ANA Software Safety Assurance System

The journey so far…

Recognition

- We recognise:
  - Our systems are becoming more complex
  - Our systems are software dependent
  - There is a regulatory requirement for a SSAS

CERTISA™

Objectives

- We have two very simple objectives:
  - Ensure that the current levels of safety are at least maintained
  - To satisfy our NSA that we meet the regulatory requirement

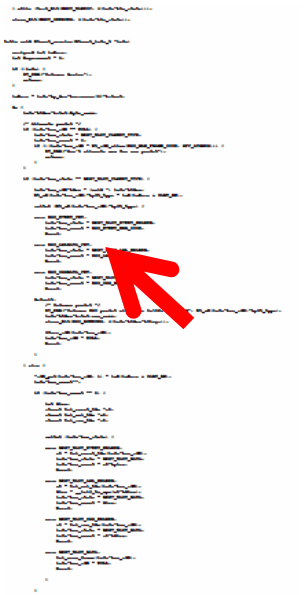
Scope of the Problem

- Here are a few thousand lines of code (about 6 kloc)
- Its in a part of the Linux kernel
- Less than 0.05% of the operating system
- Can you spot the bug?

CERTISA™

Obvious bug in kernel driver

- Come on, its obvious…



- You must be able to see it now?

CERTISA™

A simple mistake

```
if (info->rx_state == RECV_WAIT_PACKET_TYPE) {

    info->rx_skb->dev = (void *) info->hdev;
    bt_cb(info->rx_skb)->pkt_type = inb(iobase + UART_RX);

    switch (bt_cb(info->rx_skb)->pkt_type) {

    case HCI_EVENT_PKT:
        info->rx_state = RECV_WAIT_EVENT_HEADER;
        info->rx_count = HCI_ENT_HDR_SIZE;
        break;

    case HCI_ACLDATA_PKT:
        info->rx_state = RECV_WAIT_ACL_HEADER;
        info->rx_count = HCI_ENT_HDR_SIZE;
        break;

    case HCI_SCODATA_PKT:
        info->rx_state = RECV_WAIT_SCO_HEADER;
        info->rx_count = HCI_SCO_HDR_SIZE;
        break;

    default:
        /* Unknown packet */
        BT_ERR("Unknown HCI packet with type 0x%02x received", bt_cb(info->rx_skb)->pkt_type);
        info->hdev->stat.err_rx++;
        clear_bit(HCI_RUNNING, &(info->hdev->flags));

        kfree_skb(info->rx_skb);
        info->rx_skb = NULL;
        break;

    }
```

- Cut and paste leads to duplication

CERTISA™

Correct code

```
if (info->rx_state == RECV_WAIT_PACKET_TYPE) {

    info->rx_skb->dev = (void *) info->hdev;
    bt_cb(info->rx_skb)->pkt_type = inb(iobase + UART_RX);

    switch (bt_cb(info->rx_skb)->pkt_type) {

    case HCI_EVENT_PKT:
        info->rx_state = RECV_WAIT_EVENT_HEADER;
        info->rx_count = HCI_ENT_HDR_SIZE;
        break;

    case HCI_ACLDATA_PKT:
        info->rx_state = RECV_WAIT_ACL_HEADER;
        info->rx_count = HCI_ACL_HDR_SIZE;
        break;

    case HCI_SCODATA_PKT:
        info->rx_state = RECV_WAIT_SCO_HEADER;
        info->rx_count = HCI_SCO_HDR_SIZE;
        break;

    default:
        /* Unknown packet */
        BT_ERR("Unknown HCI packet with type 0x%02x received", bt_cb(info->rx_skb)->pkt_type);
        info->hdev->stat.err_rx++;
        clear_bit(HCI_RUNNING, &(info->hdev->flags));

        kfree_skb(info->rx_skb);
        info->rx_skb = NULL;
        break;
```

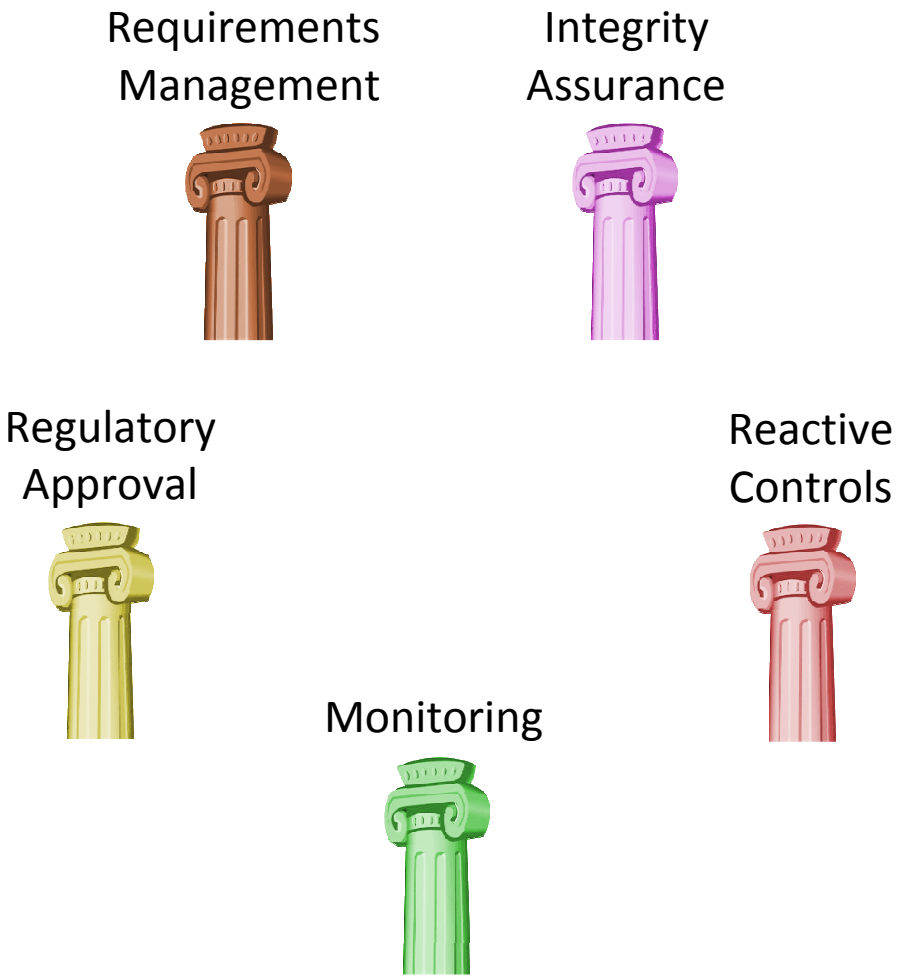
- Count should be the ACL header size, not the event size

CERTISA™

Scope

- So we cannot expect to find every defect
- We simply don’t have the resources or expertise
- We need to be able to select suppliers we can trust
- We need to be prepared when things go wrong

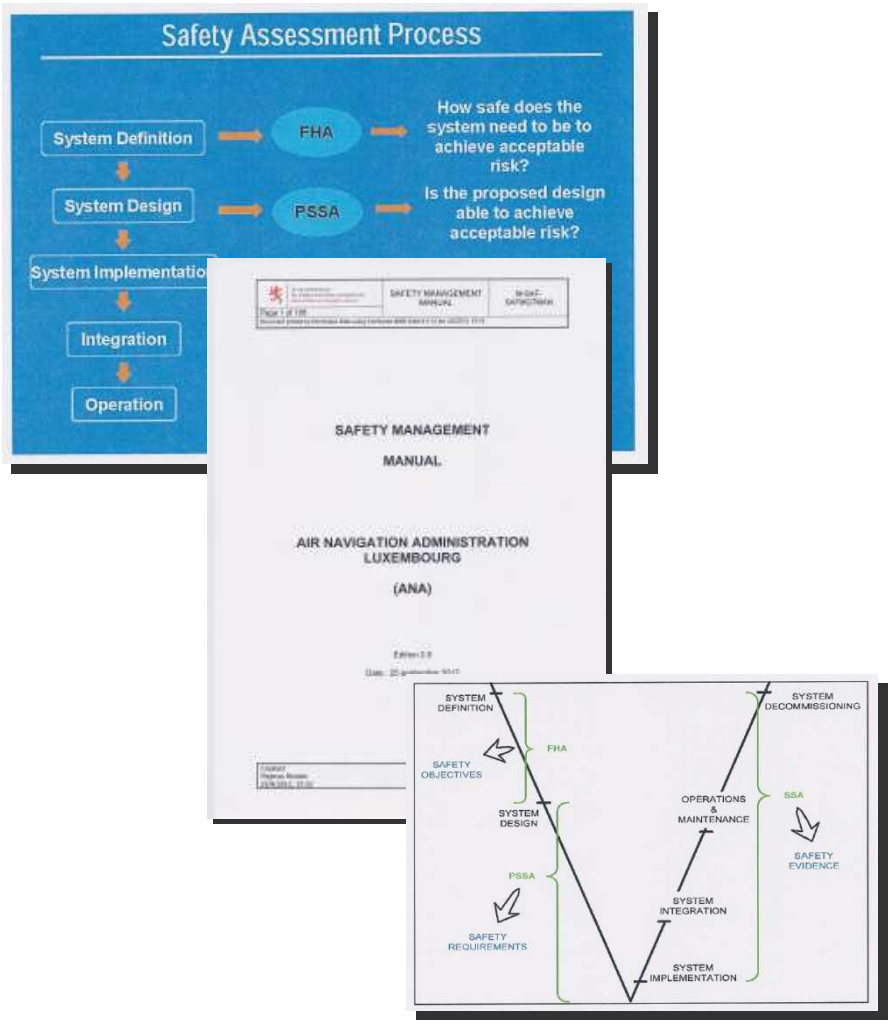
Our Strategy



- Divide the problem into chunks
- Establish “pillars” to support our objectives

CERTISA™

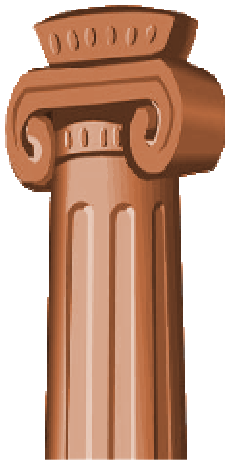
Use Existing Processes



- We already have project and safety management processes
- Documented in the SMS
- Make sure they properly cover software

CERTISA™

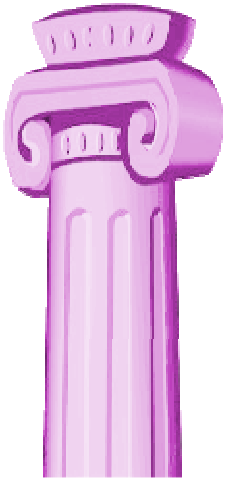
Requirements Management



- Review operational requirements capture process
- Review safety requirements capture process
- Review requirements management tools (DOORS?, Trac?, Access? Excel?)
- Requirement compatibility process (non-interference)

CERTISA™

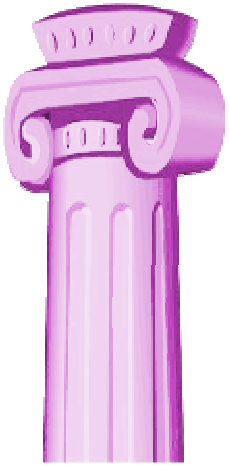
Integrity Assurance



- Probably the biggest challenge
- We are still developing the best approach
- Initially likely to include a mixture of:
  - Checklists / supplier questionnaires
  - Supplier audits
  - Evaluation tables (score cards)

CERTISA™

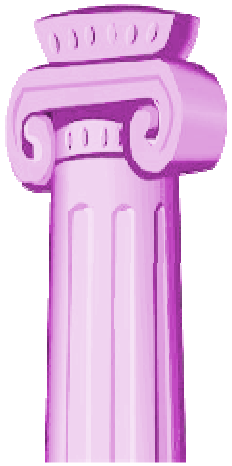
Integrity Assurance contd.



- Series of workshops
- Key aspects to be expected of suppliers
  - A product safety assessment has been performed
  - An established software quality system is in place and results are measured
  - Configuration management is in place

CERTISA™

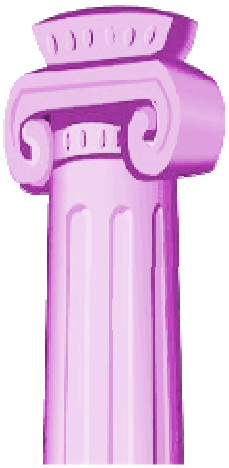
Verification and Validation



- Plan V & V at the system boundary
- Test defences (pro-active controls)
  - Identified during hazard assessment
  - I.e. auxiliary supply operation, fail-over
- Inspect correctness of documentation
- Validation of assumptions
- Validation of ‘known configuration’

CERTISA™

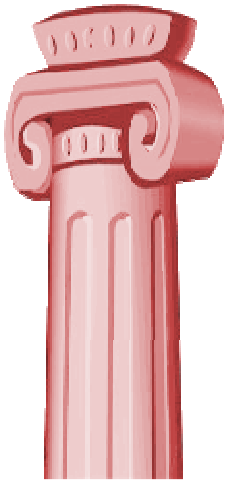
Training to Audit



- Risk based approach
- How to *generally* recognise good and bad software suppliers
  - Requirements management
  - Coding standards
  - Code reviews
  - Unit tests
  - Traceability
  - Configuration management
- Site visit to a good ‘reference’ supplier

CERTISA™

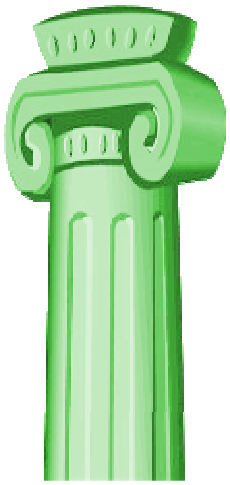
Reactive Controls



- Not yet able to properly assure our software
- Reactive controls can ensure safety
- Emphasis on failure conditions
- ATC defined procedures
- Training in unusual circumstances
- Planned workload reduction – flow control?

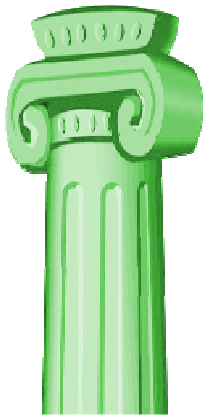
CERTISA™

Measure Defects



- Not yet developed, but some thoughts:
- Existing occurrence scheme (RAT)
  - but overload with new software systems?
- How to classify and measure ‘bugs’
  - Most are probably trivial
  - Many minor defects may indicate poor quality
  - What about ‘incorrect’ functionality?

Measure Effectiveness



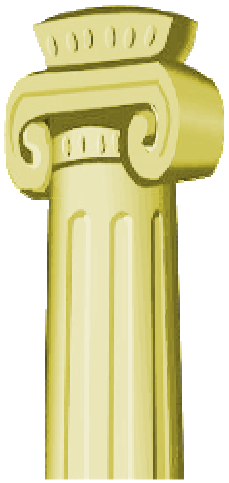
- Two trial projects
- Process to capture:
  - Defects
  - ‘Incorrect’ functionality
  - Inconsistencies
  - Operator satisfaction(?)
- As much detail as possible initially
- Establish a baseline
- Monitor improvements over time

Phased Introduction

- Phase 1: Action Plan
- Phase 2: Procedures, Templates and Training
- Phase 3: Two trial projects ALCMS and ASMGCS
- Phase 4: Performance review, critical systems review
- Phase 5: Risk assessment for legacy equipment

CERTISA™

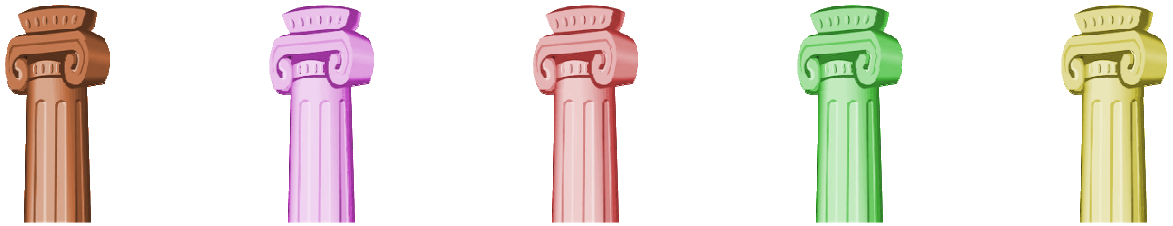
Regulator



- Agree action plan
- Approve SSAS procedures
- Approve two trial projects
- Opportunity to review and adjust SSAS
- On-going dialogue and approvals

In Summary

- Our approach is to break SSAS into smaller ‘chunks’



- Ensure we know our [safety] requirements
- Learn how to assess suppliers and where to target V&V efforts
- Know that software fails, learn how to predict software failure modes and plan to react
- Keep measuring performance
- Involve the regulator

CERTISA™