Event-B Development of a Smart Ballot Box

Dana Dghaym

HD-Sec Workshop, 16th September 2021

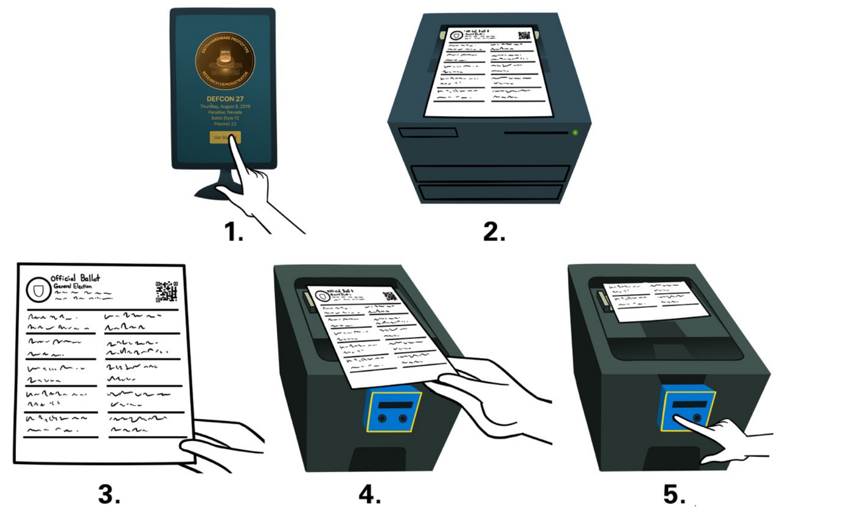
Outline

- **Part 1 :** Event-B development of a smart ballot box
  - Case study: Smart Ballot Box
  - Aims & Motivation
  - Event-B System Model of a Smart Ballot Box (SBB)
- **Part 2 :** Event-B to SPARK-Ada
  - Introduction to SPARK
  - High-Level Transformation Patterns : SBB Example
  - Transformation Issues
- Conclusions and Future Work

UNIVERSITY OF Southampton

Case Study: Smart Ballot Box

- Key Function of SBB:
  - Ensures only valid ballot papers are cast in ballot boxes for later tabulation.
- Why the SBB?
  - Security Properties: Confidentiality, integrity and availability.
  - Relatively Small Case Study
  - Can be used as Demonstrator



Galois and Free & Fair. The BESSPIN Voting System (2019).

UNIVERSITY OF Southampton

Motivation

- Application of refinement-based formal modelling in building a Correct-by-Construction secure system.
- Refinement of the security properties of a system.
- Overall Aim of case study:
  - Show how the Smart Ballot Box can be correctly implemented on capability hardware according to the system-level security specification.
- Develop a tool-supported approach to translate an Event-B model to verified code.

SBB System Model: Refinement Strategy

0. Abstract level: Model an ideal voting system.
1. Model possible attackers' behavior by distinguishing between different types of ballot papers.
2. Introduce time and invalidate ballots with expired timestamps.
    - Time can be the subject of more attacks.
3. Data refine the voter information by encrypting ballots.
4. Ensure the legitimacy of ballots through the Message Authentication Code (MAC).

SBB Model: Abstract Level

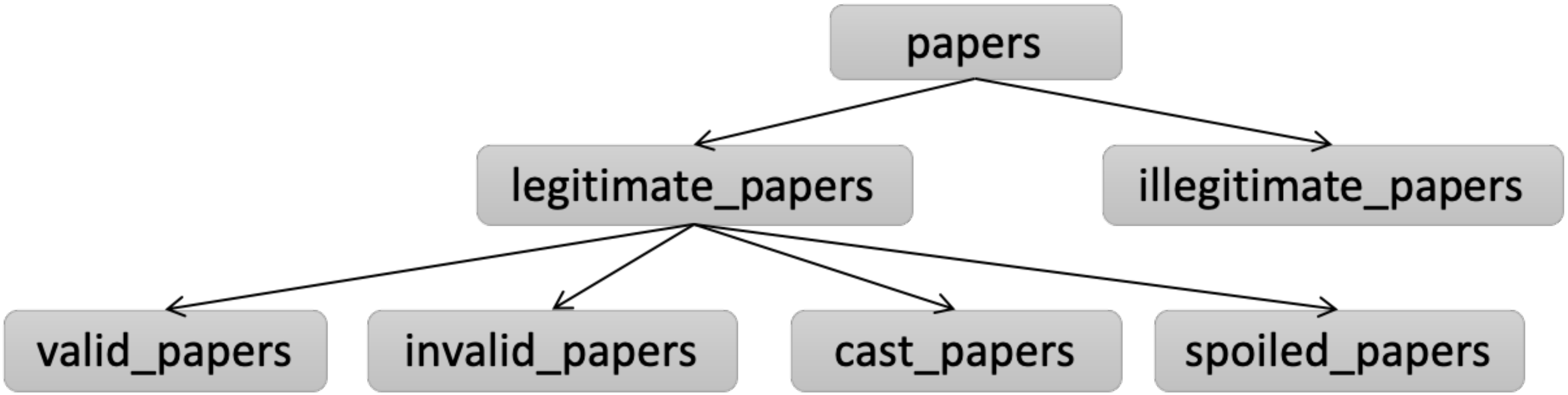
- Events: create_ballot, cast ballot, invalidate_ballot
- Model an ideal voting system
  - Each voter can have at most one legitimate ballot

$$ballots \in VOTER \nrightarrow VOTE$$

  - The cast ballots must be legitimate

$$cast \subseteq ballots$$

First Refinement: Ballot types



- Possible attacks
  - Attacker create ballot/duplicate valid ballot ..
- Model the main security properties of SBB
  1. Accept all valid ballots
  2. Reject invalid ballots

First Refinement: Availability Property

- Availability property: Ensure valid ballots are not blocked from being cast.

- Availability property is captured by the guard of the relevant events.
  - Specify *cast_paper* as rigid event with *paper* as rigid parameter .

    *Rigid: the guard cannot be strengthened

```
event [cast_paper] refines cast_ballot
any [paper] where
  @valid-paper: paper ∈ valid_papers
then
// actions for casting a ballot
end
```

Second Refinement: Time & Availability

UNIVERSITY OF Southampton

- This theorem ensures that a ballot paper is considered valid only if:
  - Paper time has **not expired**
  - Voter has **not cast** their vote before
  - The paper is **not spoiled**
  - Issued by a **legitimate** source

```
theorem @accept-valid-paper:
∀ paper · paper ∈ valid_papers ⇒
// paper not already expired
 paper_time(paper) ≥ current_time −
expiry_duration
// copy not already cast
∧ paper_voter(paper) ∉
paper_voter[cast_papers]
// copy not already spoiled
∧ (∀ sp · sp ∈ spoiled_papers ⇒
paper_voter(paper) ≠ paper_voter(sp)
∨ paper_vote(paper) ≠ paper_vote(sp)
∨ paper_time(paper) ≠ paper_time(sp)
)
// paper is not illegitimate
∧ paper ∉ illegitimate_papers
```

Third Refinement: Ballot Encryption

- Introduce encryption to prevent SBB from accessing the voter's information.
  - Apply data refinement to replace *paper_voter* and *paper_vote* with encrypted ballot

```
theorem @accept-valid-paper:
∀ paper · paper ∈ valid_papers ⇒
 paper_time(paper) ≥ current_time −
expiry_duration
// copy not already cast
∧ paper_encrypted_ballot(paper) ∉
paper_encrypted_ballot[cast_papers]
// copy not already spoiled
∧ (∀sp · sp ∈ spoiled_papers ⇒
paper_encrypted_ballot(paper) ≠
paper_encrypted_ballot(sp) ∨
paper_time(paper) ≠ paper_time(sp)
)
∧ paper ∉ illegitimate_papers
```

Fourth Refinement: Ballot Authentication

- Introduce MAC to check the legitimacy of the source issuing the ballot.
  - We assume the attacker does not know the secret key; therefore, it is crucial to ensure the secrecy of this key.

```
@mac-legitimate_papers: ∀paper · paper ∈ legitimate_papers ⇒
paper_mac(paper) = MACAlgorithm(
paper_time(paper) ↦ paper_encrypted_ballot(paper) ↦ MACKey
)
```

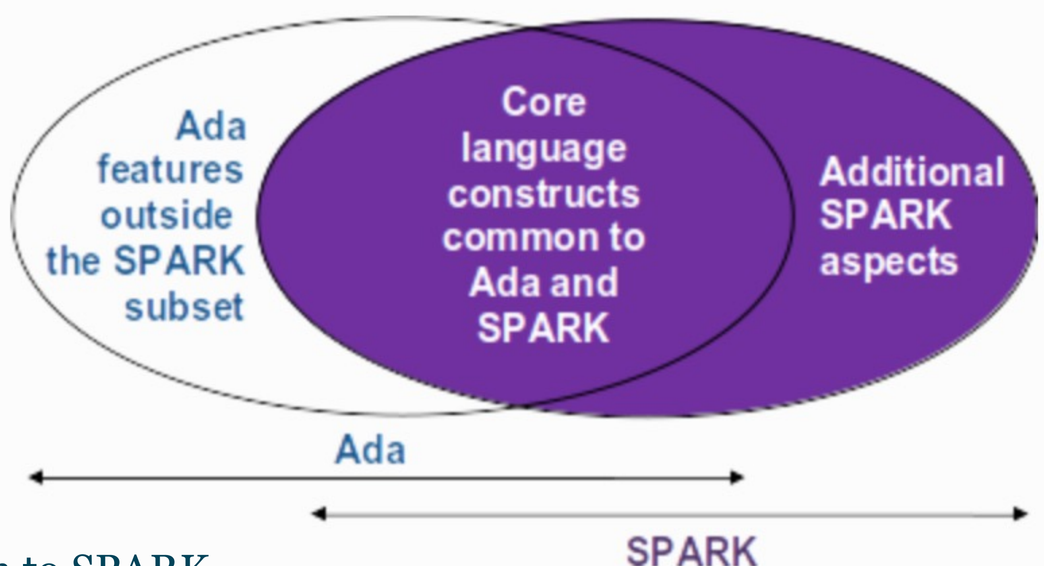
- All proofs are automatically proved with the help of SMT-Solver plugin

UNIVERSITY OF Southampton

Part 2: Translating Event-B to SPARK-Ada

Introduction

- What is **SPARK**?
  - A programming language based on a subset of the Ada language,
  - Targeted at functional specification and static verification.
  - A set of development and verification tools for that language.



*AdaCore: Introduction to SPARK

From Event-B to SPARK

UNIVERSITY OF Southampton

```
machine m
sees C
variables
…
invariants
…
events
 event
INITIALISATION
  then
    @act1: …
  end
  event evt
   any
    parameters
   where
    @grd1: ..
   then
    @act1: …
  end
end
```

```
context C
sets
…
constants
…
records
 record T
   A : Integer
   B : Integer
 …
axioms
….
end
```

```
package P
with SPARK_Mode => On
is
Some_Global : G;
type T is record
   …
   A : Integer;
   B : Integer;
end record;

function F(X :)
return ..
  …
;
procedure Proc (X : in T)
with
 Global => (Input => (..),
               … ),
 Pre => .. ,
 Post => .. ;
end P;
```

```
package body P
with SPARK_Mode => On
is
procedure Proc (X : in T) is
begin
    … ;
end Proc;
end P;
```

Refinement towards Implementation

- Sets → arrays
  - Data Refine a set to Total function from Integer range to the set type
  - Can introduce a counter variable to track the size of array
- Event-B records are more general than SPARK (Event-B records supports optional and relational fields)
  - Use only total functions
  - Define a special null record element to reflect Event-B optional possibility and it can be used for initialisations

High level Event-B Transformation

- **Event-B Models Translation**
  - Each context → specification package using all extended context packages
  - Last Refined Machine → specification and body packages using all context and extended contexts packages
- **Machine Elements Translation**
  - Variables → Global variables, initialised according to the INITIALISATION event actions
  - Event / INITIALISATION → Procedures
  - Event Guards → Pre-conditions
  - Event Actions → Post-conditions
  - Event Parameters → Procedure Parameters (Output, input, in out depends on guards and actions)

Smart Ballot Example

```
event cast_paper
refines cast_paper
any
paper
where
@grd1: paper ∈ BARCODE
@grd2: cast_count ∈ 0 ‥ max_votes −1
                    ....
then
@act1: cast_arr(cast_count):= paper
@act2: cast_count ≔ cast_count + 1
end
```

```
procedure cast(paper : in  barcode) with
 Global => (Proof_In => ( spoiled_arr,
curr_time, spoil_count),
 In_Out => (cast_arr, cast_count)),
 Pre => cast_count in 0 .. Max_Votes-1),
   and then not already_cast(paper)
                 ...
Post => already_cast(paper)
    and then cast_count = cast_count' old + 1);
```

```
procedure cast(paper : in  barcode) is
 begin
    cast_arr(cast_count) := paper;
    cast_count := cast_count + 1;
end cast;
```

Transformation Issues

- What do we **prove** at SPARK level?
  - **Not** necessarily all system invariants need to be re-proved in SPARK (already proved in Event-B)
  - Need to prove Ada is a correct implementation of the Event-B model
    - Some invariants might be required (e.g., well definedness)

Conclusions

- The SBB Event-B model
  - Modelled different security properties: Availability, confidentiality & integrity
  - Showed how we applied a refinement-based approach to model security properties
- Manual Transformation of Event-B Models to SPARK
- Identification of Translation Patterns
  - Applied to SBB & Tokeneer

Future Work

- What additional assertions are needed at SPARK level (invariants)
- Automatic Code Generation
  - Define a SPARK EMF Metamodel using XSD schema generated by GNAT
  - Event-B EMF to SPARK EMF Transformation

UNIVERSITY OF Southampton

Thank you

Questions?