study were used to inform four Issues Guidance Paper addressing key aspects of Def (Aust) 5679:

- Advice to Project Offices [3];
- Methods for Architecture Assurance [4];
- Methods for Design Assurance [5]; and
- Non-Development Items [6].

The Safety Architecture Report from this Def (Aust) 5679 Case Study uses an Event-B modelling approach [1], supported by the Rodin tool [25], to carry out the required architecture modelling and

verification. This approach was chosen because of the familiarity of the authors with Event-B and in expectation of the need to make further formal arguments at the Component Design level.

Event-B is a mature and powerful tool for exploring and verifying algorithmic designs, especially suitable for Software or Complex Hardware Components. In its application to the PARTI Architectural model, Event-B proved less satisfactory in some aspects. In particular, the need to structure the model in terms of (small-scale) operations represented a significant development overhead and also a barrier to comprehension of the model and the Architecture Assurance argument. Also, the main driver for using Event-B was the desire to adopt a common formalism for Architectural and Design modelling. This is not a compelling motivation in the Def (Aust) 5679 environment. The Standard introduces a deliberate break between the Architecture and Design levels, at least partially in acknowledgement of the difficulty in adopting a uniform formal approach through an entire development hierarchy.

The aim of this paper is to demonstrate an alternative Architecture Assurance approach using the Isabelle theorem prover [23]. Our claim is that this approach offers both a lower development overhead to the Supplier and also produces a model and associated arguments that are more accessible to a wider audience.

2 Some Isabelle Basics

Isabelle/Isar [26, 27] is a state-of-the-art theorem proving and mathematical modelling environment with sophisticated features for what is called *literate* proof development. A literate (formal) proof is a machine checkable proof that is also readily comprehensible to the (mathematically literate) human reader. Many theorem proving environments aim primarily to assist their users in *checking* the truth of mathematical propositions. Isabelle/Isar also aims to assist its users in *communicating* the truth of mathematical propositions.

The modelling approach taken in this paper makes use of an Isar feature called the *locale*. A locale is theory structuring mechanism that bundles a collection of fixed variables with associated assumptions about their properties. For example, a locale consisting of a non-empty carrier set may be defined by the following Isabelle commands.

locale *carrier* =
fixes
  S :: "α *set*"
assumes
  non_empty: "$S \neq \varnothing$"

This locale might then be extended to include a relation on the carrier set S as follows.

locale *relation* = *carrier* +
fixes
  R :: "$(\alpha \times \alpha)$ *set*"
assumes
  bound_relation: "$(\forall\ a\ b \mid (a \mapsto b) \in R \bullet a \in S \wedge b \in S)$"

Derived properties of a locale can be proved using the lemma command. For example, the following commands prove that a carrier set has at least one element.

lemma (**in** *carrier*) *witness*:
"(∃ *a* • *a* ∈ *S*)"
apply (*insert non_empty*)
apply (*auto*)
done

This result can now be used freely to show other properties of the *carrier* locale or indeed of the *relation* locale which inherits all of the properties of its parent locale.

Additionally, entailment relations can be shown between locales, demonstrating that all properties of the one are also properties of the other. For example, the following reformulation of the *relation* locale can be shown to be an *interpretation* of the original. That is to say that any structure satisfying the properties of *relation′* also satisfies those of *relation*.

locale *relation′* = *carrier* +
fixes
R :: "(α × α) *set*"
assumes
bound_domain: "dom *R* ⊆ *S*" **and**
bound_range: "ran *R* ⊆ *S*"

interpretation *relation′* ⊆ *relation*
proof (*auto simp add*: *relation_axioms_def*)
fix *a b*
assume *b1*: "(*a* ↦ *b*) ∈ *R*"
from *b1 bound_domain* **show** "*a* ∈ *S*"
by (*auto*)
from *b1 bound_range* **show** "*b* ∈ *S*"
by (*auto*)
qed

A brief discussion of Isabelle proof conventions is now in order. Proofs are either simple or compound. Simple proofs consist of a sequence of proof methods that are applied to the proof state to discharge the proof goal. If there is just one method, the "by" command is used. If more than one, a series of "apply" commands is used, terminated by a "done" command. Compound proofs also involve the application of methods to transform the proof state, but include explicit statements of the local proof goals that establish the higher level goal. Compound proofs are enclosed between "**proof**" and "**qed**" commands.

The locale mechanism is useful in a number of ways.

It provides a logically safe modelling environment in that it is logically separated from the underlying Isabelle/HOL formalism. Creating or combining locales with inconsistent assumptions does not lead to a collapse of the underlying formalism, since all locale results are predicated on the satisfaction of the locale assumptions.

The locale mechanism provides a very natural approach to systems modelling. The system observables can be modelled as fixed variables in a locale and the system properties or specifications as assumptions to the same locale. The hierarchical nature of locales also provides natural support for system design hierarchies and the interpretation mechanism provides a natural approach to showing refinement between the various system design phases. In particular, we can define one locale to model the System Safety Requirements and another to model the System Architecture in terms of the Component Safety Requirements and then show an interpretation between the two to establish that the Architecture satisfies the System Safety Requirements.

We conclude this section with a discussion of the mathematical conventions adopted in this paper. The mathematical syntax adopted in the following is just that of the Isabelle/HOL logic, but augmented with a surface syntax to make it more familiar to the Z practitioner [22]. Additionally, we make use of an Isabelle/HOL implementation of the Z Mathematical Toolkit [24] so that the overall modelling approach should be easily accessible to the large Z and B user communities.

3 The PARTI System

theory *PARTI*

imports
Math_Kit prelim

begin

The PARTI (Phased Array Radar and Target Illumination) System is a ship-borne Surface to Air Missile (SAM) targeting support system. It uses Phased Array Radar (PAR) to direct laser illumination of hostile missiles and aircraft. The laser illumination provides targeting information to an existing ownship SAM capability. The main items of interest in the PARTI and environment are depicted in Figure 1.

Briefly, the system operates in the following manner. A companion general purpose radar system provides course-grained tracking of nearby objects and an Identify Friend and Foe (IFF) system sorts friendly from hostile tracks. The PARTI operator uses this information to identify, in particular, potential airborne threats to the ownship. The PAR subsystem emanates directed beams of microwave radiation and analyses returning reflections to provided precision *tracking* of these threats. This precision tracking information is used to accurately direct, in real-time, beams of laser light that act as target illumination home-all-the-way for a companion SAM system that is then used to destroy the threats.

In this section, we formally describe the PARTI and its environment in terms of the inflows and outflows of the system, as summarised in Figure 2.

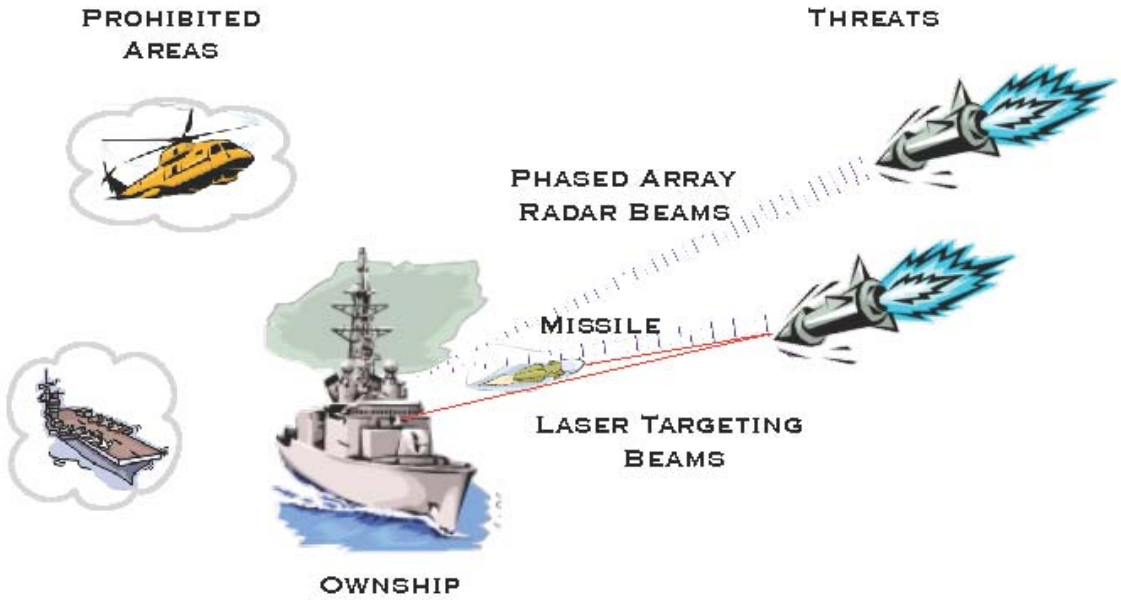


Figure 1: PARTI System Overview

3.1 Outflows

The outflow of the PAR subsystem consists of *parnum* time-varying beams of microwave energy.

locale *PARTI_const_parnum* =
  fixes *parnum* :: "ℕ"
locale *PARTI_con_parbeams* =
  fixes *parbeams* :: "[time, ℕ] → *volume*" ("parbeams")

For the purposes of describing the safety architecture of the PARTI, we find it convenient to approximate the notion of beam as a cylindrical volume of space emanating from an *origin* point on the ownship.

consts
  beam :: "[ℝ, *point*, *point*] → *point set*"

defs
  beam_def:
    "beam $w\ \vec{p}_1\ \vec{p}_2 \triangleq (\bigcup \vec{z} \mid \vec{z} \in \{\ \alpha \mid \alpha \in [0 \ldots 1] \bullet \vec{p}_1 + \alpha(\vec{p}_2 - \vec{p}_1)\ \} \bullet$
      $\{\ \vec{q} \mid sqrt\ (\vec{q} \odot \vec{q}) \leq |w| \wedge \vec{q} \odot (\vec{p}_1 - \vec{p}_2) = 0 \bullet \vec{z} + \vec{q}\ \})$"

locale *PARTI_const_origin* =
  fixes *origin* :: "*point*"

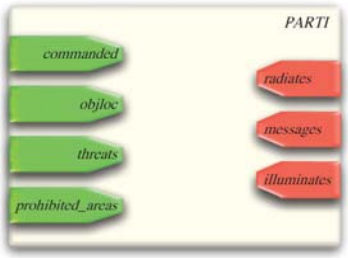


Figure 2: PARTI System Boundary

The width of the radar beams is *parbw*.

locale *PARTI_const_parbw* =
fixes *parbw* :: "ℝ"

The overall pattern of radiation from the PAR is then the union of the volumes irradiated by the individual beams.

locale *PARTI_con_radiates* = *PARTI_const_parnum* + *PARTI_con_parbeams* +
fixes
radiates :: "ℝ → *volume*" ("radiates")
defines
radiates_def: "radiates ≙ (λ *t* • (⋃ *r* | *r* ≤ *parnum* • parbeams *t r*))"

Similarly, the outflows from the laser illuminator consist of *lasnum* beams of laser light energy. We require that there be no more laser beams than there are radar beams as each laser beam requires a dedicated PAR beam to ensure accurate illimation of the target.

locale *PARTI_const_lasnum* = *PARTI_const_parnum* +
fixes
lasnum :: "ℕ"
assumes
lasnum_le_parnum: "*lasnum* ≤ *parnum*"

locale *PARTI_con_lasbeams* =
fixes
lasbeams :: "[ℝ, ℕ] → *volume*" ("lasbeams")

locale *PARTI_con_illuminates* = *PARTI_const_lasnum* + *PARTI_con_lasbeams* +
fixes
illuminates :: "ℝ → *volume*" ("illuminates")
defines
illuminates_def: "illuminates ≙ (λ *t* • (⋃ *l* | *l* ≤ *lasnum* • lasbeams *t l*))"

We also require that the width of the laser beams is no greater than the width of the radar beams.

locale *PARTI_const_lasbw* = *PARTI_const_parbw* +

fixes
lasbw :: "$\mathbb{R}$"
assumes
lasbw_le_parbw: "$|lasbw| \leq |parbw|$"

This ensures the the laser beams are always entirely contained in the corresponding radar beam.

lemma (**in** *PARTI_const_lasbw*) *lasb_in_parb*:
"*beam lasbw* $\vec{p}$ $\vec{p}'$ $\subseteq$ *beam parbw* $\vec{p}$ $\vec{p}'$"
proof

Follows directly from the smaller radius of the laser beams.

fix $\vec{q}$
assume *b1*: "$\vec{q} \in$ *beam lasbw* $\vec{p}$ $\vec{p}'$"
then obtain α $\vec{z}$ $\vec{r}$ **where**
b2: "$\alpha \in [0 \ldots 1]$" "$\vec{z} = \vec{p} + \alpha(\vec{p}' - \vec{p})$" **and**
b3: "*sqrt* $(\vec{r} \odot \vec{r}) \leq |lasbw|$" **and**
b4: "$\vec{r} \odot (\vec{p} - \vec{p}') = 0$" **and**
b5: "$\vec{q} = \vec{z} + \vec{r}$"
by (*auto simp add*: *beam_def*)
from *lasbw_le_parbw b3*
have *b6*: "*sqrt* $(\vec{r} \odot \vec{r}) \leq |parbw|$"
by (*auto*)
from *b2 b6 b4 b5*
show "$\vec{q} \in$ *beam parbw* $\vec{p}$ $\vec{p}'$"
apply (*simp add*: *beam_def*)
apply (*rule exI*)
apply (*inference*)
apply (*witness* "$\vec{z}_x$")
apply (*witness* "$\vec{z}_y$")
apply (*witness* "$\vec{z}_z$")
apply (*inference*)
apply (*rule refl*)
apply (*auto*)
apply (*witness* "$\vec{r}_x$")
apply (*witness* "$\vec{r}_y$")
apply (*witness* "$\vec{r}_z$")
apply (*auto*)
done
qed

The final outflow from the PARTI is in the form of targeting control messages to the companion missile system. These messages are of two forms: the *acquire* message says that a given object is now being illuminated and conveys its current position; while the *release* says that the object is no longer being illuminated.

datatype

message = *acquire* "*object* × *volume*" | *release* "*object*"

locale *PARTI_con_messages* =
fixes *messages* :: "ℝ → *message set*" ("messages")

locale *PARTI_consts* =
PARTI_const_parnum + *PARTI_const_lasnum* + *PARTI_const_parbw* +
PARTI_const_lasbw + *PARTI_const_origin* +
fixes
ASd :: "ℕ" **and**
SMax :: "ℝ" **and**
dist :: "[*point set*, *point set*] → ℝ" ("δ′(_, _′)" [0,0] 1000)
defines
dist_def: "δ(*obj*, *obj′*) ≡ ⨅$_0$ { $\vec{p}$ | $\vec{p}$ ∈ *obj* • ⨆$_{ASd}$ { $\vec{p}'$ | $\vec{p}'$ ∈ *obj′* • *pdist* $\vec{p}$ $\vec{p}'$ }}"

3.2 Inflows

The primary inflow from the environment is the tactical situation understood in terms of the kinematics of the nearby objects surrounding the ownship. In practice, the PARTI observes this situation through two channels, firstly through its own reflected radar beams and secondly through compiled track data transmitting from the general purpose radar system. In the abstract, we find it convenient to identify both channels into an oracular view of the tactical situation that identifies each nearby object with the volume of space it occupies at any given time.

locale *PARTI_env_objloc* =
fixes *objloc* :: "ℝ → (*object* ↔ *volume*)" ("objloc")

In addition, the PARTI makes use of the collection of threats identified by its companion IFF system.

locale *PARTI_env_threats* =
fixes *threats* :: "ℝ → *object set*" ("threats")

In order to ensure the protection of personnel and fragile assets, the PARTI may be required not to irradiate or illuminate into certain *prohibited_areas*

locale *PARTI_env_prohibited_areas* =
fixes *prohibited_areas* :: "ℝ → *volume*" ("prohibited_areas")

or even to shut down completely.

locale *PARTI_env_commanded* =
fixes *commanded* :: "ℝ → 𝔹" ("commanded")

The collection of inflows and outflows described above and depicted in Figure 2 constitute the system boundary identified for the purposes of arguing the safety of the PARTI system.

locale *PARTI_consts_sig* =
PARTI_const_parnum + *PARTI_const_lasnum* +

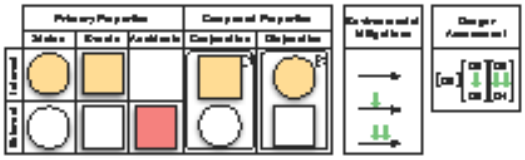


Figure 3: Accident scenario diagram conventions

PARTI_const_parbw + *PARTI_const_lasbw* +
PARTI_const_origin

locale *PARTI_env_sig* =
PARTI_env_commanded + *PARTI_env_objloc* +
PARTI_env_threats + *PARTI_env_prohibited_areas*

locale *PARTI_con_sig* =
PARTI_con_radiates + *PARTI_con_illuminates* + *PARTI_con_messages*

locale *PARTI_sig* =
PARTI_consts_sig + *PARTI_env_sig* + *PARTI_con_sig*

4 System Safety Requirements

Extensive analysis has identified a number of potential accident scenarios involving the radar, laser, and communications functions of the PARTI system.

The most serious of these scenarios are shown in Figures 4, 5, 6 and 7, using the conventions described in Figure 3. Factors in accident scenarios may comprise either events (square boxes) or states (round boxes); which may be either internal (coloured yellow) or external (coloured white). Accident events are shown in red. Factors may be aggregated either in conjunction (and-gate symbol) or disjunction (or-gate symbol) and sequenced with arrows. Green down-arrow labels on transition arrows indicate estimated levels of mitigation against the labelled transition. Finally the assessed default and mitigated DANGER LEVELS are presented in square brackets beside the relevant accident.

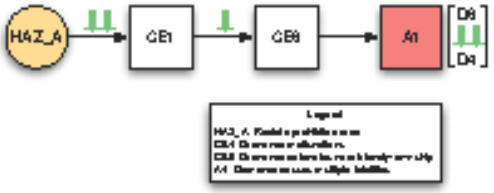
Note, that the danger levels associated with these accident scenarios yield a System Danger Level of D4, requiring a fully formal treatment of the System Architecture Analysis. This we perform with the use of Isabelle/HOL as described above.

These scenarios involve five different hazards.

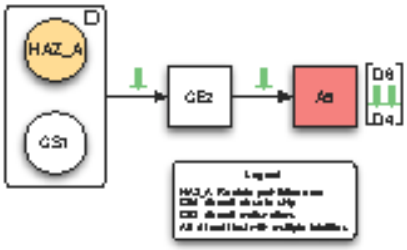
HAZ_A The PAR irradiates a prohibited area.

HAZ_B The laser illuminates a non-target object.

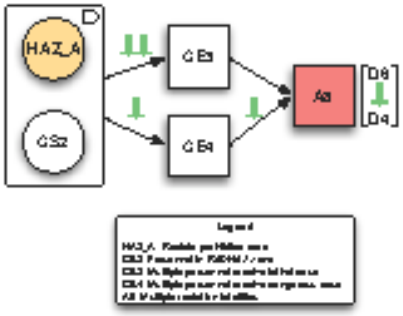
HAZ_C The PARTI sends an erroneous communication.



(a) AS_A_1: Radiated ordnance detonates

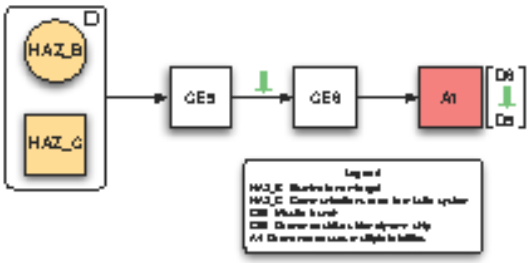


(b) AS_A_2: Radiated aircraft destroyed



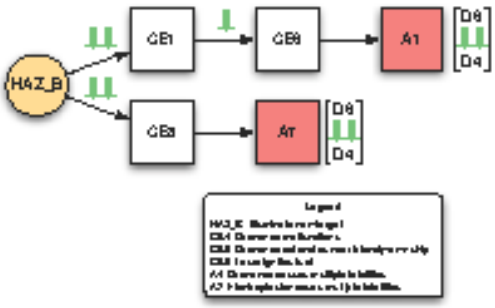
(c) AS_A_3: Radiated personnel die

Figure 4: Radar related Accident Scenarios

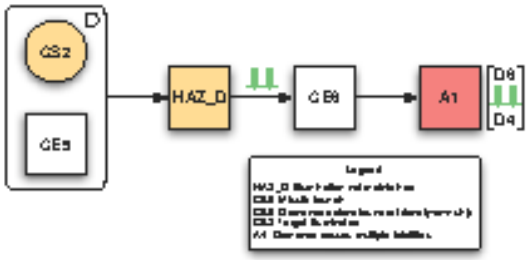


(a) AS_B_1: Friendly targeted and destroyed

(b) AS_B_2: Laser disables eye

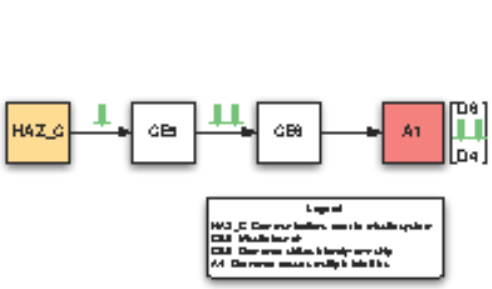


(c) AS_B_3: Laser cause explosion/fire

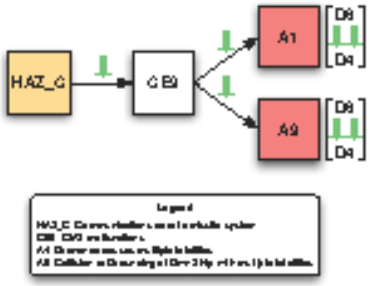


(d) AS_D_1: Laser loses target fix

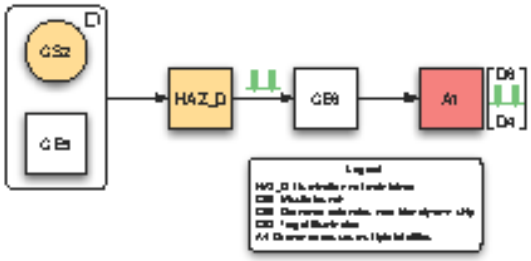
Figure 5: Laser related Accident Scenarios



(a) AS_C_1: Missile launched without target



(b) AS_C_2: PARTI brings down mission system



(c) AS_E_1: Friendly occludes target during targeting

Figure 6: Communications related Accident Scenarios

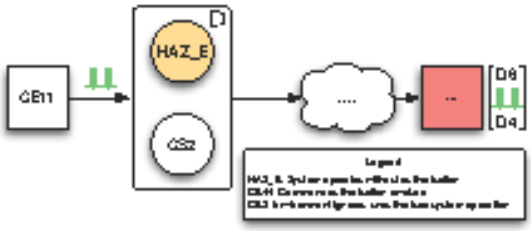


Figure 7: Command related Accident Scenarios

HAZ_D The laser fails to maintain target illumination.

HAZ_E The PARTI operates without command authorisation.

For the PARTI system to be acceptably safe, it must be engineered so as not to exhibit any of these hazards.

As a first step in this engineering process, we develop a collection of *System Safety Requirements* (SSRs) that are mutually consistent and complete with respect to the identified hazards. In principle, the SSRs may be the simple negation of the hazard conditions. In practise, interactions with other safety requirements and operational requirements can often mean that it desirable to adopt a more sophisticated collection of SSRs.

4.1 *HAZ_A*

In the case of *HAZ_A*, the situation is relatively straightforward. The negation of the hazard is that at all times the intersection of the radiated volume with the prohibited areas is empty. This creates some tension with the operational objective to track and illuminate all incoming threats, in particular for threats that are occluded by prohibited areas. However, it seems reasonable to believe that this factor will be taken into consideration when determining and setting the prohibited areas. We therefore resolve the conflict in favour of safety considerations and adopt the obvious SSR. This formalised as *SSR_A* below.

4.2 *HAZ_B*

As is clear from the various accident scenarios, *HAZ_B* has considerable interactions with the other hazards. There is also a strong tension with the operational imperative to track and illuminate all incoming threats. The PARTI system cannot be allowed to act safely by simply never operating the laser illuminator, as a simple negation of *HAZ_B* would allow. It is necessary for the SSRs to provide some direction as to how to resolve this tension.

Three scenarios of interest arise in relation to *HAZ_B*.

In the first the non-target is illuminated when there is no legitimate target and hence the laser should not be illuminating at all. This is a clear safety violation.

In the second, there is a legitimate target, but the laser is incorrectly oriented so that a non-target is instead illuminated. Again, this is a clear safety violation.

In the third, there is again a legitimate target, but the laser is occluded from illuminating it by a non-target which is therefore illuminated instead (see Figure 5a). In this case, the PARTI is pursuing a legitimate operational objective, that it cannot ignore, but is then confronted with a safety conflict that also cannot be ignored. We resolve this conflict by requiring the PARTI to communicate a *release* message before the non-target can occlude the target.

Thus all scenarios lead to the requirement that the laser never illuminate a non-target object. This is

formalised as *SSR_B* below. In addition, the third scenario leads to a requirement that the PARTI stops targeting a threat when it loses line-of-sight to the threat. This is formalised as *SSR_E* below.

4.3 *HAZ_C*

The communications hazard, *HAZ_C*, is particularly vague, especially in its role in *AS_C_2*, wherein some unspecified illegal communication causes the Combat Management System to malfunction. For the purposes of formulating a corresponding SSR, it is easiest to identify what constitutes a legal message.

This is straightforward for target *acquire* messages, which can only be sent when the PARTI is authorised to operate and for nearby objects that are threats with properly determined positions. This is formalised as *SSR_C* below.

Unfortunately, the legality of target *release* messages is again complicated by operational considerations. Clearly a target *release* can only be sent subsequent to a prior target *acquire* message and only provided that a target *release* has not already been sent. Additionally, operational considerations require that the PARTI does not communicate target *release* unless the target is indeed released and further that the target not be released until either: the target is destroyed, line-of-sight is lost to the target, or command authorisation to the PARTI is lost. Such operational considerations on target release overlap strongly with related safety requirements on laser illumination. Both considerations are formalised in *SSR_D* and *SSR_E* below. The import of *SSR_D* on PARTI behaviour is somewhat subtle, but the upshot is that a threat can only be targeted while the PARTI enjoys command authorisation and that a target *release* message must be sent if and when authorisation is lost.

4.4 *HAZ_D*

The operational need for the laser to maintain illumination has already been considered above. The safety need embodied by *HAZ_D* is essentially to ensure that the homing missile is guided away from friendly objects.

While the PARTI is targeting a threat, it must maintain illumination on the threat. This is formalised in *SSR_D* below.

The times at which the PARTI is considered to be targeting a threat are formalised in *SSR_B*, *SSR_D*, and *SSR_E* below.

4.5 *HAZ_E*

The accident scenarios arising from *HAZ_E* are essentially those that may occur during properly authorised operation, with the distinction that many operational mitigations will (by definition) not be in place. This situation is depicted in Figure 7 by the use the "cloud bubble" to represent the repetition of the operational accident scenarios.

The primary SSR arising from *HAZ_E* is that neither the radar nor the laser equipment should be powered without authorisation. This is formalised in *SSR_F* below.

More subtly, we require that target *acquire* messages are not sent without command authorisation, formalised by *SSR_C* and *SSR_D* below, and that a target *release* message is sent for all current targets when command authorisation is revoked, formalised by *SSR_D* below.

4.6 The formal SSRs

Before stating the SSRs formally, it is convenient to clarify some of the informal notions used in the discussion above.

An object is nearby (is_nearby) if it is in the domain of objloc, i.e. it is visible.

An object is a threat (is_threat) if it is nearby and it has been identified as a threat.

An object is a target (is_target) from the time a target acquire message is sent to the time the corresponding target release message is sent.

An object is friendly (is_friendly) if it is nearby and is not identified as a threat.

An object is in sight (in_sight) if it is possible to shine a laser beam on it without illuminating any other object.

locale *PARTI_defs* = *PARTI_sig* + *PARTI_consts_sig* +
fixes
is_nearby :: "*object* → ℝ → 𝔹" ("is_nearby") **and**
is_threat :: "*object* → ℝ → 𝔹" ("is_threat") **and**
is_target :: "*object* → ℝ → 𝔹" ("is_target") **and**
is_friendly :: "*object* → ℝ → 𝔹" ("is_friendly") **and**
in_sight :: "*object* → ℝ → 𝔹" ("in_sight")
defines
nearby_objects_def: "is_nearby ≙ (λ *obj t* • *obj* ∈ dom (objloc *t*))" **and**
is_threat_def: "is_threat *obj* ≙ (λ *t* • is_nearby *obj t* ∧ *obj* ∈ threats *t*)" **and**
is_target_def:
"is_target *obj* ≙ (λ *t* • (∃ t_0 *v* • $t_0 \le t$ ∧ *acquire* (*obj*, *v*) ∈ messages t_0 ∧
(∀ *s* | *s* ∈ ⌈t_0 . . *t*⌉ • *release obj* ∉ *messages s*)))" **and**
is_friendly_def: "is_friendly ≙ (λ *obj t* • is_nearby *obj t* ∧ *obj* ∉ threats *t*)" **and**
in_sight_def:
"in_sight *obj* ≙ (λ *t* • is_nearby *obj t* ∧
(∀ *obj′* | is_nearby *obj′ t* ∧ *obj′* ≠ *obj* •
(objloc *t*)·*obj′* ∩ (⋂ *p* | *p* ∈ (objloc *t*)·*obj* • *beam lasbw origin p*) = ∅))"

locale *OC* = *PARTI_defs* +
assumes
A: "(∀ *t t′ obj* • *t* < *t′* ∧ (∀ *s* | *s* ∈ ⌈*t* . . *t′*⌉ • is_nearby *obj s*) ⇒
δ((objloc *t*)·*obj*, (objloc *t′*)·*obj*) ≤ *SMax*∗(*t′* − *t*))"

We identify six formal SSRs for the PARTI system.

SSR_A – no prohibited areas are irradiated.

SSR_B – no friendly object is illuminated.

SSR_C – each target acquired message gives the correct position of a current threat.

SSR_D – if an object is a target then the PARTI has command authorisation and the object is being illuminated.

SSR_E – if an object is not in sight then it is not a target.

SSR_F – if the PARTI does not have command authorisation then it is not radiating or illuminating.

locale *SSR* = *PARTI_defs* +
assumes
SSR_A [*rule_format*]:
"(∀ *t* • radiates *t* ∩ prohibited_areas *t* = ∅)" **and**
SSR_B [*rule_format*]:
"(∀ *t obj* • is_friendly *obj t* ⇒
(objloc *t*)·*obj* ∩ illuminates *t* = ∅)" **and**
SSR_C [*rule_format*]:
"(∀ *t obj v* • *acquire* (*obj*, *v*) ∈ messages *t* ⇒
is_threat *obj t* ∧ *v* ⊆ (objloc *t*)·*obj*)" **and**
SSR_D [*rule_format*]:
"(∀ *t obj* • is_target *obj t* ⇒
commanded *t* ∧ (objloc *t*)·*obj* ∩ illuminates *t* ≠ ∅)" **and**
SSR_E [*rule_format*]:
"(∀ *t obj* • ¬(in_sight *obj t*) ⇒ ¬(is_target *obj t*))" **and**
SSR_F [*rule_format*]:
"(∀ *t* • ¬(commanded *t*) ⇒ radiates *t* = ∅ ∧ illuminates *t* = ∅)"

5 The Safety Architecture

The component interfaces of the the PARTI architecture are shown in Figure 8.

Three primary subsystems are required to support the desired functionality, corresponding essentially to the three outflows of the system boundary.

The Radar subsystem provides precision radar tracking of know threats and illuminated targets. The Laser subsystem provides target fix illumination in support of ownship's SAM system. The Control subsystem maintains the system's situational awareness database and co-ordinates the behaviour of the PAR, Laser and SAM subsystems.

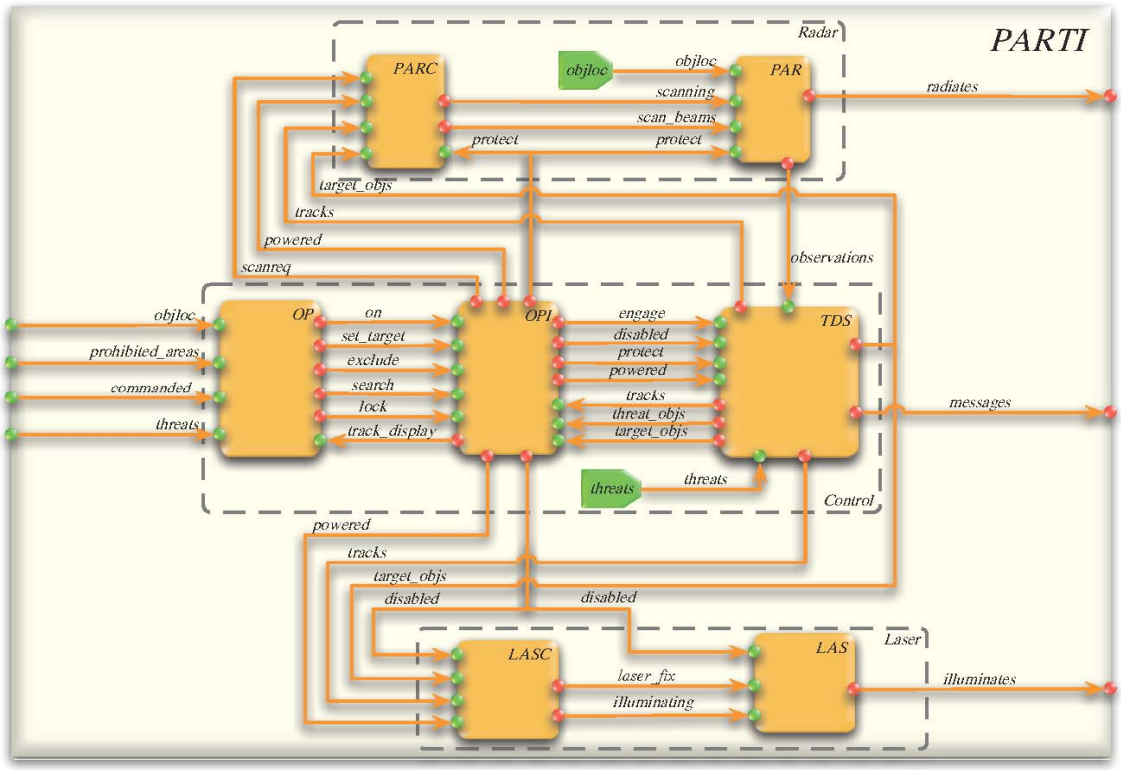


Figure 8: PARTI Architecture

The aim of the safety architecture is to provide protective measures and redundancy to a degree appropriate to the identified dangers and in light of the associated costs and benefits. We explain and justify these measures by consideration of the respective SSRs.

5.1 *Safety features of the architecture*

SSR_A

The PAR subsystem is required to protect designated prohibited areas. In support of this, we separate the PAR into two components, an actuator component *PAR* providing the core functionality and an electronic control component *PARC* for directing the behaviour of the PAR. Both components are required to protect the prohibited areas, the *PARC* by directing the *PAR* beams so as to avoid the prohibited areas and the *PAR* by disabling any beams that are directed toward a prohibited area.

Also of concern is the potential for the wrong prohibited areas to be transmitted to the PAR subsystem. This is guarded against by the vigilance of the operator component *OP* in checking feedback of the set areas on the operator display maintained by the operator interface *OPI*. It is assumed that the prohibited areas are static over time frames in the order of minutes to hours, so that this task is humanly feasible.

A possible enhancement of this operator function would be to also provide display feedback of the actual pattern of radar beams, but it is not thought that the added complexity and cost is justified given the strong potential for common-mode failures of the PAR subsystem in the directing and reporting of the beams. The fielding of a completely independent radar sensor array is obviously infeasible.

SSR_B

The Laser subsystem is required not to illuminate any friendly objects.

The primary responsibility for ensuring this, is invested in the Control subsystem as it can only be performed successfully in the presence of comprehensive situational awareness and may in some situations require co-ordination with the SAM system. As a redundancy measure, the Control subsystem is partitioned into an electronic command and control component and an operator component, supported by an operator interface component.

The Target Discrimination and Selection (*TDS*) component directs the targeting tasks of Laser subsystem, ensuring that the Laser is only directed to illuminate valid threats and ensuring that, when a friendly object interferes with line-of-sight to the target, illumination is ceased and a target *release* message set.

As a protective measure, the *OP* component is provided with appropriate Laser targeting feedback on the track display and is provided with controls for disabling any laser beam that is threatening to illuminate a friendly object. The task of responding to an operator cut-off command is distributed to both the *LASC* and the *LAS* components, providing some further measure of redundancy. The *TDS* component also responds by sending an appropriate target *release* message.

SSR_C

The targeting sequence is commenced when the *OP* component, having determined the existence and position of a threat, orders the PARTI to begin targeting the threat. The *TDS* component responds by checking that the object to be targeted is indeed a valid threat. If so, it sends a target *acquire* message and orders the Laser subsystem to commence illumination of the target.

The main alternatives to this approach would be to allow the *TDS* full autonomy in target selection or to provide a more elaborate operator-based target selection protocol, perhaps requiring concurrence from multiple operators. The first alternative may be forced in an environment where threat response times are required to be significantly less than a few seconds, but the added protection of operator initiation is considered valuable in the specified environment. Equally, a more elaborate operator protocol may be of value in an environment where response times measured in minutes are acceptable, but are considered an unnecessary overhead in the specified environment.

SSR_D

This SSR has two aspects, the requirement to illuminate a target from the time the target *acquired* message is sent and the requirement to send a target *release* message when illumination ceases, in particular when command authorisation is revoked and the PARTI is turned off.

Target illumination is maintained primarily by the *LAS* component, but it requires control support from the *LASC* component and tracking support from the Radar and Control subsystems. Such distributed safety responsibilities are not ideal, but are clearly necessary in this case as the function absolutely requires this situational awareness support. The provision of protective measures or redundancy is problematic in this case. Multiple redundant lasers could be used, but this is not ideal for the target illumination function and would still be subject to common-mode failures in the control support functions.

The transmission of target *release* messages is the primary responsibility of the *TDS* component. There is no practicable method of providing redundancy in this function.

SSR_E

The primary responsibility for detect loss of line-of-sight to a target lies with the *TDS* component, with support from the Radar subsystem in providing situational awareness. As a protective measure, the *OP* is given the ability to disable lasers that threaten to illuminate a friendly.

SSR_F

The primary responsibility for shutdown when command authorisation is revoked is given to the *OP* component. Shutdown is achieved by removing power from the subsystems.

5.2 The component interfaces

The *PAR* component produces the externally visible scanning beams radiates and analyses the reflected beams to produce a collection of observed object positions or plots.

locale *PAR_con_plots* =
  fixes *plots* :: "$\mathbb{R} \to (object \leftrightarrow volume)$" ("plots")
locale *PAR_con_sig* = *PARTI_con_radiates* + *PAR_con_plots*

The *PARC* component determines if there is currently command authorisation for radar scanning scanning and calculates the necessary scanning beam for each PAR element scan_beams.

locale *PARC_con_scanning* =
  fixes *scanning* :: "$\mathbb{R} \to \mathbb{B}$" ("scanning")
locale *PARC_con_scan_beams* =
  fixes *scan_beams* :: "$[\mathbb{R}, \mathbb{N}] \to volume$" ("scan_beams")
locale *PARC_con_sig* = *PARC_con_scanning* + *PARC_con_scan_beams*

The *TDS* component is the automated nerve centre of the PARTI. It maintains a central database of tracking information for observed objects tracks and identifies which tracks correspond to threats threat_objs. It responds to target engage requests, keeping track of current targets target_objs and communicating with the missile system messages.

locale *TDS_con_tracks* =
  fixes *tracks* :: "$\mathbb{R} \to (object \leftrightarrow volume)$" ("tracks")
locale *TDS_con_target_objs* =
  fixes *target_objs* :: "$\mathbb{R} \to \mathbb{N} \leftrightarrow object$" ("target_objs")
locale *TDS_con_threat_objs* =
  fixes *threat_objs* :: "$\mathbb{R} \to object\ set$" ("threat_objs")
locale *TDS_con_sig* = *PARTI_con_messages* + *TDS_con_target_objs* + *TDS_con_tracks* + *TDS_con_threat_objs*

locale *OPI_con_powered* =
  fixes *powered* :: "$\mathbb{R} \to \mathbb{B}$" ("powered")
locale *OPI_con_scanreq* =
  fixes *scanreq* :: "$\mathbb{R} \to volume$" ("scanreq")
locale *OPI_con_protect* =
  fixes *protect* :: "$\mathbb{R} \to volume$" ("protect")
locale *OPI_con_engage* =
  fixes *engage* :: "$\mathbb{R} \to object\ set$" ("engage")
locale *OPI_con_disabled* =
  fixes *disabled* :: "$\mathbb{R} \to \mathbb{N}\ set$" ("disabled")
locale *OPI_con_track_display* =
  fixes *track_display* :: "$\mathbb{R} \to ((object \leftrightarrow volume) \times object\ set \times (\mathbb{N} \leftrightarrow object) \times volume)$" ("track_display")
locale *OPI_con_sig* =
  OPI_con_powered + *OPI_con_scanreq* + *OPI_con_protect* + *OPI_con_engage* +
  OPI_con_disabled + *OPI_con_track_display*

locale *OP_con_on* =

fixes *on* :: "$\mathbb{R} \to \mathbb{B}$" ("on")
locale *OP_con_set_target* =
fixes *set_target* :: "$\mathbb{R} \to$ *object set*" ("set_target")
locale *OP_con_exclude* =
fixes *exclude* :: "$\mathbb{R} \to$ *volume*" ("exclude")
locale *OP_con_search* =
fixes *search* :: "$\mathbb{R} \to$ *volume*" ("search")
locale *OP_con_lock* =
fixes *lock* :: "$\mathbb{R} \to \mathbb{N}$ *set*" ("lock")
locale *OP_con_sig* =
OP_con_on + *OP_con_set_target* + *OP_con_exclude* + *OP_con_search* + *OP_con_lock*

locale *LASC_con_illuminating* =
fixes *illuminating* :: "$\mathbb{R} \to \mathbb{B}$" ("illuminating")
locale *LASC_con_laser_fix* =
fixes *laser_fix* :: "$\mathbb{R} \to (\mathbb{N} \leftrightarrow$ *point*$)$" ("laser_fix")
locale *LASC_con_sig* = *LASC_con_illuminating* + *LASC_con_laser_fix*

locale *LAS_con_sig* = *PARTI_con_illuminates*

locale *PAR_env_sig* = *PARTI_env_objloc* + *OPI_con_protect* + *PARC_con_scanning* + *PARC_con_scan_beams*
locale *PARF_env_sig* = *OPI_con_protect* + *PARC_con_scan_beams*
locale *PARC_env_sig* =
OPI_con_protect + *OPI_con_powered* + *OPI_con_scanreq* +
TDS_con_target_objs + *TDS_con_tracks*
locale *TDS_env_sig* =
PAR_con_plots + *PARTI_env_threats* +
OPI_con_powered + *OPI_con_protect* + *OPI_con_engage* + *OPI_con_disabled*
locale *OPI_env_sig* =
TDS_con_target_objs + *TDS_con_tracks* + *TDS_con_threat_objs* +
OP_con_on + *OP_con_set_target* + *OP_con_exclude* + *OP_con_search* + *OP_con_lock*
locale *OP_env_sig* =
PARTI_env_commanded + *PARTI_env_objloc* + *PARTI_env_threats* +
PARTI_env_prohibited_areas + *OPI_con_track_display*
locale *LASC_env_sig* =
OPI_con_powered + *OPI_con_disabled* +
TDS_con_target_objs + *TDS_con_tracks*
locale *LAS_env_sig* = *LASC_con_laser_fix* + *OPI_con_disabled* + *LASC_con_illuminating*

5.3 Formal CSRs

The *PARC* component calculates beam directions to maintain contact with current targets and threat identified by the operator. The *targfix* algorithm is used to center the beams on the object.

The controller also ensures that no beams intersect the prohibited areas (protect and that the radar is

turned off (scanning) when command authorisation is revoked (*powered*).

locale *PARTI_arch_consts* =
 fixes
  targfix :: "*volume* → *point*"
 assumes
  targfix_targets [*rule_format*]: "(∀ *v* • *targfix v* ∈ *v*)"

locale *PARC_CSR* =
 PARTI_consts + *PARTI_arch_consts* + *PARC_con_sig* + *PARC_env_sig* +
 assumes
  PARC_A [*rule_format*]:
   "(∀ *t* • scanning *t* = powered *t*)" **and**
  PARC_B [*rule_format*]:
   "(∀ *t l obj* •
     (*l* ↦ *obj*) ∈ target_objs *t* ∧
     protect *t* ∩ *beam parbw origin* (*targfix* ((tracks *t*)·*obj*)) = ∅ ⇒
      scan_beams *t l* = *beam parbw origin* (*targfix* ((tracks *t*)·*obj*)))" **and**
  PARC_C [*rule_format*]:
   "(∀ *t r* • scan_beams *t r* ∩ protect *t* = ∅)"

The *PAR* component projects the radar beams as directed by *PARC*, provided that it is powered (scanning) and the beam does not intersect with the prohibited areas (protect).

locale *PAR_CSR* = *PARTI_consts* + *PAR_con_sig* + *PAR_env_sig* +
 assumes
  PAR_A [*rule_format*]:
   "(∀ *t r* • parbeams *t r* =
     if scanning *t* ∧ *r* ≤ *parnum* ∧ protect *t* ∩ scan_beams *t r* = ∅ then
      scan_beams *t r*
     else
      ∅
     fi)" **and**
  PAR_B [*rule_format*]:
   "(∀ *t* • plots *t* =
     (λ *obj* | *obj* ∈ dom (objloc *t*) ∧ (objloc *t*)·*obj* ∩ radiates *t* ≠ ∅ •
      (objloc *t*)·*obj* ∩ radiates *t*))"

The *TDS* is the nerve centre of the PARTI system. Its main safety function is to ensure that only legitimate threats are targeted and that the target illuminating process is enacted safely. The target_objs tells the Laser subsystem what threats to target with which lasers, an object is being targeted if and only if a laser has been allotted in target_objs to target it.

The CSRs for *TDS* are as follows.

TDS_A : that a target *acquire* message is setting when an object begins being targeted and that a target *release* message is sent when it finishes.

TDS_B : that each target *acquire* message is eventually followed by a *release* message.

TDS_C : that the *TDS* correctly keep track of the plots made by the radar.

TDS_D : that each enabled laser is targeting an observed threat.

TDS_E : that targeting commence when so ordered by the operator (*engage*) and that it continue, using a single laser, while there is line-of-sight to the target and command authorisation.

locale *TDS_defs* = *PARTI_consts* + *PARTI_arch_consts* + *TDS_con_sig* + *TDS_env_sig* +
fixes
observed :: "$\mathbb{R} \to$ *object set*" ("observed") **and**
line_of_sight :: "[*object*, $\mathbb{R}$] $\to \mathbb{B}$" ("line_of_sight") **and**
target_times :: "*object* $\to \mathbb{R}$ *set*" ("target_times") **and**
target_ints :: "*object* $\to (\mathbb{R} \times \mathbb{R}$ *option*) *set*" ("target_ints")
defines
observed_def: "observed $\widehat{=}$ (λ *t* • dom (plots *t*))" **and**
line_of_sight_def: "line_of_sight $\widehat{=}$ (λ *obj t* • *obj* $\in$ observed *t* $\wedge$
($\forall$ *obj'* | *obj'* $\in$ observed *t* $\wedge$ *obj'* $\neq$ *obj* •
(plots *t*)·*obj'* $\cap$ *beam lasbw origin* (*targfix* ((plots *t*)·*obj*)) = $\varnothing$) $\wedge$
protect *t* $\cap$ *beam parbw origin* (*targfix* ((plots *t*)·*obj*)) = $\varnothing$)" **and**
target_times_def: "target_times *obj* $\widehat{=}$ { τ | *obj* $\in$ ran (target_objs τ) }" **and**
target_ints_def: "target_ints *obj* $\widehat{=}$
{ *s t* | *s* < *t* $\wedge$ ⟦*s*...*t*⟧ $\subseteq$ target_times *obj* $\wedge$
($\forall$ *s'* • *s'* < *s* $\Rightarrow$ $\neg$(⟦*s'*...*s*⟧ $\subseteq$ target_times *obj*)) $\wedge$ *t* $\notin$ target_times *obj* •
(*s*, *Some t*) } $\cup$
{ *s* | ($\forall$ τ | *s* < τ • ⟦*s*...τ⟧ $\subseteq$ target_times *obj*) $\wedge$
($\forall$ *s'* • *s'* < *s* $\Rightarrow$ $\neg$(⟦*s'*...*s*⟧ $\subseteq$ target_times *obj*)) •
(*s*, *None*) }"

locale *TDS_CSR* = *PARTI_consts* + *PARTI_arch_consts* + *TDS_defs* +
assumes
TDS_A [*rule_format*]:
"($\forall$ *t* • messages *t* =
{ *obj t'* | (*t*, *t'*) $\in$ target_ints *obj* • *acquire* (*obj*, (plots *t*)·*obj*) } $\cup$
{ *obj s* | (*s*, *Some t*) $\in$ *target_ints obj* • *release obj* })" **and**
TDS_B [*rule_format*]:
"($\forall$ *obj* • target_times *obj* = ($\bigcup$ *s t* | (*s*, *Some t*) $\in$ target_ints *obj* • ⟦*s*...*t*⟧))" **and**
TDS_C [*rule_format*]:
"($\forall$ *t* • tracks *t* = plots *t*)" **and**
TDS_D [*rule_format*]:
"($\forall$ *t* • target_objs *t* $\in$ ({*0*..*lasnum*}\(disabled *t*)) $\rightarrowtail\!\!\!\!\!\!\rightarrow$ (threats *t* $\cap$ observed *t*))" **and**
TDS_E [*rule_format*]:
"($\forall$ *obj s t* • (*s*, *Some t*) $\in$ target_ints *obj* $\Rightarrow$
obj $\in$ engage *s* $\wedge$
($\forall$ τ • τ $\in$ ⟦*s*...*t*⟧ $\Rightarrow$

```
        powered τ ∧
        ((target_objs τ)˜)·obj = ((target_objs s)˜)·obj ∧
        line_of_sight obj τ))"

lemma (in TDS_CSR) target_ints_conv:
 "target_ints obj =
   { s t | s < t ∧ ⟦s...t⟧ ⊆ target_times obj ∧
     (∀ s' • s' < s ⇒ ¬(⟦s'...s⟧ ⊆ target_times obj)) ∧ t ∉ target_times obj •
     (s, Some t)}"
proof (auto simp add: target_ints_def)
 fix s assume b1 [rule_format]: "(∀ τ | s < τ • ⟦s...τ⟧ ⊆ target_times obj)" and
  "(∀ s' • s' < s ⇒ ¬(⟦s'...s⟧ ⊆ target_times obj))"
 from b1 [of "s+1"] have b2: "s ∈ target_times obj"
  by (auto simp add: cINTVLo_def)
 with TDS_B obtain t t' where b3: "(t, Some t') ∈ target_ints obj" and b4: "s ∈ ⟦t...t'⟧"
  by (auto)
 from b3 have "t' ∉ target_times obj"
  by (simp add: target_ints_def)
 with b1 [of "t'+1"] b4 show "false"
  by (auto simp add: cINTVLo_def subset_def)
qed
```

The *OPI* component ensures that the *OP* component is properly informed about the state of the PARTI and passes on operator directives to the appropriate subsystems.

```
locale OPI_CSR = PARTI_consts + OPI_con_sig + OPI_env_sig +
 assumes
  OPI_A [rule_format]:
   "(∀ t • track_display t = (tracks t, threat_objs t, target_objs t, protect t))" and
  OPI_B [rule_format]:
   "(∀ t • powered t = on t)" and
  OPI_C [rule_format]:
   "(∀ t • scanreq t = search t)" and
  OPI_D [rule_format]:
   "(∀ t • protect t = exclude t)" and
  OPI_E [rule_format]:
   "(∀ t • engage t = set_target t)" and
  OPI_F [rule_format]:
   "(∀ t • disabled t = lock t)"
```

As well as determining the objects to be targeted, the *OP* component parallels many of the safety functions of the *TDS*. It has the following CSRs.

OP_A : That the operator requests targeting of threats that have been identified as threat by the IFF as well as the *TDS*.

OP_B : That prohibited areas are protected from radar scanning.

OP_C : That the PARTI is turned of when command authorisation is revoked.

OP_D : That lasers are disabled to prevent illumination of non-targets.

locale *OP_defs* = *PARTI_consts* + *PARTI_arch_consts* + *OP_con_sig* + *OP_env_sig* +
fixes
dis_tracks :: "$\mathbb{R} \rightarrow (object \leftrightarrow volume)$" ("dis_tracks") **and**
dis_threat_objs :: "$\mathbb{R} \rightarrow object\ set$" ("dis_threat_objs") **and**
dis_target_objs :: "$\mathbb{R} \rightarrow \mathbb{N} \leftrightarrow object$" ("dis_target_objs") **and**
dis_observed :: "$\mathbb{R} \rightarrow object\ set$" ("dis_observed") **and**
dis_line_of_sight :: "$[\mathbb{N}, object, \mathbb{R}] \rightarrow \mathbb{B}$" ("dis_line_of_sight")
defines
dis_tracks_def: "dis_tracks $\widehat{=}$ ($\lambda\ t \bullet$ fst (track_display t))" **and**
dis_threat_objs_def: "dis_threat_objs $\widehat{=}$ ($\lambda\ t \bullet$ fst (snd (track_display t)))" **and**
dis_target_objs_def: "dis_target_objs $\widehat{=}$ ($\lambda\ t \bullet$ fst (snd (snd (track_display t))))" **and**
dis_observed_def: "dis_observed $\widehat{=}$ ($\lambda\ t \bullet$ dom (dis_tracks t))" **and**
dis_line_of_sight_def: "dis_line_of_sight $\widehat{=}$
($\lambda\ i\ obj\ t \bullet$
$obj \in$ dis_observed $t \wedge$
$(\forall\ obj' \mid obj' \in$ dis_observed $t \wedge obj' \neq obj \bullet$
(dis_tracks t)$\cdot obj' \cap$ *beam lasbw origin* (*targfix* ((dis_tracks t)$\cdot obj$)) $= \varnothing) \wedge$
prohibited_areas $t \cap$ *beam parbw origin* (*targfix* ((dis_tracks t)$\cdot obj$)) $= \varnothing$)"

locale *OP_CSR* = *OP_defs* +
assumes
OP_A [*rule_format*]:
"$(\forall\ t \bullet$ set_target $t \subseteq$ threats $t \cap$ dis_threat_objs t)" **and**
OP_B [*rule_format*]:
"$(\forall\ t \bullet$ exclude $t =$ prohibited_areas t)" **and**
OP_C [*rule_format*]:
"$(\forall\ t \bullet$ on $t =$ commanded t)" **and**
OP_D [*rule_format*]:
"$(\forall\ t\ i\ obj \bullet (i \mapsto obj) \in$ dis_target_objs $t \wedge \neg$ (dis_line_of_sight $i\ obj\ t$) $\Rightarrow$
$i \in$ lock t)"

The *LASC* directs the various lasers to ensure that they illuminate the targets identified by the *TDS*, provided that they have not been disabled by *OP*. It also powers down the lasers when command authorisation is revoked.

locale *LASC_CSR* =
PARTI_consts + *PARTI_arch_consts* + *LASC_con_sig* + *LASC_env_sig* +
assumes
LASC_A [*rule_format*]:
"$(\forall\ t \bullet$ illuminating $t =$ powered t)" **and**
LASC_B [*rule_format*]:
"$(\forall\ t \bullet$ laser_fix $t =$

(λ *l* | *l* ≤ *lasnum* ∧ *l* ∉ disabled *t* ∧ *l* ∈ dom (target_objs *t*) ∧
(target_objs *t*)·*l* ∈ dom (tracks *t*) •
targfix ((tracks *t*)·((target_objs *t*)·*l*))))"

The *LAS* component projects the laser beams as directed by *LASC*, provided that they are not disabled by *OP* and that command authorisation is not revoked.

locale *LAS_CSR* = *PARTI_consts* + *LAS_con_sig* + *LAS_env_sig* +
assumes
LAS_A [*rule_format*]:
"(∀ *t l* • lasbeams *t l* =
if *l* ∈ dom (laser_fix *t*) ∧ *l* ∉ disabled *t* ∧ illuminating *t* then
beam lasbw origin ((laser_fix *t*)·*l*)
else
∅
fi)"

locale *CSR* =
OC +
PAR_CSR + *PARC_CSR* +
TDS_CSR + *OPI_CSR* + *OP_CSR* +
LASC_CSR + *LAS_CSR*

6 Verifying the architecture

The collection of CSRs described above ensure satisfaction of the SSRs described in Section 4.6.

interpretation *CSR* ⊆ *SSR*
apply (*unfold SSR_axioms_def*)
apply (*inference*)
proof −

6.1 *SSR_A*

The *PAR_A* CSR ensures that protect areas are never irradiated. The *OPI_D* and *OP_B* CSRs ensure that the protect areas are exactly the prohibited_areas.

fix *t*
from *PAR_A*
show "(⋃ *r* | *r* ≤ *parnum* • parbeams *t r*) ∩ prohibited_areas *t* = ∅"
by (*auto simp add*: *OPI_D OP_B*)
next

6.2 *SSR_B*

Suppose that *obj* is friendly and is nearby.

```
fix t obj
assume "obj ∉ threats t" "obj ∈ dom (objloc t)"
note b1 = this
```

Note that *obj* is not in the range of target_objs, since *TDS_D* ensures that these are all threats.

```
from b1 TDS_D [of t]
have "obj ∉ ran (target_objs t)"
 apply (fspace) — properties of injections
 apply (auto)
 done
note b2 = this
```

We show that *obj* is not illuminated by case analysis over the existence of targets and the power status of the PARTI.

```
show "(objloc t)·obj ∩ (⋃ l | l ≤ lasnum • lasbeams t l) = ∅"
proof –
  apply_end (cases "powered t")
  apply_end (cases "target_objs t = ∅")
```

If there are no targets, *LASC_B* ensures that no laser beams are projected.

```
 assume "target_objs t = ∅"
 with LASC_B [of t]
 have "laser_fix t = ∅"
  by (auto simp add: glambda_mem)
 with LAS_A
 show ?thesis
  by (auto)
next
```

If there is no power, *LASC_A* and *LAS_A* ensure that no laser beams are projected.

```
 assume "¬(powered t)"
 show ?thesis
  apply (simp ! add: LAS_A LASC_A)
  apply (auto)
  done
next
```

Finally, suppose that the lasers are powered and that there are targets acquired.

```
 assume "powered t" "target_objs t ≠ ∅"
 note c1 = this
```

We know that *TDS* ensures that all targets have clear line-of-sight.

have "(∀ *i obj* • (*i* ↦ *obj*) ∈ target_objs *t* ⇒ line_of_sight *obj t*)"
proof −

Follows from *TDS_B*, *TDS_D* and *TDS_E*.

apply_end (*inference*)
fix *i obj*
assume *d1*: "(*i* ↦ *obj*) ∈ target_objs *t*"
then obtain *s s'* **where**
d2: "(*s*, *Some s'*) ∈ target_ints *obj*" **and**
d3: "*t* ∈ [*s*...*s'*)"
proof −
from *d1* **have** "*t* ∈ target_times *obj*"
by (*auto simp add*: *target_times_def*)
with *TDS_B* **show** *?thesis*
by (*auto*)
qed
from *TDS_D* [*of* "*t*"] *d1*
have *d4*: "((target_objs *t*)˘)·*obj* = *i*"
by (*rule pinj_inv_beta*)
from *TDS_E* [*OF d2*] *d3 d4*
show "line_of_sight *obj t*"
by (*auto*)
qed
note *c2* [*rule_format*] = *this*
have *c3*: "(∀ *l* $\vec{p}$ • (*l* ↦ $\vec{p}$) ∈ laser_fix *t* ⇒ (objloc *t*)·*obj* ∩ *beam lasbw origin* $\vec{p}$ = ∅)"
proof −
apply_end (*inference*)
fix *l* $\vec{p}$
assume *d1*: "(*l* ↦ $\vec{p}$) ∈ laser_fix *t*"
with *LASC_B* [*of t*] **have**
d2: "*l* ∈ dom (target_objs *t*)" **and**
d3: "$\vec{p}$ = *targfix* ((tracks *t*)·((target_objs *t*)·*l*))"
by (*auto simp add*: *glambda_mem*)
from *TDS_D* [*of t*, *THEN pinj_appl*, *OF d2*] *b2* **have** *d4*: "*obj* ≠ (target_objs *t*)·*l*"
by (*auto*)
from *d2 TDS_D* [*of t*] *lasnum_le_parnum*
have *d5*: "0 ≤ *l*" **and** *d6*: "*l* ≤ *lasnum*" **and** *d6'*: "*l* ≤ *parnum*"
apply (*fspace*)
apply (*auto*)
done
from *pinj_appl* [*OF TDS_D* [*of* "*t*"], *OF d2*, *THEN c2*] *d4*
have *d7*: "protect *t* ∩ *beam parbw origin* $\vec{p}$ = ∅"
by (*auto simp add*: *line_of_sight_def d3 TDS_C*)
from *lasbw_le_parbw* **have** *d8*: "*beam lasbw origin* $\vec{p}$ ⊆ *beam parbw origin* $\vec{p}$"

```
 apply (auto simp add: beam_def)
 apply (rule exI)
 apply (inference)
 apply (intro exI)
 apply (inference)
 apply (rule refl)
 apply (auto intro!: exI)
 done
from d7 TDS_D [of t, THEN pinj_appl, OF d2] PARC_B
have d9: "scan_beams t l = beam parbw origin p⃗"
 by (simp add: d3)
show "(objloc t)·obj ∩ beam lasbw origin p⃗ = ∅"
proof (cases "obj ∈ observed t")
 assume e1: "obj ∈ observed t"
 with pinj_appl [OF TDS_D [of "t"], OF d2, THEN c2] d4
 have e2: "(plots t)·obj ∩ beam lasbw origin p⃗ = ∅"
  by (auto simp add: line_of_sight_def d3 TDS_C)
 from e1 have "(plots t)·obj = (objloc t)·obj ∩ radiates t"
  by (simp add: PAR_B glambda_beta observed_def glambda_dom)
 then have "(objloc t)·obj ∩ beam parbw origin p⃗ ⊆ (plots t)·obj"
  apply (simp add: radiates_def PAR_A PARC_A c1 d9)
  apply (auto)
  apply (witness "beam parbw origin p⃗")
  apply (auto)
  apply (witness "l")
  apply (simp add: order_trans [OF d6 lasnum_le_parnum] d9 d7)
  done
 then have
  "(objloc t)·obj ∩ beam parbw origin p⃗ ∩ beam lasbw origin p⃗ ⊆
    (plots t)·obj ∩ beam lasbw origin p⃗"
  by (auto)
 with d8 have
  "(objloc t)·obj ∩ beam lasbw origin p⃗ ⊆
    (plots t)·obj ∩ beam lasbw origin p⃗"
  by (auto)
 with e2 show "(objloc t)·obj ∩ beam lasbw origin p⃗ = ∅"
  by (auto)
next
 assume e1: "obj ∉ observed t"
 then have "(objloc t)·obj ∩ radiates t = ∅"
  by (simp add: observed_def PAR_B glambda_dom b1)
 then have "(objloc t)·obj ∩ beam parbw origin p⃗ = ∅"
  apply (rule contrapos_pp)
  apply (insert d7)
  apply (simp add: nempty_conv)
  apply (simp add: radiates_def PAR_A PARC_A c1 d9 [symmetric])
```

```
    apply (elim exE)
    apply (auto intro!: exI simp add: d6')
    done
  with d8 show "(objloc t)·obj ∩ beam lasbw origin p⃗ = ∅"
   by (auto)
 qed
qed
```

From this we can deduce that *obj* is not being illuminated.

```
have "(∀ l | l ≤ lasnum • (objloc t)·obj ∩ lasbeams t l = ∅)"
proof –
```

From *c2*, we know that the targeting beams, as fixed by *LASC_B*, do not illuminated the *plotted* position of *obj*, that is the part of *obj*'s body (if any) that is irradiated by a radar beam.

From *PARC_B*, we know that there is a radar beam that entirely encompasses each laser beam, so that every object illumination event must be observed by the corresponding radar beam. Hence, *obj* cannot be illuminated at an unobserved part of its body.

```
  apply_end (inference)
  fix l
  assume "l ≤ lasnum"
  with c3 [rule_format, of l "targfix ((tracks t)·((target_objs t)·l))"]
  show "(objloc t)·obj ∩ lasbeams t l = ∅"
   apply (simp add: LAS_A LASC_A LASC_B c1 dsub_dom glambda_dom)
   apply (inference)
   apply (simp add: dsub_beta glambda_beta glambda_mem)
   done
 qed
```

SSR_B now follows immediately.

```
 then
 show ?thesis
  by (auto)
qed
next
```

6.3 *SSR_C*

Suppose that the message *acquire* (*obj*, *v*) is sent at time *t*.

```
fix t obj v
assume b1: "acquire (obj, v) ∈ messages t"
have b2: "observed t ⊆ dom (objloc t)"
 apply (simp add: observed_def PAR_B glambda_dom)
 apply (auto)
```

```
  done
from b1 TDS_D [of t]
```

From *TDS_A* and *TDS_D*, we know that *obj* is an observed threat.

```
have "obj ∈ threats t ∩ observed t"
  apply (fspace)
  apply (auto simp add: TDS_A target_ints_conv cINTVLo_def target_times_def)
  done
note b3 = this
with b2
```

From *PAR_B*, we know that observed objects are nearby objects, so that *obj* is a nearby threat as required.

```
show "obj ∈ threats t" "obj ∈ dom (objloc t)"
  by (auto)
note b4 = this
from b3 have b5: "(objloc t)·obj ∩ radiates t ≠ ∅"
  by (simp add: observed_def PAR_B glambda_dom)
```

From *TDS_A*, we know that *v* is the plotted position of *obj* and from *PAR_B*, we know that the plotted position of *obj* is the part of the real position irradiated by the radar beams.

```
from b1
have "v = (plots t)·obj"
  by (simp add: TDS_A)
also from b4 b5 have "... = (objloc t)·obj ∩ radiates t"
  by (simp add: PAR_B glambda_beta)
finally show "v ⊆ (objloc t)·obj"
  by (auto)
```

6.4 *SSR_D*

Now suppose that it is some later (than *t*) time *s* and that the target *obj* has not yet been released.

```
fix s
assume "t ≤ s"
note b6 = this
assume "(∀ τ | τ ∈ [t...s] • release obj ∉ messages τ)"
note b6' [rule_format] = this
```

Since the target *release* message has not yet been sent, *obj* is still being targeted. From *TDS_E*, *OPI_B* and *OP_C* we know that the PARTI still enjoys command authorisation.

```
from b1 obtain t' where b7: "(t, Some t') ∈ target_ints obj" and b7': "t < t'"
  by (auto simp add: TDS_A target_ints_conv)
then have b8: "release obj ∈ messages t'"
```

```
  by (auto simp add: TDS_A)
with b6' [of t'] b7' have b9: "s < t'"
  by (auto simp add:  cINTVLc_def linorder_not_less [symmetric])
from b6 b9 TDS_E [OF b7]
show "commanded s"
  by (auto simp add: cINTVLo_def OPI_B OP_C)
note b6'' = this
from b9 b7 b6 have "s ∈ target_times obj"
  by (auto simp add: TDS_B cINTVLo_def)
then have b10: "obj ∈ ran (target_objs s)"
  by (simp add: target_times_def)
then
```

Since *obj* is still being targeted, there is a laser, say *l*, assigned to illuminating it.

```
obtain l where "(l ↦ obj) ∈ target_objs s"
  by (auto)
```

From *LAS_A* and *LASC_A* we know that *l*'s laser beam is targeted on *targfix* of its current tracking position. From *TDS_C*, we know that the current tracking position is the current plotted position.

```
note b10' = this
with TDS_D [of s] lasnum_le_parnum
have
  b11a: "l ≤ lasnum" and b11b: "l ∉ disabled s" and
  b11c: "(target_objs s)·l = obj" and b11d: "l ≤ parnum"
  apply (fspace)
  apply (auto simp add: subset_def Domain_def functional_beta)
  done
from b6'' b10' TDS_D [of s]
have "lasbeams s l = beam lasbw origin (targfix ((plots s)·obj))"
  apply (fspace)
  apply (simp add: LAS_A LASC_A OPI_B OP_C LASC_B TDS_C observed_def)
  apply (simp add: dsub_dom glambda_dom dsub_beta glambda_beta beam_def
    b11a b11b b11c)
  apply (auto)
  done
then have "targfix ((plots s)·obj) ∈ lasbeams s l"
  apply (simp add: beam_def)
  apply (rule exI)
  apply (rule conjI)
  apply (witness "(targfix ((plots s)·obj))_x")
  apply (witness "(targfix ((plots s)·obj))_y")
  apply (witness "(targfix ((plots s)·obj))_z")
  apply (rule conjI)
  apply (rule refl)
  apply (witness "1::ℝ")
```

```
  apply (auto simp add: cINTVLc_def pscale_def)
  apply (cases "targfix ((plots s)·obj)")
  apply (cases "origin")
  apply (auto simp add: prod_plus_def prod_minus_def)
  apply (witness "0::ℝ")
  apply (witness "0::ℝ")
  apply (witness "0::ℝ")
  apply (auto simp add: dprod_def)
  done
with targfix_targets [of "(plots s)·obj"]
have b12: "(plots s)·obj ∩ lasbeams s l ≠ ∅"
  by (auto)
have b12a: "(plots s)·obj ⊆ (objloc s)·obj"
proof −
  from b10 TDS_D [ of "s"] have "obj ∈ observed s"
    apply (fspace)
    apply (auto)
    done
  then show ?thesis
    apply (simp add: observed_def PAR_B glambda_beta glambda_dom)
    apply (auto)
    done
qed
```

Thus the laser beam must intersect the plotted position (at least at *targfix* ((plots *s*)·*obj*)) and as the plotted position is a subset of the real position, so also must the laser beam intersect the real position.

```
with b12
have "(objloc s)·obj ∩ lasbeams s l ≠ ∅"
  by (auto)
```

Hence *obj* is illuminated as required.

```
with b11a
show "(objloc s)·obj ∩ (⋃ l | l ≤ lasnum • lasbeams s l) ≠ ∅"
  by (auto)
```

6.5 *SSR_E*

We must establish that *obj* is *in_sight*.

Firstly, from *TDS_D*, *obj* is observed and therefore, from *PAR_B*, nearby as above.

```
apply_end (rule contrapos_nn)
apply_end (assumption)
apply_end (inference)
from b10 TDS_D [ of "s"]
```

```
show "obj ∈ dom (objloc s)"
  apply (fspace)
  apply (simp add: observed_def PAR_B glambda_beta glambda_dom)
  apply (auto)
  done
```

Secondly, suppose that *obj′* is another nearby object.

```
fix obj'
assume "obj' ∈ dom (objloc s)" "obj' ≠ obj"
note b13 = this
```

From *TDS_E* we know that there is line-of-sight to *obj*.

```
from b7 [THEN TDS_E] b9 b6
have "line_of_sight obj s"
 by (auto simp add: cINTVLo_def)
note b14 = this
then have b15: "protect s ∩ beam parbw origin (targfix ((plots s)·obj)) = ∅"
 by (simp add: line_of_sight_def)
with b10' PARC_B [of l obj s] b11d b6''
have b16: "parbeams s l = beam parbw origin (targfix ((plots s)·obj))"
 by (simp add: PAR_A PARC_B TDS_C PARC_A OPI_B OP_C)
from lasbw_le_parbw
have b17: "beam lasbw origin (targfix ((plots s)·obj)) ⊆
 beam parbw origin (targfix ((plots s)·obj))"
 apply (auto simp add: beam_def)
 apply (rule exI)
 apply (inference)
 apply (intro exI)
 apply (inference)
 apply (rule refl)
 apply (auto intro!: exI)
 done
```

Thus we know that the laser beam illuminating *obj*, *beam lasbw origin* (*targfix* ((plots *s*)·*obj*)), does not illuminate the observed part of *obj′*, if any. As above, since the radar beam supporting this laser beam entirely encompasses it, any illumination of *obj′* would be on its observed part. Thus there is a beam to *obj* that does not illuminate *obj′*, as required.

```
show "(objloc s)·obj' ∩ (⋂ p⃗ | p⃗ ∈ (objloc s)·obj • beam lasbw origin p⃗) = ∅"
proof (cases "obj' ∈ observed s")
 assume c1: "obj' ∉ observed s"
 with b13 have "(objloc s)·obj' ∩ radiates s = ∅"
  by (simp add: observed_def PAR_B glambda_dom)
 then have c2:
  "(objloc s)·obj' ∩ beam parbw origin (targfix ((plots s)·obj)) = ∅"
  apply (rule contrapos_pp)
```

```
  apply (simp add: nempty_conv)
  apply (simp add: radiates_def)
  apply (inference)
  apply (intro exI conjI)
  apply (assumption)
  apply (rule b11d)
  apply (simp add: b16)
  done
 with b17 have c4:
  "(objloc s)·obj' ∩ beam lasbw origin (targfix ((plots s)·obj)) = ∅"
  by (auto)
 from targfix_targets [of "(plots s)·obj", THEN subsetD [OF b12a]]
 have "(⋂ p⃗ | p⃗ ∈ (objloc s)·obj • beam lasbw origin p⃗)
  ⊆ beam lasbw origin (targfix ((plots s)·obj))"
  apply (intro subsetI)
  apply (rule InterD)
  apply (assumption)
  apply (cases "targfix ((plots s)·obj)")
  apply (auto)
  done
 with c4 show "(objloc s)·obj' ∩ (⋂ p⃗ | p⃗ ∈ (objloc s)·obj • beam lasbw origin p⃗) = ∅"
  by (auto)
next
 assume c1: "obj' ∈ observed s"
 with b13 b14 have c2:
  "(plots s)·obj' ∩ beam lasbw origin (targfix ((plots s)·obj)) = ∅"
  by (simp add: line_of_sight_def)
 from c1 have c3: "(plots s)·obj' = (objloc s)·obj' ∩ radiates s"
  by (simp add: PAR_B observed_def glambda_beta glambda_dom)
 from b6'' b11d b15 b10' PARC_B [of l obj s]
 have c4: "beam parbw origin (targfix ((plots s)·obj)) = parbeams s l"
  by (simp add: PAR_A PARC_A OPI_B OP_C PARC_B TDS_C)
 with b11d have c5: "beam parbw origin (targfix ((plots s)·obj)) ⊆ radiates s"
  by (auto simp add: radiates_def)
 with c3 have "(objloc s)·obj' ∩ beam parbw origin (targfix ((plots s)·obj))
  ⊆ (plots s)·obj'"
  by (auto)
 with c2 have
  "(objloc s)·obj' ∩ beam parbw origin (targfix ((plots s)·obj)) ∩
  beam lasbw origin (targfix ((plots s)·obj))
  = ∅"
  by (auto)
 with b17 have c6:
  "(objloc s)·obj' ∩ beam lasbw origin (targfix ((plots s)·obj)) = ∅"
  by (auto)
 from targfix_targets [of "(plots s)·obj", THEN subsetD [OF b12a]]
```

```
  have "(⋂ p⃗ | p⃗ ∈ (objloc s)·obj • beam lasbw origin p⃗)
    ⊆ beam lasbw origin (targfix ((plots s)·obj))"
    apply (intro subsetI)
    apply (rule InterD)
    apply (assumption)
    apply (cases "targfix ((plots s)·obj)")
    apply (auto)
    done
  with c6 show "(objloc s)·obj' ∩ (⋂ p⃗ | p⃗ ∈ (objloc s)·obj • beam lasbw origin p⃗) = ∅"
    by (auto)
 qed
next
```

6.6 *SSR_F*

Suppose that the PARTI is not currently authorised to operate. From *PAR_A*, *PARC_A*, *OPI_B* and *OP_C* we know that all radar beams are off. From *LAS_A*, *LASC_A*, *OPI_B* and *OP_C* we know that all laser beams are off.

```
 fix t
 assume "¬(commanded t)"
 show
   "(⋃ r | r ≤ parnum • parbeams t r) = ∅"
   "(⋃ l | l ≤ lasnum • lasbeams t l) = ∅"
   by (auto ! simp add: PAR_A PARC_A OPI_B OP_C LAS_A LASC_A)
qed

end
```

7 Conclusion

DEF(AUST)5679/Issue 2 [2] includes a new requirement to model and verify the Safety Architecture against the System Safety Requirements. Confidence in the correctness of the Safety Architecture is critical to developing assurance of System safety. Additionally, analysis of the Safety Architecture has the potential to highlight safety concerns at an early stage of System Development, allowing redesign for safety with a minimum of expense and delay. Given the potential value of an clear understanding of the Safety Architecture, the Standard requires Formal support at a relatively low System Danger Level.

The Isabelle/Isar tool [23] is a well supported, mature theorem proving environment that is used worldwide; with significant levels of expertise and support available in Australia. We have used Isabelle's **locale** mechanism to provide natural support for a block diagram based approach to modelling

and verifying the SAFETY ARCHITECTURE. The resulting model is small and maintainable, while nevertheless capturing and demonstrating the effectiveness of the key safety features of the SYSTEM. When combined with judicious use of auxiliary block diagrams, such models can be made accessible to a fairly wide audience and provide an important mechanism for transferring safety assurance.

This paper has demonstrated that FORMAL analysis of the SAFETY ARCHITECTURE can be performed using lightweight FORMAL tools; with easily justifiable effort and expense.

References

1. Abrial, J. (1996) *The B-Book: Assigning Programs to Meanings*, Cambridge University Press, Cambridge.

2. Australian Government Department of Defence (to appear in 2008) *Def (Aust) 5679, Issue 2: Safety Engineering for Defence Systems.*

3. Defence Material Organisation (2009) *IGP-001: Guidance Notes for Project Offices.* Published as part of [20].

4. Defence Material Organisation (2009) *IGP-002: Methods for Safety Architecture Analysis.* Published as part of [20].

5. Defence Material Organisation (2009) *IGP-003: Methods for Design Assurance.* Published as part of [20].

6. Defence Material Organisation (2009) *IGP-004: Guidance on the Use of NDIs.* Published as part of [20].

7. Defence Materiel Organisation (2009) *PARTI-HAR: Hazard Analysis Report for PARTI System.* Published as part of [20].

8. Defence Materiel Organisation (2009) *PARTI-SAR: Safety Architecture Report for PARTI System.* Published as part of [20].

9. Defence Materiel Organisation (2009) *PARTI-DAR-ILL: Design Assurance Report (Laser Illuminator) for PARTI System.* Published as part of [20].

10. Defence Materiel Organisation (2009) *PARTI-DAR-LAS: Design Assurance Report (Laser Interlock) for PARTI System.* Published as part of [20].

11. Defence Materiel Organisation (2009) *PARTI-DAR-OP: Design Assurance Report (Operator) for PARTI System.* Published as part of [20].

12. Defence Materiel Organisation (2009) *PARTI-DAR-FIL: Design Assurance Report (PAR Filter) for PARTI System.* Published as part of [20].

13. Defence Materiel Organisation (2009) *PARTI-DAR-TDS: Design Assurance Report (TDS) for PARTI System*. Published as part of [20].

14. Defence Materiel Organisation (2009) *PARTI-SCS-1: Safety Case Summary (Hazard Analysis Phase) for PARTI System*. Published as part of [20].

15. Defence Materiel Organisation (2009) *PARTI-SCS-2: Safety Case Summary (Safety Architecture Phase) for PARTI System*. Published as part of [20].

16. Defence Materiel Organisation (2009) *PARTI-SCS-3: Safety Case Summary (Design Assurance Phase) for PARTI System*. Published as part of [20].

17. Defence Materiel Organisation (2009) *PARTI-SER-1: Safety Evaluation Report (Hazard Analysis Phase) for PARTI System*. Published as part of [20].

18. Defence Materiel Organisation (2009) *PARTI-SER-2: Safety Evaluation Report (Safety Architecture Phase) for PARTI System*. Published as part of [20].

19. Defence Materiel Organisation (2009) *PARTI-SER-3: Safety Evaluation Report (Design Assurance Phase) for PARTI System*. Published as part of [20].

20. Department of Defence (2009) *DEF(AUST)10679/Issue 1, Guidance Material For DEF(AUST)5679/Issue 2*.

21. Department of Defence (2009) *DEF(AUST)5679/Issue 2, Safety Engineering for Defence Systems*.

22. International Organization for Standardization (2002) *Information technology – Z formal specification notation – Syntax, type system and semantics*. URL – http://standards.iso.org/ittf/PubliclyAvailableStandards/c021573_ISO_IEC_13568_2002(E).zip.

23. Isabelle Project (2008) The Isabelle theorem proving environment. URL – http://www.cl.cam.ac.uk/Research/HVG/Isabelle/. Accessed June 2008.

24. Mahony, B., McCarthy, J., Williams, K. & Vu, L. (2008) Z support in the HiVe mathematical toolkit. To be published as DSTO Technical Report (72 pages).

25. Rodin Project (2008) Event-B webpages. URL – http://www.event-b.org/. Accessed June 2008.

26. University of Cambridge Computer Laboratory (n.d.) The Isabelle 2005 theorem proving environment, http://www.cl.cam.ac.uk/users/lcp/archive/Isabelle2005.tar.gz. URL – http://www.cl.cam.ac.uk/users/lcp/archive/Isabelle2005.tar.gz.

27. Wenzel, M. (2005) *The Isabelle/Isar Reference Manual*. URL – http://www.cl.cam.ac.uk/Research/HVG/Isabelle/dist/Isabelle2005/doc/isar-ref.pdf.

Page classification:

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA				1. CAVEAT/PRIVACY MARKING
2. TITLE The PARTI Architecture Assurance			3. SECURITY CLASSIFICATION Document (U) Title (U) Abstract (U)	
4. AUTHORS B Mahony A Cant			5. CORPORATE AUTHOR Defence Science and Technology Organisation PO Box 1500 Edinburgh, South Australia 5111, Australia	
6a. DSTO NUMBER DSTO–GD–0557	6b. AR NUMBER AR 014-350		6c. TYPE OF REPORT General Document	7. DOCUMENT DATE October, 2008
8. FILE NUMBER	9. TASK NUMBER DMO 07-007	10. SPONSOR DMO	11. No OF PAGES 40	12. No OF REFS 27
13. URL OF ELECTRONIC VERSION http://www.dsto.defence.gov.au/corporate/reports/DSTO–GD–0557.pdf			14. RELEASE AUTHORITY Chief, Command, Control, Communications and Intelligence Division	
15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT *Approved For Public Release* OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111				
16. DELIBERATE ANNOUNCEMENT No Limitations				
17. CITATION IN OTHER DOCUMENTS No Limitations				
18. DSTO RESEARCH LIBRARY THESAURUS Safety engineering Requirements management Standards				

19. ABSTRACT

Safety Critical Systems are those with the potential to cause death or injury as a result of accidents arising from unintended system behaviour. The arguments for safety, along with the body of supporting evidence, make up what is called the *Safety Case*. Requirements and guidance for Safety Cases are given in Def (Aust) 5679 Issue 2 [2]; in this standard the key stages of the Safety Case are: *Hazard Analysis*, *Safety Architecture* and *Design Assurance*. The process is driven by the identification of *System Safety Requirements*. The standard requires an argument be made that the Safety Architecture meets the System Safety Requirements. In the most serious cases, this argument is required to be made in a formal language and supported by formal reasoning tools. In this paper, we demonstrate the feasibility of such formal argument through the presentation of a formal verification argument for a simplified case study in Defence safety engineering.

Page classification: