



dependable evolvable pervasive software engineering group

Modellazione Formale con UML e Logica Temporale



Pierluigi San Pietro, Matteo Rossi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
{pierluigi.sanpietro, matteo.rossi}@polimi.it



I *metodi* formali (*FM*)

- Uso di notazioni (*linguaggi*) formali (matematicamente definite) per definire, progettare, verificare sistemi (informatici e non)
 - Maggior rigore/precisione con tutti i benefici annessi (da quelli contrattuali a quelli realizzativi)
 - Maggiore automazione (supporto di strumenti)
 - Maggiore affidabilità complessiva
 - ...
- Si parla di *metodi* formali, in quanto i linguaggi sono usati -seguendo opportuni metodi- per specificare, progettare verificare

Esempio: Verificare/validare specifiche

- Se specifica non formalizzata, l'unica convalida possibile è un'analisi manuale.
- Se invece specifica (almeno in parte) formalizzata:
 - Prototipazione (simulation, prototyping, “testing” specs)
 - Verifica “automatica” di proprietà (safety,...), ad esempio con model checking
- In analogia con l'ingegneria tradizionale
 - modello fisico (di una diga, ...)
 - modello matematico e relative elaborazioni

Il metodo TRIO

- Linguaggio di specifica sviluppato fine anni 80
 - [Ghezzi, Mandrioli Morzenti, 90]
- esteso con costrutti orientati agli oggetti (TRIO+)
 - [San Pietro, Morzenti, 93]
- Tool automatici di supporto (es. Model checking)
- Applicato a molti esempi industriali, spesso non solo sw
 - Contatori enel, reti e stazioni di distribuzione e di produzione energia elettrica (ENEL, CESI)
 - Sistemi di segnalamento ferroviario (MM, RFI)
 - Sistemi di controllo del traffico (ITALTEL)
 - ...
- Più recentemente TRIO+ esteso verso la progettazione, in congiunzione con UML (ArchiTRIO)

TRIO

- TRIO è una logica temporale del prim'ordine, con una nozione metrica di tempo
 - le formule TRIO impongono dei vincoli sul comportamento dei fenomeni di un sistema
 - ogni formula TRIO è espressa rispetto all'istante temporale corrente, lasciato implicito
 - l'operatore *Dist* permette di esprimere proprietà in istanti diversi da quello corrente
 - esempio: $Dist(A, -2)$ dice che A valeva 2 istanti prima di quello corrente
- Esempio di formula TRIO:

$$\text{all } \tau \left(\text{temp} = \tau \right. \\ \left. \begin{array}{l} \rightarrow \\ \text{all } \epsilon \left(-10 < \epsilon < 10 \rightarrow Dist(\tau - 2 < \text{temp} < \tau + 2, \epsilon) \right) \right) \end{array} \right);$$

TRIO (2)

- TRIO ha vari operatori temporali adatti a descrivere in modo naturale proprietà temporali (inclusi requisiti quantitativi)
 - gli operatori derivati sono definiti a partire da *Dist*
- Esempio di formula TRIO con operatori derivati:

```
A $\exists$ w( temp =  $\tau$ 
  ->
  Lasts( $\tau$ -2 < temp <  $\tau$ +2, 10));
```

ArchiTRIO

- Idee chiave alla base di ArchiTRIO
 - la notazione grafica è standard UML 2.0
 - class diagram, composite structure diagram
 - si usa UML per descrivere quali sono gli elementi in gioco (i componenti), e come questi si compongono staticamente
 - la dinamica degli elementi è descritta in modo formale mediante formule di logica temporale
 - la logica usata è una estensione di ordine superiore della logica temporale TRIO
 - Higher Order TRIO (HOT)
 - non solo la dinamica può essere descritta in TRIO/HOT, ma anche relazioni complesse tra gli elementi

L'approccio ArchiTRIO

- Si inizia da una descrizione degli elementi del sistema mediante diagrammi UML
 - class/composite structure diagram
 - notazione usata a livello industriale
- Le varie classi (non tutte, solo quelle “critiche”) vengono arricchite con formule espresse in logica temporale
 - approccio “lightweight”: si formalizza solo dove utile/necessario
 - più elevato il livello di criticità dell’applicazione, più esteso l’uso della notazione formale nel modello
- E’ possibile fare analisi formale sul modello così creato (o su parti di esso)

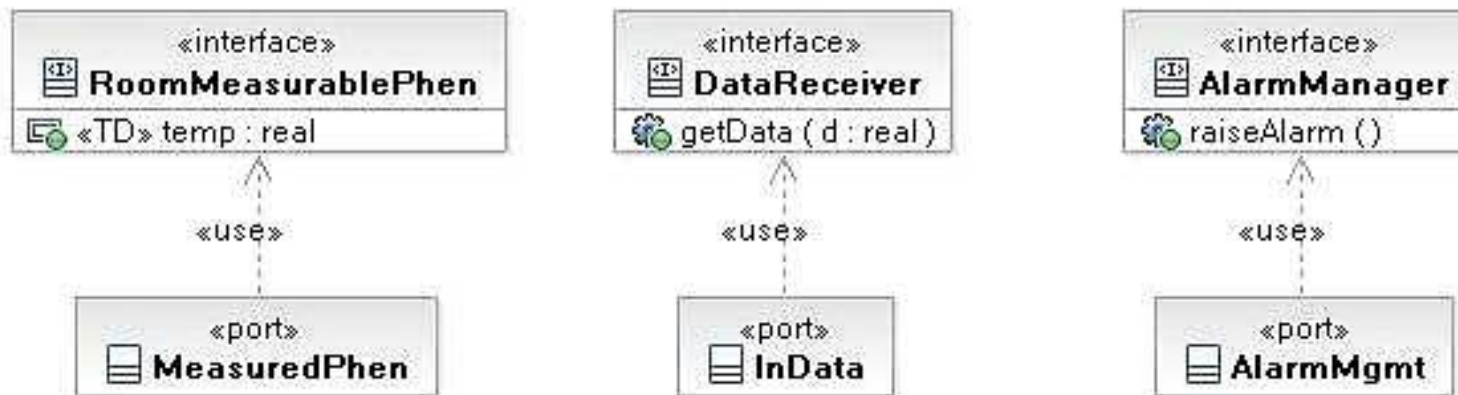
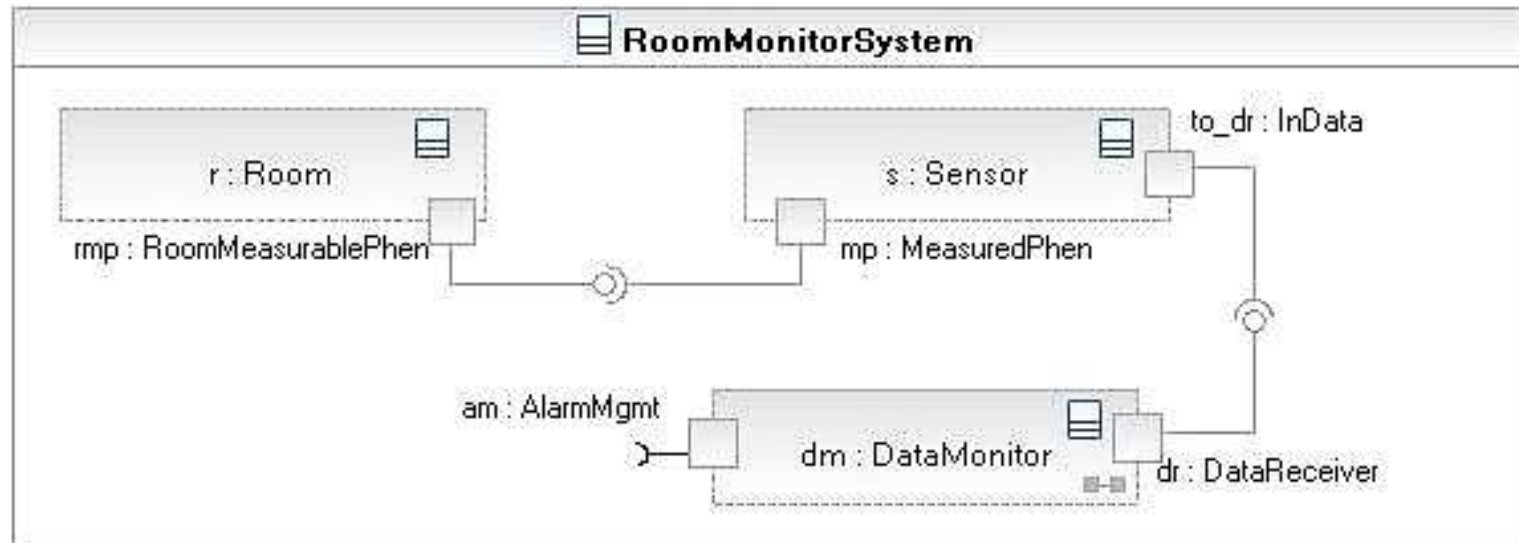
Esempio di classe UML/ArchiTRIO

- Una classe che rappresenta una stanza di cui si vuole misurare la temperatura:

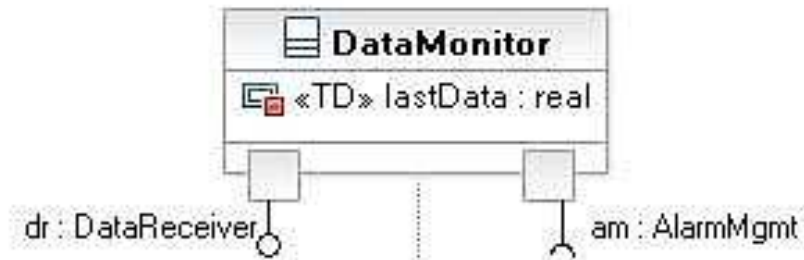


- *temp* modella la reale temperatura della stanza, che è un fenomeno (fisico) visibile (e misurabile) della stanza
 - *temp* è nell'interfaccia della classe
 - lo stereotipo **«TD»** indica che è una quantità che cambia nel tempo

Esempio (cont.)



Esempio (cont.)



```

alarm_raising
{
  Lasted(lastData > 30, 5)
  ->
  ex ra_op: AlarmMgmt.raiseAlarm (WithinF(am.ra_op.executing, 2))
}
  
```

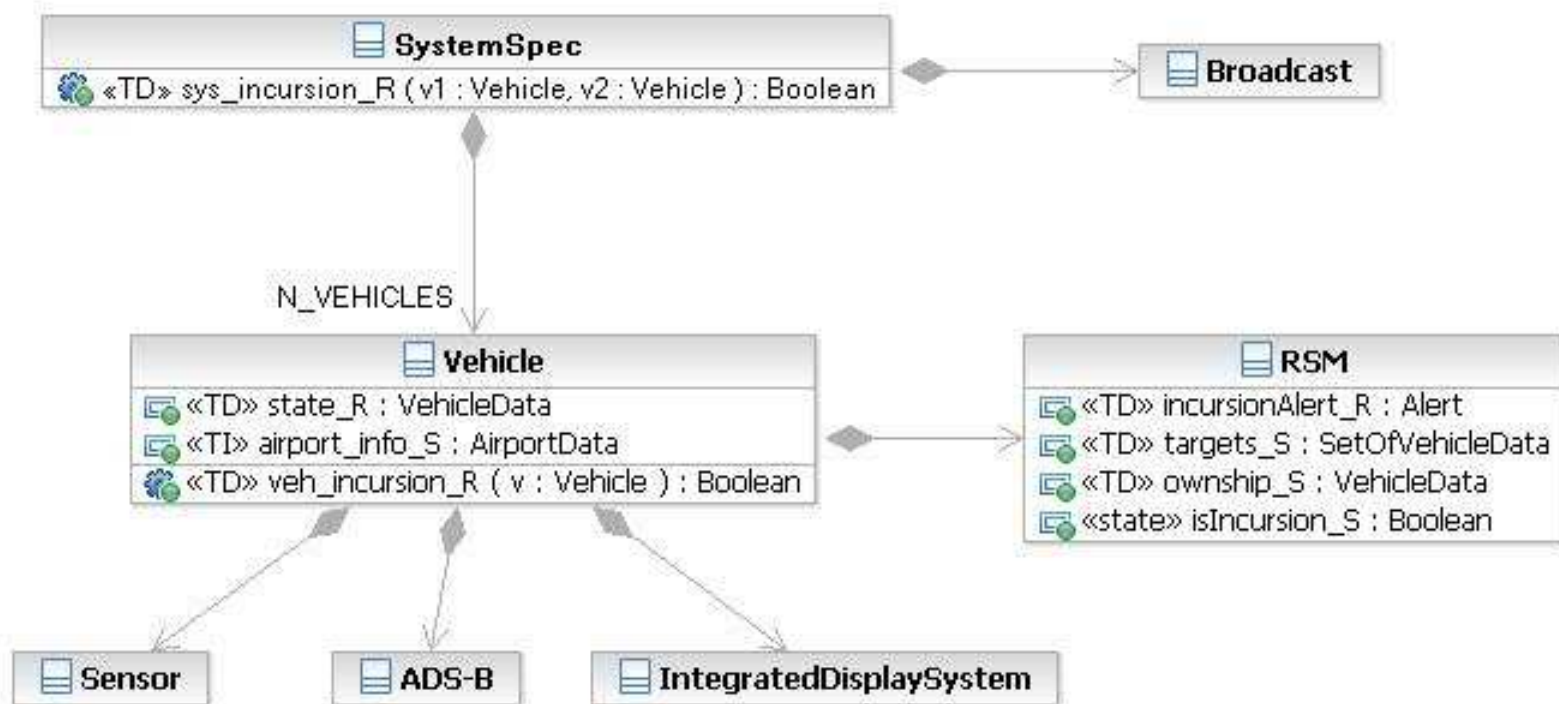
Lasted(lastData>30, 5)

->

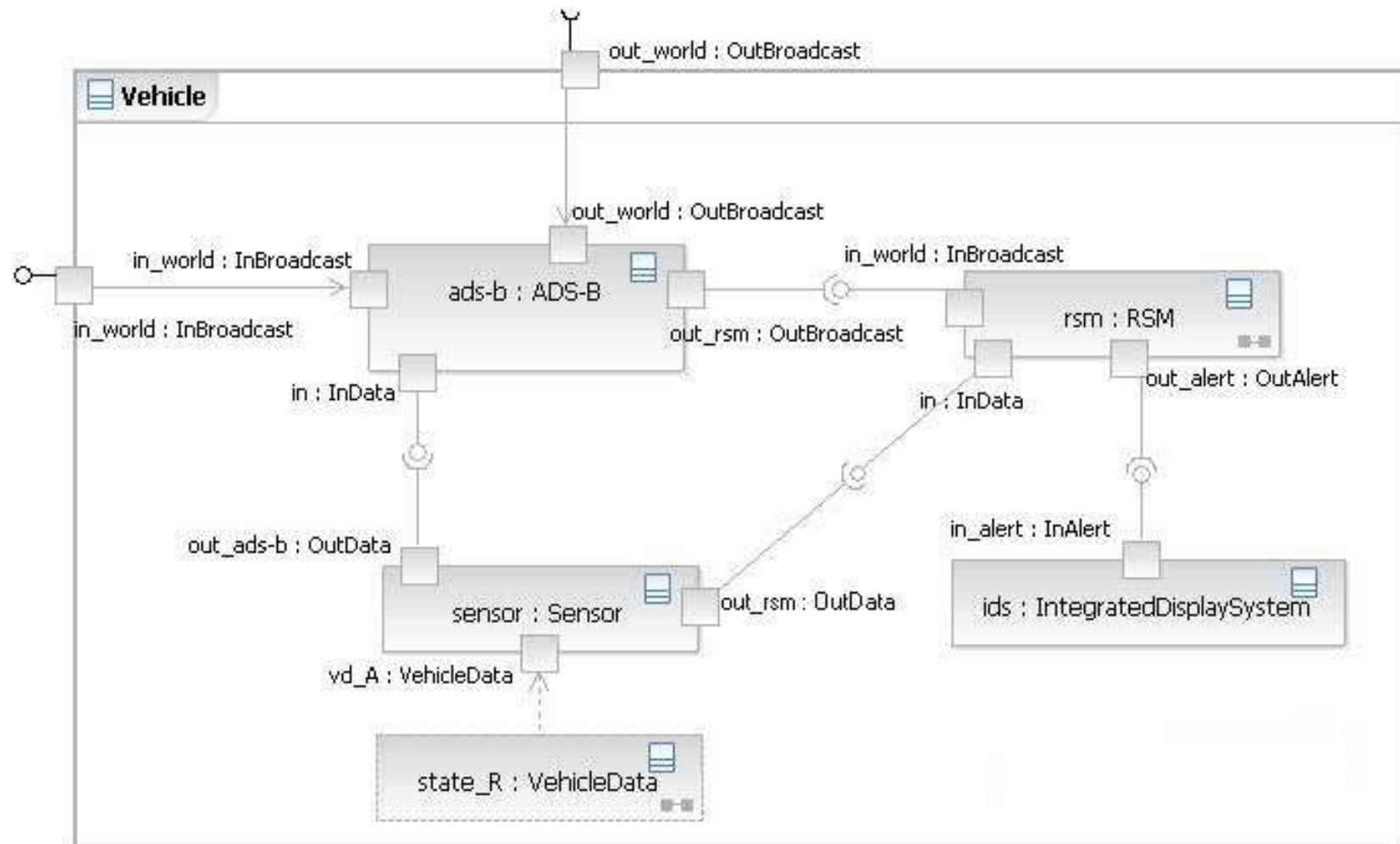
```

ex ra_op : AlarmMgmt.raiseAlarm
  (withinF(am.ra_op.executing, 2));
  
```

Esempio: Runway Safety Monitor



RSM (2)



RSM: formule logiche

- Formula della classe Vehicle:

```
incursion_detection_R:  
  Lasts(veh_incursion_R(v), L_INC)  
  ->  
  ex a : Alert  
    (withinF(rsm.incursionAlert_R = a, D_INC));
```

- Formule della classe Sensor:

```
data_sent_def_A:  
  all d:VehicleData, rsm_rD:out_rsm.recData  
    (rsm_rD.invoke(d) -> withinP(vd_A=d, D_DS));
```

```
data_collection_liveness:  
  withinF(ex d (rsm_rD.invoke(d)), D_SR);
```

Verifica formale con TRIO/ArchiTRIO

- Varie tecniche (di diverso “peso”)
 - history checking (validazione)/
test result verification (oracolo)
 - constrained simulation (generazione di tracce)
 - test case generation
 - prova di proprietà
 - theorem proving (logiche di ordine superiore, PVS)
 - model checking (SPIN)
 - bounded model checking (SAT-based)
- Strumenti prototipali a supporto della verifica

Verifica formale di RSM

- Il modello ArchiTRIO dell'RSM è stato tradotto nella logica di ordine superiore del Theorem Prover PVS
 - PVS è stato usato per verificare che il modello soddisfacesse il requisito espresso da `incursion_detection_R`
 - verifica condotta essenzialmente dal progettista *con il supporto* dello strumento
 - la verifica ha evidenziato una lacuna nelle assunzioni fatte sul dominio dell'applicazione

Lavori futuri

- Strumenti e metodi di progettazione
 - ArchiTRIO nasce per lo sviluppo di sistemi SW-intensive, in cui l'accento è sul *sistema*: naturale pensare ad una evoluzione verso SysML
- Integrazione più stretta tra strumenti di modellazione e di verifica
 - verso l'automatizzazione della verifica
- Disponibilità a collaborazioni industriali in cui applicare le nostre tecniche di progettazione/analisi