Tokeneer ID Station
Project Plan

S.P1229.2.1
Issue: 1.2
Status: Definitive
19th August 2008

Originator

Janet Barnes (Project Manager)

Approver

David Stokes (Line Manager)

Copies to:

NSA	Praxis High Integrity Systems
	File

Contents

1 Introduction

In order to demonstrate that developing highly secure systems to the level of rigour required by the higher assurance levels of the Common Criteria is possible, the National Security Agency in the US (the NSA) has asked Praxis High Integrity Systems to undertake a research project to re-develop part of an existing secure system (the *Tokeneer System*) in accordance with Praxis' own high-integrity development process. The component of the Tokeneer System that is to be redeveloped is the core functionality of the Token ID Station (TIS). This re-development work will then be used to show the security community that it is possible to develop secure systems rigorously in a cost-effective manner.

The project is fixed price. However, as a research project, the scope of delivery is allowed to be flexible; if the initial scope of delivery proves to be too ambitious (or too conservative) then it will be possible to re-negotiate a revised scope with the client.

The prime output of the project is the evidence of the cost and effectiveness of the development process, not the resulting software product itself.

1.1 Scope

There are five systems of interest:

- the operational Tokeneer system
- the operational Tokeneer ID Station (a component of the operational Tokeneer system)
- the re-developed Token ID Station (TIS) (functionally equivalent to the operational Tokeneer ID Station)
- the re-developed TIS core functions (a subset of the software in the re-developed ID Station)
- the re-developed TIS support functions (all of the re-developed ID Station *except* the re-developed TIS core functions)

This Project Plan relates to the **re-developed TIS**.

2 Technical Plan

2.1 Requirements and Assumptions

The project will be based on an existing product, so the functional requirements are relatively clear, although the re-development will be performed within the context of a new Protection Profile [5]. We have only to scope the work by identifying which aspects of the existing system will be re-developed, and which we will exclude, as outlined in the proposal [4].

We are demonstrating the applicability of the Praxis high-integrity system development process, and therefore we must include REVEAL®, INFORMED, and SPARK. Plus, we must get it right.

2.2 Options

The options considered during the proposal phase and the initial requirements meeting with the NSA and SPRE Inc. were:

1. Re-development of the TIS software, interfacing into the working peripherals.
2. Re-development of only part of the TIS software, integrating our code into the existing NSA code, resulting in a complete, working system.
3. Re-development of all the TIS software, but interfacing into peripheral simulations, to be supplied by SPRE Inc. on behalf of NSA.
4. Re-development of only part of the TIS software, writing stubs of the parts not implemented, and interfacing into peripheral simulators, supplied by SPRE Inc. on behalf of NSA.

We selected option 4 because

1. it resulted in the smallest amount of work outside of the primary objectives of the project;
2. it avoided (1)'s real peripherals, which would have been in danger of taking up a lot of time in integration with no benefit for showing how Praxis' development process was superior;
3. (2) was rejected because the client software is not sufficiently well structured and modular to allow pieces to be removed and re-worked safely.

2.3 Planned Approach and Justification

The Praxis High Integrity Systems' high integrity systems development approach consists of:

1. Requirements analysis (the REVEAL® process)

2 Formal specification (using the formal language Z)

3 Design (the INFORMED process and refinement of the formal specification)

4 Implementation in SPARK Ada

5 Verification (using the SPARK Examiner toolset)

The NSA's main objective is to obtain evidence for the suitability of a formal approach in developing secure systems.

The chosen technical approach is thus focused on the development of those aspects of the TIS that benefit most from a formal treatment. We shall identify with the help of the NSA a set of TIS core functions during the initial requirements elicitation phase. We shall then specify, design and implement selected core functions according to our formal development approach. Sufficient functionality will be included in this core TIS to produce the basis for a useful and coherent system, but we shall exclude peripheral or less important functions.

The scoping decisions will be documented in the System Requirements Specification.

We shall implement the new code such that it interfaces correctly with peripheral simulators, which represents the TIS peripherals for the purposes of this development. These peripheral simulators will be developed by SPRE.

The benefit of this focused approach is that it is consistent with the NSA's overall technical objectives and budgetary constraints.

We shall perform the sequence of technical activities indicated in the following subsections.

The structure of the overall satisfaction argument from SPARK code to security properties is presented in Figure 1.

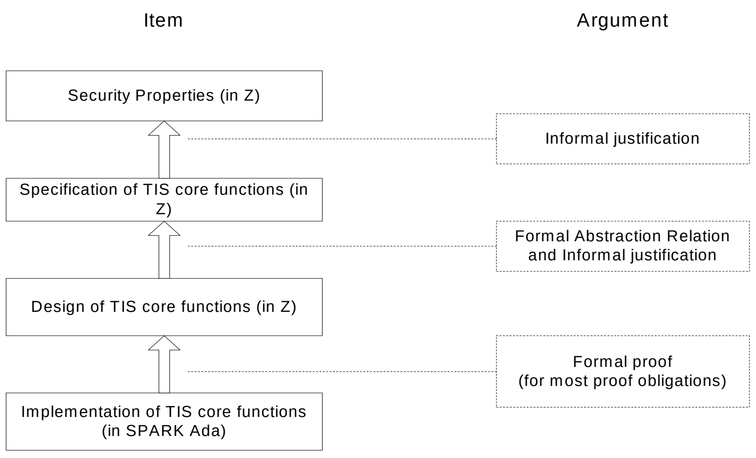

Item
Argument
Security Properties (in Z)
Informal justification
Specification of TIS core functions (in Z)
Formal Abstraction Relation and Informal justification
Design of TIS core functions (in Z)
Formal proof (for most proof obligations)
Implementation of TIS core functions (in SPARK Ada)

Figure 1: Overall Satisfaction Argument

2.3.1 Requirements Elicitation and Analysis

We shall conduct requirements elicitation and analysis on:

- the function and purpose of the TIS and Tokeneer (its parent system), including an investigation of their operational profiles and a consideration of existing documentation;
- the objectives of the project (including reliability objectives), definitions and classification of failures, and a definition of the performance metrics to be collected;
- the design requirement for encryption of sensitive data (as data encryption may not be necessary, given the physical and human environment for TIS);
- the intended purpose of the products from this project.

We shall use our REVEAL® method for this task.

The deliverable from this task is the **System Requirements Specification**. The scope of the System Requirements Specification will cover the TIS core functions identified and agreed during the requirements task, with tracing to system-level requirements from NSA Tokeneer documents.

We shall in addition review the TIS Kernel Protection Profile produced by SPRE [5], and based on this, we will write a **Security Target**, summarising the way in which the TIS as developed and deployed will meet the requirements of the Protection Profile. Specifically, this will explain why a number of security features are not implemented (e.g. internal data encryption, protection against tampering, etc.)

2.3.2 Formal Specification of the TIS Software

The TIS core functions (as identified by the System Requirements Specification) will be formally specified in Z. The boundary of the “TIS core functions Computer Software Configuration Item (CSCI)” is formed by the other TIS CSCIs with which the core functions interact (although these other CSCIs will be represented by peripheral simulators during the development). Figure 2 presents the initial view of the CSCIs and their relationships. The definitive statement will be documented in the System Requirements Specification.

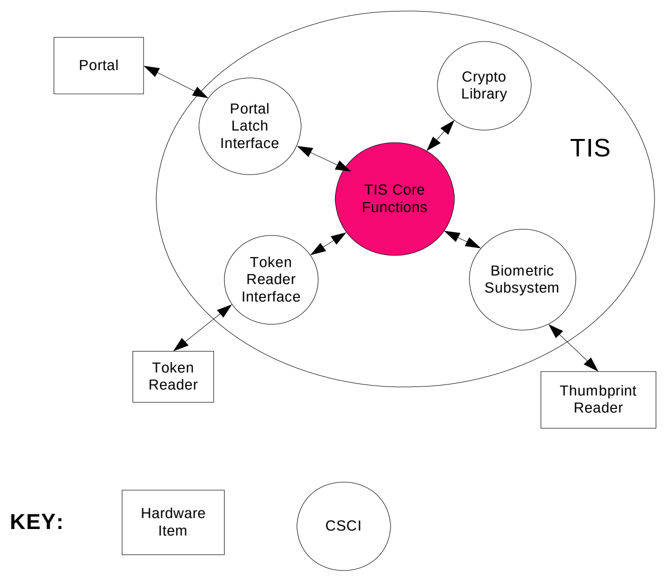


Figure 2: The Initial TIS CSCIs

The output from this activity will be a **formal specification** in Z of the TIS core functions.

2.3.3 Formal Specification and Proof of TIS Security Properties

The functional security requirements are specified in the TIS Kernel Protection Profile [5]. One of the following possibilities will apply.

- If a TIS security requirement can be addressed by a *single* TIS core function, then the "proof" of that requirement can be obtained merely by tracing from the relevant specified core function to the requirement and checking informally that the core function does indeed implement the requirement.

- If the satisfaction of a TIS security requirement can be obtained only via a sequence of TIS core functions, then we shall write a formal statement in Z of the security requirement (using the state components, inputs and outputs that are already present in the formal specification) and provide an informal justification that the relevant sequence of core functions satisfies the formal security requirement.

The output from this activity will be a document containing both the formal Z specification of the **security properties** required to be upheld by the TIS core functions and an informal **justification** that the functional specification satisfies these properties.

2.3.4 Formal Design of the TIS Core Functions

2.3.4.1 The Design

We shall construct a design document for selected TIS core functions which will consist of the following items.

- An informal description of the design philosophy, including an account of the interactions between design components (e.g. via UML collaboration diagrams, if appropriate).
- A formal Z design which is a data refinement of the functional CSCI specification and where the interface with the peripheral simulators is at a lower level of abstraction than that defined in the functional CSCI specification.

Note that the design specifically excludes the TIS graphical user interface. The current TIS graphical user interface is used only by the security officer to read and set some configuration parameters. We will use a file and command-line interface to mimic the security officer logon and data re-configuration.

The output from this activity will be a TIS core functions CSCI **design document**.

2.3.4.2 Summary of Exclusions from the Design Phase

The following items of work specified by the Statement of Work [2] are explicitly excluded from consideration in this Project:

1 Only selected TIS core functions will be designed.

2 The TIS Graphical User Interface will not be redesigned; we will mimic this using a file interface.

3 The encryption of sensitive data will not be included if our analysis concludes that such encryption is unnecessary given the constrained TIS physical and human environment (and of course if the NSA accepts the results of this analysis).

4 Operational configuration management, prevention of the introduction of malicious code and detection of physical tampering will not be designed and implemented, as these items are not relevant in accomplishing the main NSA objective for this project.

2.3.5 Design-to-Specification Correspondence Argument

We shall demonstrate the satisfaction of the TIS core functions specification by its formal design by means of the following activities.

1 We shall construct a formal Z specification of the abstraction relation that relates the state components of the specification to the state components of the design.

2 We shall produce an informal argument that the CSCI design correctly refines its specification, with reference to the formal abstraction relation.

The output from this activity will be a document that contains the formal specification of the **abstraction relation** and the informal **argument** that the TIS design refines the specification.

2.3.6 Implementation of the TIS Core Functions

We shall use SPARK Ada to implement selected TIS core functions[1] from the design document, interfacing with the NSA-provided simulations of the peripheral and the stubs we shall write for the other TIS CSCIs). The following checks will be implemented:

- data flow analysis;
- information flow analysis;
- proof of the absence of run-time errors.

The outputs from this activity will be the **source code** (including SPARK data and information flow analysis annotations) and **executable image** appropriate for the specified target platform.

2.3.7 Proof of the Implementation against the Formal Design

We shall perform the following activities:

1 Insert SPARK proof annotations into the TIS CSCI source code, generating these from the Z Design document automatically using a Perl script where possible, and by hand otherwise.

2 Run the SPARK Examiner and SPADE Simplifier in order to discharge the associated proof obligations.

3 Inspect any proof obligations not immediately proved by the Simplifier and provide informal justification for their satisfaction.

The outputs from this activity will be the **source code** with proof annotations and all **proof files**.

2.3.8 Software Assurance

We will conduct the following test activities:

- bottom-up integration testing of the support CSCIs with the SPRE-supplied peripheral simulators;
- functional (i.e. black-box) testing of the completed TIS; test cases will be derived from scenarios in the System Requirements Specification and the Formal Specification.
- support to SPRE while they perform their Reliability Demonstration Testing.

The outputs from this activity will be a **test report** and the **test results** (including the test scripts).

[1] except when interfacing to NT or other code

2.3.9 Reporting the Conclusions of the Project

An important objective of the project is to determine the applicability of the techniques used in this Project to the development of secure systems. We shall thus write a **Summary Report** containing the following items:

- an account of the results of the development, including an assessment of the level of effectiveness of each technique in accomplishing the desired objectives;
- an analysis of the metrics collected during the development.

We shall be seeking permission from the NSA to publish these findings in the open literature.

2.3.10 Deliverables

2.3.10.1 Administrative Deliverables

1 Programme Status Reports and Meeting Reports

2 Project Plan (incorporating Programme Management Plan, Software Quality Assurance Plan)

3 Configuration Management Plan

4 Software Test Plan and Procedures

5 Summary Report

2.3.10.2 Technical Deliverables

1 Software requirements specification for the TIS core functions

2 Formal functional specification in Z of selected TIS core functions

3 Formal specification in Z and informal justification of TIS security properties

4 Formal design in Z of selected TIS core functions

5 Formal abstraction relation and informal argument for correspondence of the TIS core functions design to its specification

6 Software product (i.e. source code and executable image)

7 Proof files for formal SPARK proof of correspondence of SPARK code to the formal design of the TIS core functions CSCI

8 Software test scripts, results and report

9 Installation instructions and release notice.

2.4 Assumptions and Dependencies

We are depending upon the NSA to supply us, in a timely manner, with technical information on the working of the current TIS.

We are depending upon SPRE to supply the peripheral simulations, and to work with us in integrating our support CSCIs with their peripheral simulations.

2.5 Risks

1 If technical information is late incomplete or contradictory, this will impact delivery. Contradictions, if present will become apparent during formal specification as will omissions. As these arise we will be reliant on technical advice from the NSA in order to complete the formal system specification. Our dependencies on the NSA arise early in the programme during requirements analysis and specification.

2 We are getting some products from SPRE (the Protection Profile and access to peripheral simulators, and possibly the specifications for the simulators, too). These deliverables may be late. The Protection Profile is required to complete the Security Target and Security Properties. The peripheral simulators are required to perform software assurance testing activities.

3 The planned mechanism for integration testing of our interfaces to the peripheral simulators is via remote access to a machine sited at SPRE Inc which hosts the peripheral simulators. Resolving integration problems over these distances may be slow, especially when the relative time zones are accounted for. There may be problems establishing an appropriate link, especially if it requires siting machines outside respective company firewalls.

4 The technical scope of the project, in terms of the breadth of functionality being re-developed may exceed the budgetary constraints imposed on the project. In this case Praxis will open discussions with the NSA technical authority to resolve the most appropriate manner by which the scope of the project can be reduced without compromising the goals of the project.

2.6 Opportunities

1 There may be an opportunity to expand the technical scope of the project, either by providing advice and, or demonstration of the additional activities that would be required to produce a trusted system to EAL6 or EAL7. Alternatively the scope of the project could be broadened to encompass additional functionality. This opportunity will be dependant on Praxis having surplus budget, if it becomes apparent that this situation will arise Praxis will open discussions with the NSA technical authority over the preferred increase in scope.

3 Quality Plan

3.1 Project Requirements

The technical requirements will be documented in the System Requirements Specification [3], being produced near the beginning of the project.

The NSA will nominate a customer technical authority. He and Praxis will agree the scope of the requirements, and any changes necessary during the course of the project.

3.2 Acceptance Procedures

As described in the proposal, this is a research project, and as such there is no formal acceptance of the software product.

The summary report and the products of the development lifecycle will be delivered to the NSA for evaluation and acceptance. In particular:

- one round of comments will be invited for documents and the documents will then be updated in accordance with agreed changes. Comments are normally expected within 2 weeks, after which time the document will be assumed accepted without comment.
- the new software will be accepted subject to a successful demonstration of
  - installation (with an identified version of the peripheral simulations used as part of the test harness).
  - correct behaviour during an agreed set of functional demonstration tests. The set of functional demonstration tests to be used will be agreed with NSA prior to acceptance.

No warranty will be supplied for the project deliverables.

3.3 Quality Control Strategy

Deliverable documents will be reviewed internally to the project team, issued to the client for one formal review cycle, updated and re-issued formally.

Code and annotations will be subject to formal code reviews according to Praxis code review standard S.Q1000.54.5.

3.4 Documentation and Configuration Management

Documentation (in terms of issued versions) will be managed through doctools, a Praxis High Integrity Systems proprietary document management system based on RCS. During development of documents they will be stored on the project file share, accessible to all project members.

Code will be managed under CVS.

3.5 Change Control Procedures

Once a client deliverable document has been issued "Definitive", any changes will be managed under change control according to Praxis Change Control Procedures S.Q1000.4.6.

3.6 Fault Management

During development failure detection and fault correction will be managed as part of the normal software development process. Within this process any failure found in a document or source code that has undergone formal review will be corrected through the raising of an incident report.

During the period of reliability demonstration testing by SPRE, failures will be raised by SPRE into three categories:

- blocking (immediate): without rapid resolution of the fault, SPRE's testing cannot usefully continue.
- blocking (can work around): there are elements of testing that cannot be completed without resolution of the fault, but SPRE can continue with other useful testing.
- non-blocking: the fault does not prevent testing of all the system functionality.

We will respond to any *blocking (immediate)* faults as fast as is reasonably possible, working directly with SPRE by phone and email to resolve the immediate issues.

We will respond formally to any *blocking (can work around)* faults, collecting together a number of fault repairs and issuing new versions weekly as necessary.

We shall not repair non-blocking faults.

The categorisations of the failures will be agreed by the fault review board, consisting of the NSA, SPRE and Praxis.

In order for Praxis to be able to respond to failures raised by SPRE the following information will be required:

- description of the failure.

- conditions required to reproduce the failure, including input values and sequence actions.
- expected behaviour and observed behaviour.

3.6.1 Fault Monitoring

Problems found with reviewed documentation or code during the development process and during reliability demonstration testing will be logged as incidents using Praxis incident reports. An incident report is presented in Appendix A.

Incident Reports are raised to document any unexpected feature of the system, when an incident is raised a description of the problem is provided and the point in the lifecycle that the incident is raised is identified. For this project the following lifecycle phases have been identified

- Requirements Analysis*
- System Specification*
- Specification of Security Properties*
- Proof of Security Properties (system specification satisfying security properties)
- Formal Design*
- INFORMED Design*
- Proof of Design (against specification)
- Code*
- Proof of Code
- Integration* (this encompasses all work involving the development of interfaces which are not part of the core TIS)
- System Test*
- Acceptance Test* (Reliability demonstration testing performed by SPRE Inc)

The problem is then evaluated and categorised into one of the following categories, note that only the first three categories apply to the TIS core functions that are being developed using the formal development process:

- Critical – this indicates a blocking (immediate) fault in the code or a fault that would result in unacceptable insecurity.

- Major – this indicates a blocking (can work around) fault in the code or a fault in the design or specification that would result in inconsistencies between deliverables.
- Minor – this indicates a non-blocking fault in the code or a cosmetic fault in the design or specification.
- Interfaces – this indicates a fault with the stubbed libraries or drivers. The stubbed libraries or drivers require modification.
- Test – this indicates a fault with the test suite. The test suite requires modification.
- No Fault – the incident is dismissed as requiring no change for resolution.

The source of the fault is also identified; this uses a subset of the lifecycle phases in which the fault may be located (marked by * in the earlier list) and it should identify the earliest lifecycle phase in which the fault is exhibited.

Faults will be allocated for resolution and updates reviewed prior to closure.

It is the responsibility of the project manager to monitor and report, within monthly status reports, on:

- The progress of incidents, through raising, evaluation, resolution and closure
- statistics on the severity of faults found and
- lifecycle stage faults found vs. lifecycle stage faults introduced.

3.7 Project Monitoring and Reporting

3.7.1 Metrics

An important objective of the client is to gain evidence of the effectiveness of the development approach we use. We will help meet this objective by recording metrics, as follows.

The following categories of skills will be used in the development process:

- Requirements analysis
- Z specification
- Security
- Design
- SPARK implementation
- Integration and system infrastructure
- Project management
- Quality control activities
- Configuration management
- Z proof

- SPARK proof
- Test.

Actual effort expended by each individual on the project categorised by the above skills will be recorded on a monthly basis.

Each activity carried out will also be identified as “Hard” or “Easy”, indicating whether the task needs an expert to carry it out, or whether it could be done by a novice. The skill level in each category will be identified for each individual (Expert, Practitioner or Novice). This will allow us to see when an expert was carrying out an easy task.

3.7.2 Status reports

Customer status reports will be issued monthly by email. They will report on:

- progress this month
- effort metrics this month, and to date
- fault statistics
- risks and their mitigations
- opportunities
- modifications to the plan
- financial: payment milestones reached or expected, and the progress of invoices.

3.8 Security

This project is unclassified.

3.9 Standards

We will use Praxis standards. There are no client standards mandated.

3.10 Project Lifecycle Controls

Deliverables generated during the lifecycle of the project will be subject to quality controls as described in the following table.

Table 1 : Controls for deliverable documentation

Deliverable		Quality Control			Deliverable
Item	**Standard**	**Method**	**Standard**	**Record**	**Approved by**
PLANNING					
Project Plan	S.Q1000.52.1 S.Q1000.52.4 S.Q1000.52.6	Formal Review	S.Q1000.4.10	Review Report	Line Manager (LM) Quality Manager (QM) Client
Configuration Management Plan	S.Q.1000.4.6	Formal Review	S.Q1000.4.10	Review Report	Project Manager (PM)
Software Test Plan	S.Q1000.55.1	Formal Review	S.Q1000.4.10	Review Report	Project Manager (PM)
DELIVERABLES					
Status Reports	S.Q1000.52.2	Inspection	S.Q1000.4.10	Signature	LM
Summary Report	S.Q1000.4.3	Formal Review	S.Q1000.4.10	Review Report	LM Client
TECHNICAL					
Software Requirements Specification	S.Q1000.53.1	Formal Review	S.Q1000.4.10	Review Report	PM or Technical Authority (TA) Client[2]
Security Target	S.Q1000.53.1	Formal Review	S.Q1000.4.10	Review Report	PM or TA
Formal Specification	S.Q1000.53.1 S.Q1000.59.11	Formal Review	S.Q1000.4.10	Review Report	PM or TA Client
Formal Security Properties	S.Q1000.53.1 S.Q1000.59.11	Formal Review	S.Q1000.4.10	Review Report	PM or TA Client
Formal Design	S.Q1000.59.11	Design Review	S.Q1000.54.2	Review Report	PM or TA Client
Formal Abstraction Relation	S.Q1000.59.11	Inspection	S.Q1000.4.10	Signature	PM or TA
Source Code		Code Reviews	S.Q1000.54.5	Review Report	PM or TA

[2] Where one of the Project Manager or the Technical Authority is the primary author the other will be the approver.

Deliverable		Quality Control			Deliverable
Item	**Standard**	**Method**	**Standard**	**Record**	**Approved by**
Code Proof Files	S.Q1000.4.3	Inspection	S.Q1000.4.10	Signature	PM or TA
Test Scripts	S.Q1000.55.2	Inspection	S.Q1000.4.10	Signature	PM or TA
Test Results	S.Q1000.55.4	Inspection	S.Q1000.4.10	Signature	PM or TA
Installation Instructions	S.Q1000.4.3	Inspection	S.Q1000.4.10	Signature	PM or TA

Technical deliverables will need to be reviewed against outputs of earlier lifecycle phases; the following describes the context in which each of the technical deliverables should be produced and checked.

Technical deliverables:

1 System requirements specification for the TIS core functions.

   This is the result of the first stage of the development (requirements analysis). It will act as the document against which the formal functional specification will be checked for completeness and correctness. It will be partially complete before the formal functional specification is started, but will not be finished until the formal functional specification has made significant progress.

2 Security Target.

   This will be derived from the Protection Profile, supplied by SPRE. It will be used in conjunction with the Protection Profile to identify the appropriate security properties to extract and formalise.

3 Formal functional specification in Z of selected TIS core functions

   This is the result of the specification stage. This will be checked against the System requirements specification.

4 Formal specification in Z and informal justification of TIS security properties.

   This will be derived from the Security target and the Protection Profile. It does not contribute to any other process stage.

5 Formal design in Z of selected TIS core functions.

   This will be developed in parallel to the INFORMED design stage. It will be derived from the Formal functional specification, and will use the Formal abstraction relation to justify its correctness.

6 Formal abstraction relation and informal argument for correspondence of the TIS core functions design to its specification.

   This will be developed in parallel, but near the end of the formal design in Z. It will demonstrate the correctness of the design against the formal functional specification.

7 Software product (i.e. source code and executable image).

This will be derived from the formal design in Z, using the INFORMED documentation produced during the INFORMED design stage. It will be quality checked by code review, by the use of the SPARK Examiner (where the code is written in SPARK Ada), and testing.

8 Proof files for formal SPARK proof of correspondence of SPARK code to the formal design of the TIS core functions CSCI.

This will be the output of the proof stage of development of the core functionality.

9 Software test scripts, results and report.

The test scripts will be derived from the system requirements specification and the formal functional specification, and the test results from running the software product.

10 Installation instructions and relase notice

Will be written based on the needs of the software product.

3.11 Client Feedback

At the end of the project, the line manager will ask for client feedback from the NSA.

It is Praxis High Integrity Systems policy to seek client feedback from all clients in order to gauge customer satisfaction and enable process improvement.

4 Resource Plan

4.1 Work Breakdown Structure

The work breakdown structure is given below. Each of the technical tasks is broken into two separate sub-tasks: Hard and Easy. This is to ease the collection of metrics (see section 3.7) although the planned effort for a technical task has not been subdivided across these categories. Those activities marked with an * result in the production of a deliverable.

WBS Number	Description
0	**Complete Project**
1000	**Manage Project**
1100	Plan Project*
1200	Reporting Project*
1300	Supporting infrastructure
1400	Close Project
2000	**Define Requirements**
2100	Study documents
2110	Hard
2120	Easy
2200	Elicit requirements
2210	Hard
2220	Easy
2300	Write SRS*
2310	Hard
2320	Easy
2400	Travel time
2500	Write security target*
2510	Hard
2520	Easy
2600	Review SRS and ST
3000	**Specify System**
3100	Specify core functions*
3110	Hard
3120	Easy
3200	Specify security properties*
3210	Hard
3220	Easy
3300	Prove security properties*
3310	Hard
3320	Easy

WBS Number	Description
3400	Review spec and properties
4000	**Design Core Functions**
4100	Formal design*
4110	Hard
4120	Easy
4200	INFORMED design
4210	Hard
4220	Easy
4300	Abstraction relation*
4310	Hard
4320	Easy
4400	Review design
4500	Prove design*
4510	Hard
4520	Easy
4600	Review proof
5000	**Code and Prove**
5100	Code*
5110	Hard
5120	Easy
5200	Proof Annotations
5210	Hard
5220	Easy
5300	Proof of code*
5310	Hard
5320	Easy
5400	Code review
5500	Proof review
6000	**System Test**
6100	Test plan and specs*
6110	Hard
6120	Easy
6200	Execute Functional testing
6210	Hard
6220	Easy
6300	Test report and results*
6310	Hard
6320	Easy
7000	**Interfaces and Integration**
7100	Interface specification*
7110	Hard
7120	Easy

WBS Number	Description
7200	Interface implementation*
7210	Hard
7220	Easy
7300	Integration test specs
7310	Hard
7320	Easy
7400	Integration testing
7410	Hard
7420	Easy
8000	**Acceptance**
8100	Write summary report*
8110	Hard
8120	Easy
8200	Review summary report
8300	Write installation guide*
8310	Hard
8320	Easy
8400	Support reliability testing
8410	Hard
8420	Easy

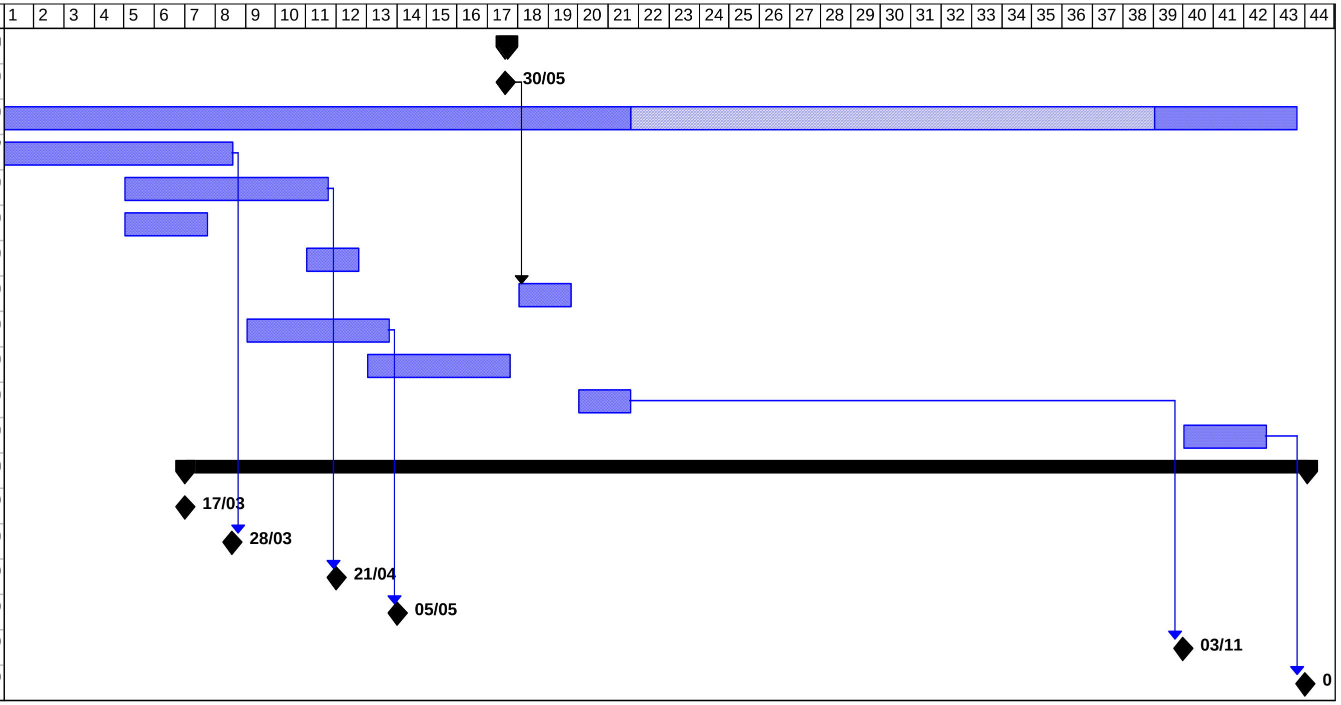
4.2 Timescales

The project starts 3rd of February 2003, and finishes no later than 1st of February 2004.

The time line given overleaf gives an aggressive schedule, putting the staff at the maximum utilisation, taking into account known dependencies and commitments. The intention is to run the project with three full time staff, see Resources (Section 4.3), as it is hoped that this will give more realistic team dynamics than using fewer staff or working part time over the available time. The only exception is the final activity of supporting SPRE with the reliability demonstration testing; this activity will be undertaken following completion of all other activities and within timescales convenient to both SPRE and Praxis.

The project schedule will be updated to reflect changes in the time line.

ID	Task Name	Start	Finish
1	**Dependencies**	**Fri 30/05/**	**Fri 30/05/**
2	SPRE complete test drivers	Fri 30/05/	Fri 30/05/
3	Manage Project	Mon 03/02/	Fri 28/11/
4	Define Requirements	Mon 03/02/	Thu 27/03/
5	Specify System	Mon 03/03/	Fri 18/04/
6	Specify Interfaces	Mon 03/03/	Fri 21/03/
7	Implement Interfaces	Mon 14/04/	Fri 25/04/
8	Integrate Interfaces	Mon 02/06/	Fri 13/06/
9	Design Core Functions	Mon 31/03/	Fri 02/05/
10	Code and Prove Core Functions	Mon 28/04/	Fri 30/05/
11	Test System	Mon 16/06/	Fri 27/06/
12	Support SPRE in Demonstration Testi	Mon 03/11/	Fri 21/11/
13	**Milestones**	**Mon 17/03/**	**Mon 01/12/**
14	Deliver Project Plan	Mon 17/03/	Mon 17/03/
15	Deliver SRS	Fri 28/03/	Fri 28/03/
16	Deliver Formal Specification	Mon 21/04/	Mon 21/04/
17	Deliver Formal Design	Mon 05/05/	Mon 05/05/
18	Deliver Software (Handover Meeti	Mon 03/11/	Mon 03/11/
19	Deliver Summary Report	Mon 01/12/	Mon 01/12/



4.3 Resources

4.3.1 Staffing

The following people will staff the project:

Project Manager:	Janet Barnes
Technical Authority:	David Cooper
Team member:	David Painter

These team members are all approximately full time for the main part of the duration of the project. Once we have handed over the software, there will be just a small level of involvement by one or two.

In order to be able to qualify effort spent by individual team members on the various activities the skills profile of each of the project team is provided in the table below:

Each team member has been allocated a skill level (Novice, Practitioner, Expert) based on the following criteria.

- Novice – Has attended relevant training but has no experience in the given (or a closely related) activity.
- Practitioner – Has attended relevant training and has sufficient experience in the given activity (or a closely related activity) to perform activity with minimal supervision.
- Expert – Has several years experience and can supervise both Practitioner and Novice in the activity.

	Janet Barnes	David Cooper	David Painter
Requirements Elicitation	Novice	Expert	Novice
Writing Z	Expert	Expert	Novice
Z Proof	Expert	Expert	Novice
Security	Practitioner	Expert	Novice
INFORMED Design	Expert	Novice	Novice
SPARK Coding	Expert	Novice	Practitioner
Writing SPARK Proof Annotations	Expert	Novice	Expert
SPARK Proof	Expert	Novice	Expert
System Testing	Practitioner	Novice	Novice

Table 2 Team Skill Levels

4.3.2 Other Resources

Staff will use their normal PCs for development work. A project share area will be set up on the IT infrastructure to hold documents, working software, etc.

Documents will be controlled under our document management system, doctools.

Software will be controlled under CVS.

A Incident Report

Project:	Incident Number/Reference:		
DESCRIPTION *(data and sequence of actions leading to fault, details of actual and expected response)*			
Found in test: *Supporting documentation attached* YES/NO	*Continued* YES/NO		
Found during: *(use actual project stages)*	Reqs /Sys spec/Security Spec/Proof of Spec/Formal Design/INFORMED Design/ Proof of Design/Code/Code Proof/Integration/Sys test/Acceptance		
Date:	Signature of Originator:		
EVALUATION *(include list of items affected, details of work required, other similar faults, tests to be re-run)*			
	Continued YES/NO		
Classification:	Critical / Major / Minor / Interfaces / Test / No Fault		
Introduced during: *(use actual project stages)*	Reqs/Sys spec/Security Spec/Formal Design/INFORMED Design/ Code/Integration/Sys test/Acceptance		
Date:	Signature of Evaluator:		
RESPONSE *(detail how incident is to be resolved, identify cause of problem, related faults and change requests)*			
	Continued YES/NO		
Date:	Signature of Project Manager:		
IMPLEMENTATION *(if applicable)*			
Assigned to:	Signature of Project Manager:		
Item modified	Date/Version	Signature of Checker	Signature of Integrator *Continued* YES/NO

Document Control and References

Changes history

Issue 0.1 (7th March 2003): Initial issue for review by Line Manager and Quality Manager.

Issue 1.0 (17th March 2003): Provisional issue following Praxis internal review S.P1229.7.2.

Issue 1.1 (2nd April 2003): Definitive issue following comments from (SPRE Inc) and (NSA).

Issue 1.2 (19th August 2008): Updated for public release.

Changes forecast

None.

Document references

1 *Token ID Station (TIS) Protection Profile*, Version 1.0, 20 August 2002

2 *NSA Statement of Work*, Draft 1, 27/9/02

3 *System Requirements Specification*, S.P1229.41.1

4 *Proposal, Formal Development of TIS Software, S.S5595.30.1*

5 *TIS Kernel Protection Profile,* Version 1.0, DocID D0205-01v10, 5 February 2003.