Sviluppo di Software Sicuro - S^3
TIS - Architettura

Corso di Laurea Magistrale in
Sicurezza Informatica: Infrastrutture e Applicazioni
Università di Pisa – Polo di La Spezia
C. Montangero
Anno accademico 2009/10

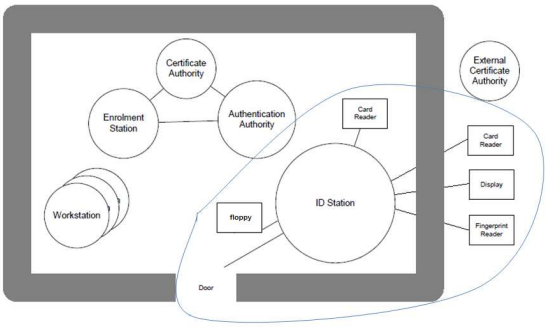
Sommario

- Struttura generale
- Preliminari: gerarchie di package
- Preliminari: variabili volatili
- Viste d'uso

S^3 2009/10 – TIS - Architettura

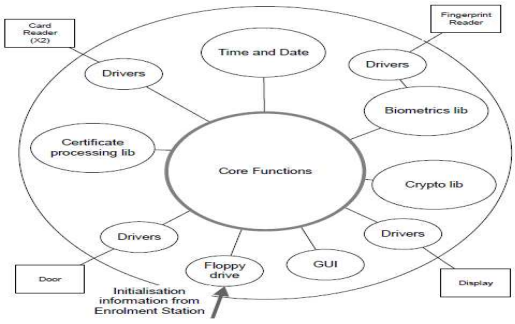
STRUTTURA GENERALE

Il contesto del sistema



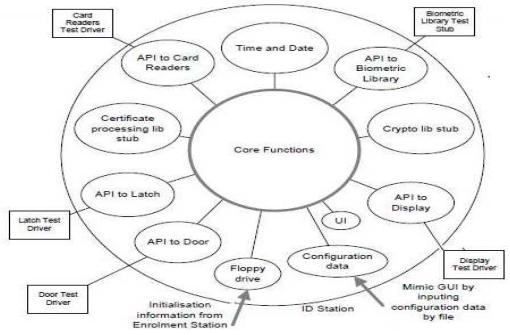
S3: SPARK - C.Montangero - Copyright 2010 4

Struttura generale originale



S3: SPARK - C.Montangero - Copyright 2010 5

Struttura generale TIS



S3: SPARK - C.Montangero - Copyright 2010 6

TisMain

- Sostanzialmente esegue un ciclo, in cui
  - aggiorna lo stato interno in funzione del mondo
  - elabora di conseguenza
  - struttura tipica dei sistemi di controllo
    - permette di assicurare i tempi di esecuzione
- User Entry e funzioni amministrative
  - concettualmente concorrenti
  - serializzazione per vincoli nei requisiti
    - due sistemi SPARK

Scheletro del ciclo del Main

```
Poll.Activity(SystemFault => SystemFault);
if not SystemFault then
   -- …
   Processing;
end if;
ShutdownCompleted := Enclave.HasShutdown;
exit when ShutdownCompleted;  -- SPARK
if SystemFault then ...
   exit;
end if;
```


Poll.Activity

```
Clock.Poll;
Door.Poll(SystemFault => SystemFault);
UserToken.Poll;
AdminToken.Poll;
UserEntry.DisplayPollUpdate;
Keyboard.Poll;
```

- Il primo oggetto del mondo interrogato é
  - clock che definisce
    - --# own CurrentTime -- inizio del ciclo
    - in funzione del clock di sistema

Clock.ads

```
package Clock
--# own    CurrentTime; -- stato interno
--#     in Now;         -- stato esterno, volatile
is
type TimeT is private;
      --internal representation of time.
-- …
procedure Poll; -- Reads the system clock,
                -- and updates CurrentTime.
   --# global in     Now;
   --#           out CurrentTime;
   --# derives CurrentTime from Now;
function TheCurrentTime return TimeT;
   --# global CurrentTime;    -- ora ultimo poll
function GetNow return TimeT;
   --# global Now;            -- ora corrente
```


Clock

- Questo package definisce anche
  - le operazioni sul tipo TimeT
  - le operazioni sul tipo Duration
    - intervalli di tempo
  - le operazioni su CurrentTime
    - aggiornamento
    - lettura

Problemi con Clock

- Minimizzare il codice che non può essere analizzato
  - perchè non rispetta i vincoli SPARK
  - e.g., accede al package Ada.Calendar
  - soluzione: *gerarchia* di package
- Definire l'interfaccia col mondo esterno
  - soluzione: variabili *volatili* (lette/assegnate dall'esterno)
  - il loro valore può cambiare senza interventi del programma
    - cfr memory mapped

S3 2009/10 – TIS - Architettura

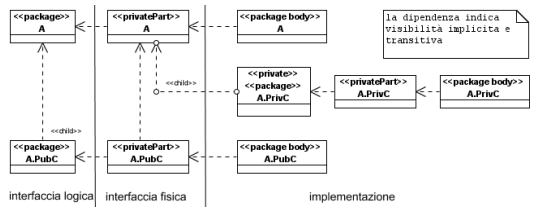
GERARCHIE DI PACKAGE

Strutturare un package

- Un package può essere strutturato ad albero
  - per facilitarne l'estensione
    - senza dover ricompilare tutti i clienti "soddisfatti"
    - le nuove operazioni introdotte in package *figli* (child) pubblici dell'originale
  - per rendere più manutenibile la realizzazione
    - esternalizzando parti del body
    - come figli **private** della specifica
  - individuando gruppi coesi di operazioni
- La struttura si riflette nei nomi:
  - A.B denota B come figlio (child) di A
  - impatto sulla visibilità, implicita ed esplicita

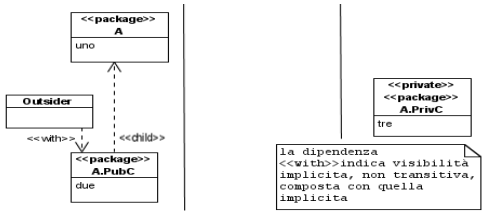
Livelli di interfaccia

- A : la radice
  - A.PubC : figlio pubblico
    - la parte pubblica vede A, la parte privata anche quella privata
  - A.PrivC : figlio privato
    - la parte pubblica vede anche la parte privata di A



Visibilità dall'esterno

- Outsider può fare with di A.PubC, e vede
  - "uno", come A.uno
  - "due", come A.PubC.due
- Non può vedere "tre" (no with A.PrivC)
- Ovviamente può anche solo fare with di A (e non curarsi di "due")

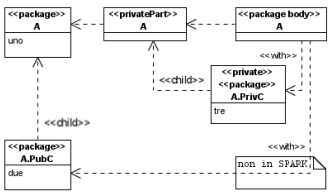


Esempio

- Complex
  - rappresentazione cartesiana
- Complex.Polar
  - rappresentazione polare
    - con operazioni su questa rappresentazione
    - con funzioni di conversione tra le due forme
- Astratto, ma non troppo

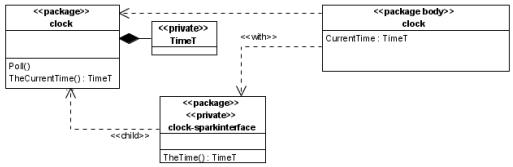
Visibilità dall'interno

- Cosa può vedere il body?
  - "uno", come uno
  - "tre", come A.PrivC.tre " (con with A.PrivC esplicito)
- Non può vedere "due" (no with A.PubC)
  - anche se in Ada sì, ma non è ovvio perché...



Esempio: Clock

- Clock definisce il tipo (provato) TimeT
- Clock.SparkInterface definisce TheTime
  - per catturare il clock di sistema
- Il body di clock usa Clock.SparkInterface.TheTime
  - per realizzare Poll e TheCurrentTime
- Sia il body sia il figlio vedono TimeT



Visibilità tra fratelli

- Esplicita tra nodi dello stesso sottoalbero
  - pubblici e privati in spazi distinti
- Si possono creare strutture complesse
- Ma: KISS!
  - keep it simple, stupid!

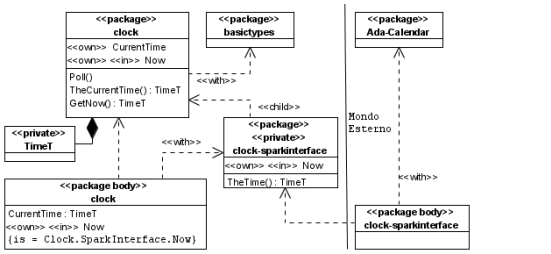
S^3 2009/10 – TIS - Architettura

VARIABILI VOLATILI

Variabili (oggetti) di confine

- Isolano il sistema SPARK dall'ambiente
- Definiscono lo scambio di informazione
  - attraverso lo stato del package di confine
  - definito da --# global
  - l'annotazione specifica anche il verso: **in**, **out**
- Stato volatile:
  - le **in** cambiano per effetto del mondo esterno
  - le **out** vengono utilizzate dal mondo esterno
  - ergo: assegnamenti, usi sempre *efficaci*
    - si posson mettere due usi o due assegnamenti di fila:
    - il mondo esterno può aver cambiato/usato il valore nel frattempo

Clock: struttura della realizzazione



- La variabile esterna Now è incapsulata nel package *figlio*
  - TheTime viene usata in Poll e GetNow
- Si minimizza la dipendenza dal modo esterno
  - I package del mondo esterno *non* vengono esaminati
    - tolti dal metafile o prefissi da --# hide

Clock.adb

```
with Clock.SparkInterface,
     BasicTypes;
package body Clock
--# own Now is in Clock.SparkInterface.Now;
-- raffinamento: notare in
is --...
 CurrentTime : TimeT; --...
 procedure Poll
 --# global in     SparkInterface.Now; --no Clock
 --#           out CurrentTime;
 --# derives CurrentTime from SparkInterface.Now;
 is
 begin
   CurrentTime := SparkInterface.TheTime;
 end Poll;
```

Visibilità nelle asserzioni

```
with Clock.SparkInterface, BasicTypes;
package body Clock
--# own Now is in Clock.SparkInterface.Now;
is --....
 procedure Poll
 --# global in     SparkInterface.Now; --no Clock
 --#           out CurrentTime;
 --# derives CurrentTime from SparkInterface.Now;
 is
 begin
   CurrentTime := SparkInterface.TheTime; -- no Clock
```

- inherit implicita…
- prefisso Clock solo nella own

Esempio di own out: Alarm

```
package Alarm
--# own out Output;
is
 procedure UpdateDevice;
 --# global    out Output;
 --#        in     Door.State;
 --#        in     AuditLog.State;
 --# derives Output from Door.State,
 --#                     AuditLog.State;
end Alarm;

with Alarm.SparkInterface, ...
package body Alarm
--# own Output is out Alarm.SparkInterface.Output;
 procedure UpdateDevice
 --# global    out SparkInterface.Output;
```


SparkInterface

```
private package Alarm.SparkInterface
--# own out Output;
is
   procedure Activate;  --    Activates the alarm
   --# global Output;
   --# derives Output from ;

   procedure Deactivate; --    Silences the alarm
   --# global Output;
   --# derives Output from ;

end Alarm.SparkInterface;
```

L'azione sull'allarme

```
procedure UpdateDevice
   --# global    out Interface.Output;
   --#         in     Door.State;
   --#         in     AuditLog.State;
   --# derives Interface.Output from Door.State,
   --#                               AuditLog.State;
is
begin
      if Door.TheDoorAlarm = AlarmTypes.Alarming or
         AuditLog.TheAuditAlarm = AlarmTypes.Alarming
      then
         Interface.Activate;
      else
         Interface.Deactivate;
      end if;
end UpdateDevice;
```

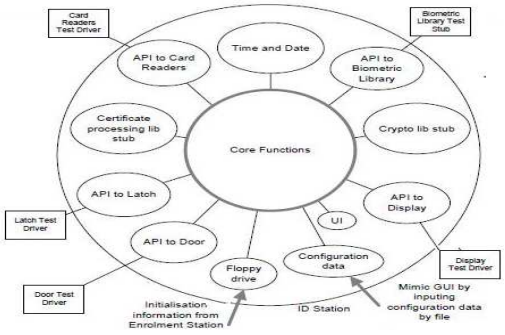
S3: TIS- C.Montangero - Copyright 2010 28

S³ 2009/10 – TIS - Architettura

VISTA D'USO

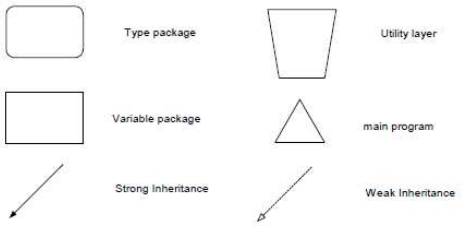
S3: SPARK - C.Montangero - Copyright 2010 29

Struttura generale TIS



S3: SPARK - C.Montangero - Copyright 2010 30

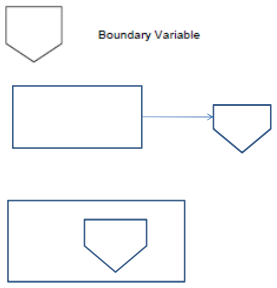
Simboli di INFORMED



Strong Inheritance = accesso in lettura/scrittura a oggetti
Weak inheritance = accesso a ADT

Simboli per il confine

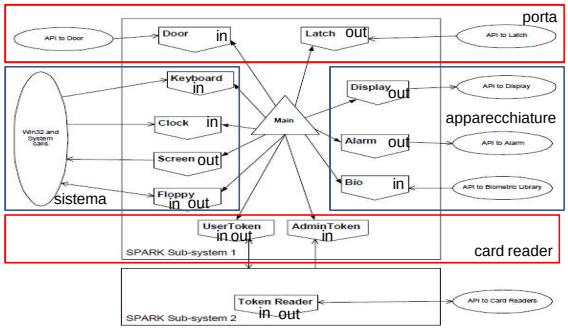
- Oggetto di confine
  - *boundary variable*
- Spesso con façade
  - per minimizzare le dipendenze dal mondo esterno
- Spesso come *child*
  - per minimizzare le dipendenze dalla rappresentazione



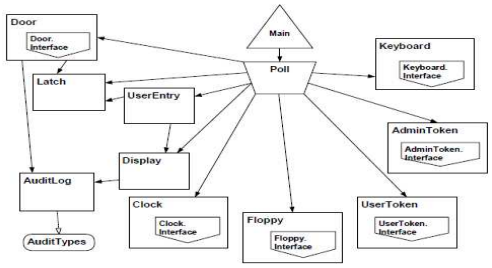
I confini dei sistemi

- Due sistemi SPARK
  - uno per il card reader
  - uno per le altre componenti esterne
  - ragioni:
    - card reader non modellato formalmente
    - così le asserzioni nel codice riflettono meglio il modello formale
    - card reader sufficientemente complesso da meritare verifica

Viste d'uso (globale, di confine)

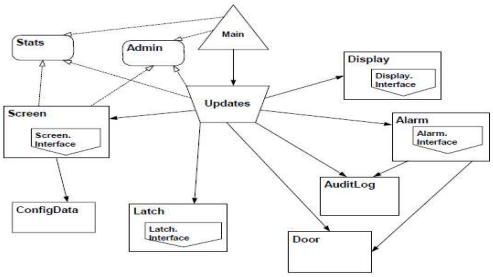


Viste d'uso: Polling

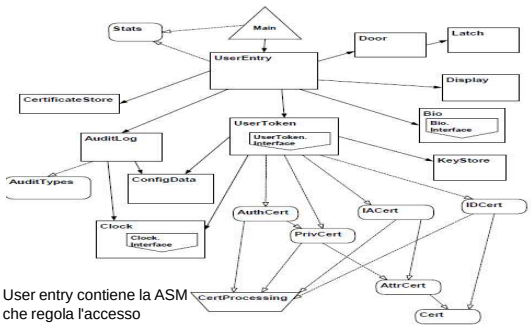


- Alcuni dati non dipendono dal mondo esterno

Viste d'uso: Updating

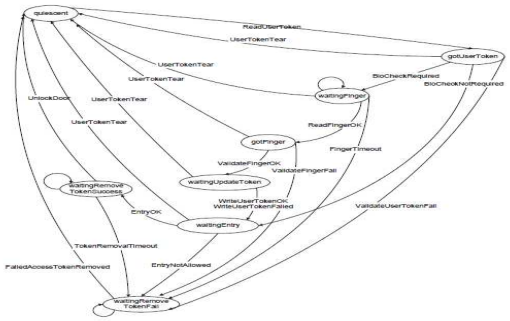


Viste d'uso: User Entry

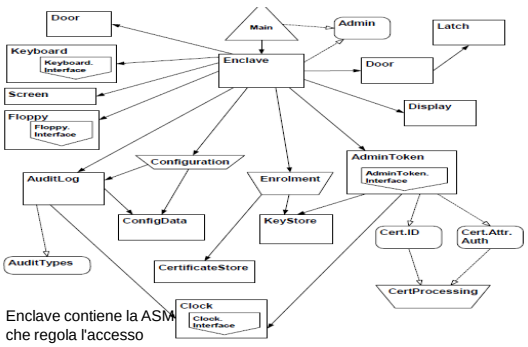


User entry contiene la ASM che regola l'accesso

La ASM di user entry

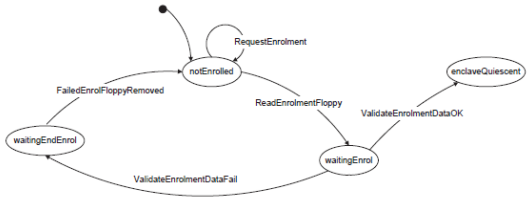


Viste d'uso: funzioni amministrative

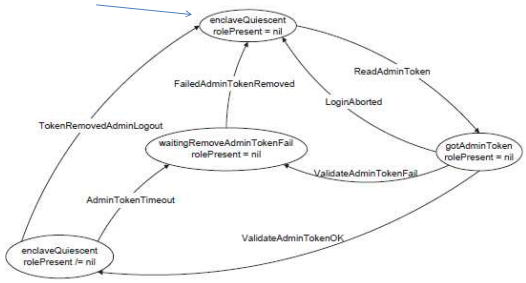


Enclave contiene la ASM che regola l'accesso

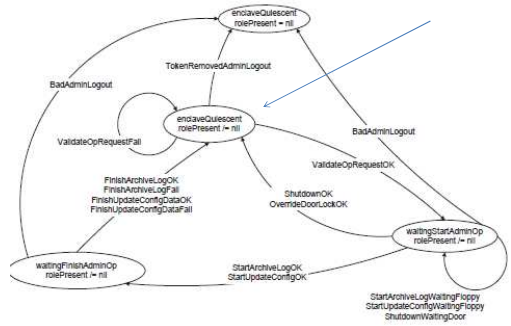
Enclave: ASM d'immatricolazione



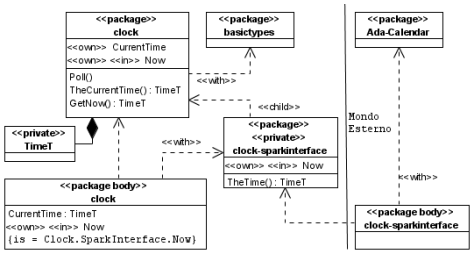
Enclave: ASM di logon/off



Enclave: ASM operazione d'admin

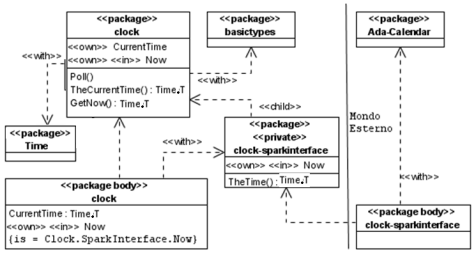


Esercizio: ADT Time per Clock



- Estrarre la definizione del tipo TimeT da Clock

Esercizio: ADT Time per Clock (2)



- Time definisce il tipo T (identico a TimeT di prima)
- Conviene portare TheTime nel body di Clock?
  - prefissa da --# hide
  - struttura più semplice, dipendenza dal mondo meno chiara