Seconda Universita' degli Studi di Napoli

Facoltà di Ingegneria - Dipartimento di Ingegneria dell'Informazione

TESI DI DOTTORATO IN INGEGNERIA ELETTRONICA

ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

UN APPROCCIO ALL'INGEGNERIA DEI MODELLI PER SISTEMI DI ELABORAZIONE CRITICI E COMPLESSI

Stefano Marrone

Tutor:

Prof. Ing. Nicola Mazzocca

Coordinatore:

Prof. Ing. Rocco Pierri

Indice generale

Indice delle figure

Indice delle tabelle

Lista degli acronimi

Viene qui riportata una lista degli acronimi e delle abbreviazioni usati in questa tesi.

Acronimo	Significato
BE	Basic Event
BN	Bayesian Network
BTM	Balise Transmission Module
COTS	Commercial Off The Shelf
CPN	Coloured Petri Nets
CPT	Conditional Probability Table
CTMC	Continuous Time Markov Chain
DMI	Driver Machine Interface
ERTMS/ETCS	European Railway Traffic Management System / European Train Control System
EVC	European Vital Computer
FIS	Functional Interface Specification
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FT	Fault Tree
FTA	Fault Tree Analysis
GSM-R	Global System for Mobile communications – Railway
GSPN	Generalized Stochastic Petri Nets
JRU	Juridical Recording Unit
LTM	Loop Transmission Module
MA	Movement Authority
MC	Markov Chain
MDA	Model Driven Architecture
MDE	Model Driven Engineering
MDT	Mean Down Time
MMI	Man Machine Interface
MOF	Meta Object Facility
MTBF	Mean Time Between Failures
MTBHE	Mean Time Between Hazardous Events
MTTF	Mean Time To Fail
MTTR	Mean Time To Repair
OMG	Object Management Group
PA	Process Algebra
PFT	Parametric Fault Trees
PDM	Platform Description Model
PIM	Platform Independent Model
PSM	Platform Specific Model
PN	Petri Nets
QoS	Quality of Service
QN	Queuing Network
RAMS	Reliability Availability Maintainability Safety
RB	Repair Box
RBC	Radio Block Centre
RFT	Repairable Fault Trees
RTM	Radio Transmission Module
SIL	Safety Integrity Level

SPN	Stochastic Petri Net
SSB	SottoSistema di Bordo
STM	Specific Transmission Module
SWN	Stochastic Well-formed Nets
TC	Track Circuit
TE	Top Event
THR	Total Hazard Rate
TIU	Train Interface Unit
TMR	Triple Modular Redundancy
TS	Technological Space
UML	Unified Modeling Language
UPS	Uninterruptible Power Supply
V&V	Verification and Validation
XML	Extended Markup Language

You can take the tiger out of the jungle, but you can't take the jungle out of the tiger.

Calvin & Hobbes.

Marco Polo descrive un ponte, pietra per pietra.
- Ma qual è la pietra che sostiene il ponte? - chiede Kublai Kan.
- Il ponte non è sostenuto da questa o quella pietra, - risponde Marco, - ma dalla linea dell'arco che esse formano.
Kublai Kan rimane silenzioso, riflettendo. Poi soggiunge: - Perchè mi parli delle pietre? E solo dell'arco che m'importa.
Polo risponde: - Senza pietre non c'è arco.

Italo Calvino, Le Città Invisibili

The real voyage of discovery consists not in seeking new landscapes but in having new eyes.

Marcel Proust

A Titty,
The heart has its reasons, of which reason knows nothing
Blaise Pascal

Introduzione

I progressi sempre più marcati delle tecnologie dell'informazione hanno reso l'uso pervasivo dei calcolatori nella vita quotidiana dell'uomo un dato di fatto. Si richiedono ai moderni sistemi di controllo prestazioni ed affidabilità sempre maggiori per compiti sempre più critici per la vita dell'uomo. I metodi formali sono da sempre strumento consigliato e richiesto dalla maggior parte degli standard internazionali legati allo sviluppo ed alla validazione dei sistemi critici. Tali metodologie hanno dimostrato nel corso degli anni la loro applicabilità in tali processi produttivi anche in ambito industriale.

Un'altra classe di sistemi è costituita da quelli complessi, caratterizzati dalla molteplicità e dall'eterogeneità dei componenti di cui essi sono composti. Per tali sistemi vale la proprietà di emergenza che può essere definita come la manifestazione di comportamenti a livello sistemico non predicibili dai comportamenti e dalle interazioni tra i diversi componenti del sistema stesso. A causa della loro natura, i sistemi complessi non riescono ad essere completamente inquadrati in descrizioni e modelli basati su un solo linguaggio: sono necessari più sistemi formali per fornire diverse "viste" non dipendenti tra loro. Sono inoltre necessari approcci composizionali bottom-up per poter valutare correttamente i comportamenti di sistema a partire dalla caratterizzazione dei singoli componenti.

Per tali ragioni i metodi formali hanno mostrato le loro difficoltà nell'essere applicati su sistemi critici e complessi: è necessario dunque estendere i metodi formali esistenti migliorando le metodologie di modellazione ed analisi, al fine di permettere il supporto di meccanismi quali la composizionalità tra modelli ed il multiformalismo (l'interoperabilità cioè tra modelli espressi in linguaggi diversi).

Una disciplina emergente nell'analisi e nella progettazione di sistemi di elaborazione di questi ultimi anni è il Model Engineering: esso si pone come obiettivo quello di fornire metodologie di creazione e manipolazione di modelli per lo sviluppo di applicazioni informatiche sempre più complesse e con livelli di qualità sempre più alti. Una delle finalità che tale disciplina si pone è quella di fornire metodologie di modellazione del tutto astratte dal tipo di linguaggio e dal contesto applicativo. L'approccio del Model Engineering si basa dunque sull'astrazione e sulla gerarchizzazione tra modelli, meta-modelli (le descrizioni dei modelli) e meta-meta-modelli (le formalizzazioni dei linguaggi). Concetto centrale in tale ambito è quello di trasformazione automatica di modelli: attraverso tale meccanismo è possibile infatti generare modelli più complessi e dettagliati a partire da altri più semplici applicando le opportune trasformazioni.

Scopo di questa tesi è di fornire un inquadramento orientato al Model Engineering per le metodologie già sviluppate nell'ambito del progetto OsMoSys, il cui obiettivo è quello di fornire metodologie e strumenti a supporto della modellazione e dell'analisi multiformale e multirisoluzione. Tale contributo parte dalla definizione di un dominio tecnologico per i linguaggi di OsMoSys; successivamente si caratterizzano sotto il punto di vista teorico le trasformazioni di modelli in OsMoSys, se ne fornisce una classificazione ed alcuni esempi notevoli delle loro applicazioni ai modelli formali. Si esamina successivamente il problema della composizione di modelli come specializzazione di quello delle trasformazioni: anche in tale ambito vengono fornite caratterizzazioni teoriche e classificazioni tra vari tipi di composizione.

Tali metodologie verranno applicate a problemi di composizionalità tra modelli multiformali, a composizione di modelli omogenei basati su reti di Petri ad alto livello e a trasformazioni tra modelli all'interno delle problematiche del multiformalismo implicito.

Le metodologie e le tecniche introdotte sono infine applicate ad un caso di studio ferroviario: il sistema di segnalamento ERTMS/ETCS che rappresenta per la molteplicità dei componenti, per l'eterogeneità degli stessi e per la criticità delle funzionalità e delle specifiche di affidabilità, un ottimo rappresentante della categoria dei sistemi critici e complessi.

Capitolo I:

Metodi formali per sistemi critici e complessi

In questo capitolo verrà introdotta la classe dei sistemi critici complessi e le problematiche inerenti alla loro validazione attraverso l'uso di metodi formali. Successivamente si focalizzerà l'attenzione sugli approcci di modellazione e di analisi multiformale presenti in letteratura. Verranno infine introdotte le metodologie di Model Engineering e l'uso di tali tecniche al fine di facilitare l'analisi, la progettazione e la verifica dei sistemi critici e complessi.

Sistemi critici industriali

Gli ultimi decenni hanno visto il calcolatore elettronico entrare prepotentemente in ogni attività dell'uomo: tale pervasività ci rende ormai incapaci di immaginare la quasi totalità di queste attività senza l'uso del computer (ad esempio il sistema bancario). Molte di esse sono critiche, ossia possono produrre con un loro eventuale fallimento danni all'uomo stesso; interi settori produttivi come i trasporti (ferroviari, automobilistici, aerei, etc...) o la farmaceutica sono da ritenersi tali. Inoltre la necessità delle aziende di essere sempre più competitive impone a questi prodotti di essere sempre più performanti e disponibili possibile, pena l'esclusione dai mercati internazionali.
Sono questi i **sistemi critici** una cui prima divisione è dettata dalla natura della criticità:

- *quality critical:* sistemi per i quali un eventuale fallimento non è pericoloso ma può risultare dispendioso in termini economici (es. sistemi di telecomunicazioni);
- *safety critical:* sistemi per i quali un eventuale fallimento può provocare danni a persone e/o cose (es. sistemi di segnalamento ferroviario);
- *security critical:* sistemi per i quali un eventuale fallimento può provocare perdita di integrità e/o riservatezza nei dati custoditi dal sistema stesso (es. sistema bancario).

Il ciclo di vita di un sistema critico è anch'esso critico in quanto le varie fasi di sviluppo sono spesso regolamentate da standard internazionali; tali standard impongono processi di verifica da parte del cliente e/o di enti certificatori, chiamati a verificare l'aderenza del processo ai suddetti standard e a verificare la dimostrazione del grado di affidabilità del prodotto secondo analisi qualitative e quantitative.

Le caratteristiche di rilievo per i processi di progettazione, verifica ed assessement di tali sistemi possono essere sintetizzate nell'acronimo **RAMS** (*Reliability Availability Maintainability Safety,* ossia Affidabilità, Disponibilità, Manutenibilità e Sicurezza) o anche dall'attributo di **dependability**.
La dependability di un sistema è formalmente definita come *"la capacità di un sistema di fornire un servizio che possa essere ritenuto ragionevolmente fidato"* . La dependability ha diversi attributi in relazione alle diverse accezioni di fidatezza di un servizio e di un sistema. I più importanti di essi sono: [1]

- *reliability (affidabilità)*: continuità del servizio nel tempo, un cui indice è il Mean Time Between Failures (MTBF);
- *availability (disponibilità)*: disponibilità nel fornire il servizio, un cui indice è la disponibilità asintotica (A);

[1] Molti altri attributi sono stati aggiunti nel corso del tempo all'accezione di dependability: nel nostro studio terremo però conto solo delle proprietà definite. Citiamo ad esempio la survivability che è l'abilità di un sistema di continuare a fornire il proprio servizio anche durante e dopo disturbi naturali e/o artificiali (esplosioni nucleari, catastrofi naturali, attacchi terroristici, etc…) e la performability, ossia la valutazione delle performance di un sistema in condizioni di degrado che permettano comunque la fornitura del servizio ossia che non portino ad un fallimento .

- *maintainability (manutenibilità)*: abilità (di un sistema) nell'essere sottoposto a modifiche e/o riparazioni, un cui indice è il Mean Time To Repair (MTTR);
- *integrity (integrità)*: incapacità (del sistema) di permettere alterazioni indesiderate dei dati in suo possesso;
- *confidentiality (confidenzialità)*: incapacità (del sistema) di permettere accesso ad utenti indesiderati dei dati in suo possesso;
- *security (sicurezza)*: assenza di comportamenti catastrofici per i dati del sistema nell'ambiente in cui il sistema opera;[2]
- *safety (sicurezza)*: assenza di comportamenti catastrofici per persone e per cose nell'ambiente in cui il sistema opera, un cui indice è la Mean Time Between Hazardous Events (MTBHE).

La dimostrazione degli attributi RAMS di un sistema critico viene effettuata portando a termine una serie di attività di Verifica e Validazione (V&V) anch'esse regolamentate da standard internazionali (come ad esempio le norme europee CENELEC per le apparecchiature di segnalamento ferroviario , e). In breve mentre la fase di Verifica risponde alla domanda: "Stiamo sviluppando il sistema correttamente?", quella di Validazione risponde a: "Stiamo sviluppando il sistema corretto?". Tali attività sono pertanto critiche sia in termini di risultati che in quelli di budget: è desiderabile in altre parole stimare gli indici di RAMS quanto prima nel ciclo di vita del prodotto, al fine di provvedere ad azioni correttive che impattino quanto meno sui tempi e sui costi del prodotto stesso.

Le metodologie e le tecniche per la dimostrazione e la stima delle proprietà RAMS di un sistema critico prima dell'esercizio dello stesso, si basano sui modelli; esse possono essere divise in due categorie: quella dei metodi analitici e quella dei metodi simulativi.
I metodi simulativi, tra i quali si annoverano tecniche come il software/hardware testing, si propongono di approssimare la realtà attraverso l'uso di modelli di simulazione, sistemi che riproducano l'esatto comportamento input-output del sistema reale pur semplificandone la struttura interna. Al di là della semplicità di utilizzo di tali tecniche, i metodi simulativi pongono dei vincoli molto forti nei confronti del processo di validazione di un prodotto:

- il testing entra molto tardi nel ciclo di vita di un prodotto, pertanto i costi del bug-fixing restano ancora elevati;
- per sistemi molto grandi l'ambiente di simulazione diventa spesso un prodotto a sé stante: ciò comporta, oltre ai costi aggiuntivi, il problema della "validazione" degli strumenti sviluppati;
- la simulazione non fornisce prova matematica esaustiva della correttezza del sistema.[3]

L'ultimo fattore diventa estremamente importante per i sistemi safety critical quando non vengono individuati tutti i possibili azzardi[4] e quindi, per quei particolari azzardi non testati, l'evoluzione del sistema stesso può provocarne al fallimento con possibili conseguenze per la vita umana.
Volendo riassumere nella seguente tesi:

"Il testing software è un metodo molto efficace per provare la presenza di errori ma è irrimediabilmente inefficiente per dimostrarne l'assenza."

Edsgar W. Dijkstra

Metodi formali

Una definizione immediata di **metodi formali** è *"l'uso di tecniche matematiche nella progettazione ed analisi di sistemi"* . Queste metodologie si propongono di modellare alcuni

[2] La security può essere definita anche come intersezione degli attributi di availability, integrity e confidentiality
[3] Un esempio chiarificatore ci può venire se supponiamo di voler testare una routine per la divisione a 64 bit. Volendo testare ognuno dei 2^{128} pattern di ingresso ci vorrebbero, con una frequenza di 1 test/μs, circa 10^{25} anni.
[4] Si definisce azzardo di un sistema lo stato che può condurre ad un incidente .

aspetti del sistema descrivendolo secondo un formalismo matematico che, una volta analizzato, fornisca informazioni corrette e complete sul sistema reale. In altre parole l'utilizzo di un metodo formale nell'analisi e nello sviluppo di sistemi critici prevede l'adozione, da un lato, di un formalismo (il linguaggio di **specifica formale**) che permetta di modellare il sistema e i requisiti per esso definiti e, dall'altro, di metodologie e strumenti automatici di **verifica formale**; quest'ultima può essere definito come l'insieme degli algoritmi esatti che consentono di stabilire se un modello specificato in linguaggio formale soddisfa le proprietà per esso (formalmente) specificate.
La modellazione di sistemi hardware/software impone l'adozione di formalismi differenti, che consentano di specificarne il comportamento altamente discontinuo e basato su eventi; tra i più conosciuti ed utilizzati possiamo menzionare:

- i linguaggi basati sulla *logica matematica* che permettono di descrivere il comportamento e le proprietà dei sistemi attraverso la stesura di assiomi ed asserzioni;
- i linguaggi *algebrici* che permettono la modellazione dei sistemi come *transition system* caratterizzati da uno spazio degli stati, più o meno esplicitamente specificato, e da una serie di relazioni che esprimono le modalità di transizione da uno stato all'altro;
- linguaggi basati su notazioni *grafiche* (come le reti di Petri) che permettono una modellazione molto più intuitiva e comprensibile rispetto agli altri linguaggi e perciò molto più accessibile a persone non altamente esperte nell'ambito dei metodi formali.

Per quanto riguarda la verifica formale, essa rappresenta la fase del processo di sviluppo che consiste nello stabilire, mediante strumenti matematici, se il modello di un sistema soddisfa i requisiti per esso (formalmente) specificati. Gli approcci sviluppati in tale contesto sono riconducibili alle seguenti classi:

- *Equivalence checking* - attraverso il quale si cerca di dimostrare se due modelli sono funzionalmente equivalenti ossia hanno la stessa evoluzione .
- *Model checking* - attraverso il quale si valutano, nello spazio degli stati che caratterizza il modello del sistema, tutte le possibili tracce di esecuzione alla ricerca di quelle (i controesempi) che violano le proprietà specificate .
- *Theorem proving* - attraverso il quale si cerca di dimostrare che la specifica dei requisiti è una conseguenza logica della specifica (assiomatizzazione) dell'implementazione del sistema .

I vantaggi derivanti dall'uso dei metodi formali possono essere qui riassunti:

- forniscono una specifica del sistema non ambigua sia in termini di variabili (stato) che in quelli di operazioni (funzionalità);
- forniscono attraverso l'analisi esaustiva del modello la "prova di correttezza" del sistema basata su principi matematici esatti;
- sono utilizzabili sin dalle prime fasi del ciclo di sviluppo del prodotto: in tal modo può essere minimizzato l'impatto che la correzione di un eventuale errore ha sui costi e sui tempi del progetto;
- calcolano in tempi ridotti gli indici RAMS permettendo lo sviluppo di prodotti ad alta qualità ed in tempi contenuti.

Inoltre la maggior parte degli standard internazionali considerano come necessario o altamente raccomandato l'uso dei metodi formali per la progettazione e la verifica dei sistemi critici complessi. Consideriamo ad esempio lo standard CENELEC 50128 di forte rilevanza nell'ambito ferroviario: in tale standard la sicurezza viene quantificata assegnando ad ogni componente del sistema un indicatore, il SIL (Software Integrity Level) che va da 0 a 4 in ordine crescente di criticità. Il SIL è associato da una parte al tasso di guasti pericolosi (THR) desiderato per il sottosistema e dall'altro all'indicazione delle diverse metodologie applicabili nelle varie fasi di vita del prodotto. Focalizzando la nostra attenzione sulla categoria dei sistemi SIL4 (quelli maggiormente imputati all'espletamento delle funzionalità di sicurezza) ne

deduciamo che i metodi formali sono altamente raccomandati (ossia deve essere fornita una giustificazione al non utilizzo di tali metodologie) sia nelle fasi di sviluppo che in quelle di verifica e validazione.

Tra tutti i metodi formali ne esistono alcuni che meglio di altri sono adatti a catturare gli aspetti di dependability e che, attraverso un'analisi formale, forniscono gli indici RAMS di un sistema. I principali tra questi sono:

- *Reti di Petri* (PN): con tutte le loro varianti (GSPN ed SWN specialmente) sono indicate per modellare il comportamento di un sistema dove lo stato è distribuito tra i vari componenti ;
- *Reti Bayesiane* (BN): utilizzate storicamente per la rappresentazione della conoscenza nell'AI , hanno negli ultimi anni dimostrato di poter essere facilmente usate anche nella predizione della reliability ;
- *Alberi dei Guasti* (FT): strumento consolidato nella valutazione della disponibilità ed affidabilità di un sistema grazie alla semplicità d'uso ed all'alta risolvibilità del modello .

La scelta di uno o di un altro formalismo come linguaggio di modellazione per la valutazione della dependability dipende da diversi fattori:

- facilità d'uso e comprensibilità del linguaggio;
- potere espressivo (può il formalismo catturare con semplicità gli aspetti significativi del modello?) e di conseguenza dall'indice RAMS che si vuole valutare;
- scalabilità (di centrale interesse per casi industriali);
- risolvibilità del modello.

Un possibile confronto tra i formalismi sopra presentati mette in luce che le Bayesian Networks sembrano essere un buon compromesso tra il potere espressivo e modellante (in cui le Petri Nets sono eccellenti) e l'analizzabilità e scalabilità del formalismo rispetto al modello (punto forte dei Fault Trees).

Sistemi complessi

I sistemi, sia quelli critici che non, subiscono, a seconda di quale aspetto viene preso in considerazione, le più diverse tassonomie. Una di queste è quella che li vuole divisi in sistemi semplici e complessi. Tale divisione, intuitivamente molto semplice e che non necessiterebbe di ulteriori approfondimenti, incontra però diversi problemi di formalizzazione. Infatti, nonostante in letteratura siano presenti diversi approcci teorici alla complessità (si veda ad esempio la teoria di Kolmogorov), lo studio delle proprietà comuni ai **sistemi complessi** resta un campo di ricerca ancora molto aperto.

La comunità scientifica è comunque concorde nell'individuare alcune caratteristiche tipiche che possono presentarsi, tutte insieme o in parte, nei sistemi complessi :

- *agent-based*: il sistema è caratterizzato dall'avere diversi agenti ognuno dei quali autonomo nel compito che svolge;
- *non-linearità*: il comportamento dei singoli agenti non è modellabile in modo lineare (effetto butterfly);
- *eterogeneità*: gli agenti differiscono in comportamento, natura ed obiettivo;
- *dinamicità*: le caratteristiche del sistema e degli agenti non sono costanti nel tempo (adattamento degli agenti all'ambiente, feedback ed apprendimento, processi di selezione naturale);
- *retroazione*: le evoluzioni dei comportamenti degli agenti sono spesso conseguenza di retroazioni che gli agenti stessi ricevono dall'ambiente in cui operano;
- *organizzazione*: gli agenti sono organizzati in gruppi o gerarchie con strutture spesso di diversa complessità;
- *emergenza*: l'interazione tra gli agenti provocano macro comportamenti a livello sistemico non predicibili.

Esempi di sistemi complessi possono essere tratti da ogni campo delle scienze naturali ed ingegneristiche: sistemi economici e politici, reti di relazioni interpersonali, pathways metabolici, ecosistemi, sistemi di trasporto (aereo, ferroviario, etc…), sistemi biomedici, reti di comunicazione.
I metodi di indagine della scienza classica si fondano sul **riduzionismo Cartesiano:** ogni cosa, a parte la mente dell'uomo, può essere schematizzata come una macchina, un sistema potremmo dire noi; tale sistema può essere decomposto in parti più semplici ed ognuna di queste parti analizzata nel suo comportamento senza che vengano perse informazioni sull'evoluzione globale. Altro caposaldo del metodo scientifico classico è il **paradigma Newtoniano** basato sul principio di generalizzazione delle leggi verificate. In altre parole ogni disciplina scientifica, nell'ottica di semplificare l'argomento di studio, applica i seguenti passi: scoperta della legge, verifica, generalizzazione.
Secondo questo processo ogni sistema naturale può essere dunque decomposto e codificato in un sistema formale: dalle regole del sistema formale si derivano nuove leggi, implicazioni causali nel sistema formale che ci danno informazioni su eventi causali nel sistema naturale.
L'approccio combinato di questi due fattori, ossia il metodo scientifico, ha portato alla considerazione erronea che la complessità non fosse una caratteristica intrinseca del sistema ma una proprietà dovuta all'eccesso di fattori in gioco. In termini più semplici sono stati accomunati concetti come complessità e complicatezza. Con i sistemi complessi tale binomio è fallito principalmente a causa di due fattori :

- eccesso di informazione: il paradigma della specializzazione *"conosci sempre di più su sempre di meno"* ha portato alla costruzione di barriere di conoscenza tra le diverse discipline (e tra le diverse sottodiscipline);
- eccesso di semplificazione: in sistemi complessi la tendenza a semplificare la funzionalità di un componente elementare sotto esame contrasta con la proprietà di emergenza nel comportamento facendo perdere informazioni sull'evoluzione del sistema stesso.

In realtà per i sistemi complessi la fase di **decomposizione,** qualora non si tenga conto del comportamento globale (emergente o non) e delle funzionalità del sistema, è destinata a fallire. Infatti la complessità del (sotto)sistema è legata a quella del compito che svolge dalla Legge della Varietà[5] Necessaria: *"solo la varietà può distruggere la varietà"*. In altre parole, un sistema che svolge un compito di una certa varietà (varietà dell'insieme di ingresso) deve possedere una varietà almeno uguale .
Il risultato della decomposizione quasi mai può essere analizzato come un sistema semplice; l'analisi dei sistemi complessi si avvale dunque di metodi integranti quelli tradizionali: un sistema deve essere studiato sotto diverse lenti, attingendo da diverse discipline metodologie per una caratterizzazione non polarizzata da un singolo sistema formale. Pertanto quello dei sistemi complessi è un settore altamente multidisciplinare dove informatica, teoria dei giochi, matematica, statistica, cibernetica, teoria dei sistemi e teoria del caos sono tutti elementi necessari. In altri termini è necessario usare diversi sistemi formali (**multiformalismo**) per modellare un singolo sistema naturale.
Volendo riassumere in una frase:

"…la complessità è una proprietà del mondo reale che si manifesta nell'incapacità di un singolo formalismo a catturare tutte le proprietà di un sistema. E' necessario evidenziare le diverse vie di interazione con il sistema tanto che, quando abbiamo prodotto un buon modello, ogni sistema formale necessario per descrivere un singolo aspetto del sistema naturale, deve essere NON derivabile dagli altri…"

Bob Rosen e Don Mikulecky. Prof. in fisiologia presso
Medical College of Virginia Commonwealth University

[5] La varietà è un indice della complessità di un sistema/attività. Più formalmente si definisce varietà di un insieme il numero (o il suo logaritmo in base 2) dei suoi elementi distinti. Per i sistemi la varietà è la misura del numero degli stati distinti in cui un sistema può trovarsi.

I metodi per l'analisi e lo sviluppo dei sistemi complessi artificiali devono essere dunque rivoluzionati rispetto a quelli dell'ingegneria tradizionale. Infatti quando il progetto diventa eccessivamente complesso, i metodi tradizionali aumentano considerevolmente la probabilità di non soddisfare i vincoli di tempo, costo o qualità imposti per il progetto stesso (vedi per una lista dei maggiori progetti falliti degli ultimi anni). Occorre dunque cercare metodologie e tecniche in grado di stimare il comportamento finale del sistema con il grado di precisione maggiore e quanto prima possibile nel suo ciclo di vita. Come per lo studio dei sistemi complessi naturali anche per quelli artificiali è necessario applicare sia criteri di decomposizione che di modellazione attraverso distinti sistemi formali.

Metodi formali per i sistemi critici e complessi

L'applicazione dei metodi formali ai sistemi critici e complessi si limita pertanto a concentrarsi su poche caratteristiche per volta, mantenendo spesso un livello di astrazione non sempre soddisfacente per la validazione del sistema stesso.
E' possibile trovare numerosi esempi di quanto detto in letteratura in diversi ambiti di applicazione. Analizzeremo adesso brevemente lo stato dell'arte della ricerca dell'applicazione dei metodi formali in un settore industriale specifico: quello ferroviario, da sempre sensibile all'uso dei metodi formali come strumento per la dimostrazione del soddisfacimento dei requisiti di affidabilità e sicurezza.
Bjørne fornisce un ottimo punto di partenza nello studio dell'applicazione dei metodi formali al settore ferroviario. A tale bibliografia aggiungiamo altri lavori più recenti, soprattutto concernenti il nuovo standard di segnalamento ferroviario ERTMS/ETCS[6]. Tale sistema, di cui rimandiamo la descrizione al Capitolo VI, si pone come il nuovo standard di riferimento, non solo per l'Europa, per applicazioni ferroviarie ad alta affidabilità, prestazione, interoperabilità e sicurezza.
L'applicazione dei metodi formali all'ambito ferroviario ha però visto principalmente come oggetto di studio il problema dell'interlocking ferroviario a causa della vasta diffusione di tale tipo di applicazione. Eriksson et al. propongono un approccio basato sulla modellazione delle regole di interlocking mediante logiche predicative di primo ordine e sulla verifica delle stesse mediante tecniche di theorem proving; questo tipo di approccio (o altri basati su logiche di ordine superiore al primo) è stato seguito anche in e . Hlavatý et al. propongono invece una metodologia basata sul model checking per la verifica di specifiche di interlocking ferroviario mentre Bernardeschi et al. propongono una metodologia ed uno strumento di verifica formale basata su Communicating Concurrent Process (CCS) che però fa uso di tecniche di astrazione per rendere verificabile automaticamente il caso di studio.
Numerosi solo gli approcci basati sulle reti di ad Petri tra cui ricordiamo quelli che si basano sulle reti di Petri ad alto livello per la loro capacità di migliorare il potere espressivo e la capacità computazionale degli approcci basati su reti di Petri P/T e GSPN. Esempi di tali lavori sono , e .
Un progetto che rappresenta forse il caso più esemplare di applicazione dei metodi formali nel campo delle applicazioni ferroviarie è costituito da B . B è una sigla che comprende una metodologia di sviluppo e di verifica formale (*B-Method*) e strumenti a supporto di tale metodologia (*B-Toolkit*). La metodologia è basata, per quanto riguarda il modello di specifica, su un metodo assiomatico basato su un formalismo logico-matematico: infatti B usa l'*Abstract Machine Notation* (AMN) sia per la modellazione delle specifiche che delle implementazioni; per tali modelli vengono specificate proprietà invarianti per lo stato e pre/post-condizionali per le operazioni. Il metodo di verifica si basa su tecniche di theorem-proving. Tale linguaggio di modellazione è alla base di un intero processo di sviluppo di applicazioni ferroviarie metropolitane sviluppate relativamente al progetto SACEM prima e METEOR poi che ha portato alla progettazione della linea metropolitana n°14 di Parigi completamente automatizzata.

[6] European Railway Traffic Management System / European Train Control System

A quanto detto si aggiunge l'analisi del nuovo sistema di segnalamento ERTMS/ETCS. La dimostrazione delle proprietà RAMS di un sistema ERTMS/ETCS è un compito non banale in quanto l'architettura del sistema, la complessità delle funzionalità, le caratteristiche di distribuzione del controllo, di eterogeneità dei sottosistemi e dei requisiti che il sistema deve soddisfare, fanno di un sistema basato su tale standard un sistema complesso.
In Jansen et al. ed in Hörste si studia il sottosistema di controllo di tale standard (European Train Control System); in particolare viene utilizzato un approccio modellazione formale / analisi simulativa attraverso l'uso delle reti di Petri Colorate (CPN) e l'utilizzo del tool Design/ CPN ; il primo lavoro si concentra sulle specifiche mentre il secondo modella il sistema ad un livello di astrazione molto alto. In Esposito et al. l'attenzione viene focalizzata sul protocollo EURORADIO relativo alla comunicazione tra apparati dell'architettura di riferimento. Per tale protocollo si procede ad una modellazione semi-formale con UML e ad un'analisi basata anch'essa sulla simulazione. In Zimmermann et al. viene condotto uno studio sull'affidabilità del sistema di comunicazione GSM-R, mezzo attraverso il quale sistemi di bordo e di terra comunicano. Tale studio viene effettuato con l'uso delle DSPN (Deterministic and Stochastic Petri Nets) come formalismo per la modellazione e con l'uso della simulazione come tecnica d'analisi. In Chiappini et al. vengono invece modellate attraverso il formalismo STATEMATE, le specifiche del Radio Block Centre (RBC) elemento centrale del sistema ERTMS livello 2. A tale formalizzazione segue poi un'analisi basata su approcci simulativi e di model checking. E' da notare che in viene evidenziata la necessità di dover modellare l'intero sistema ERTMS al fine poter validare alcune proprietà del sottosistema RBC; ciò rafforza la tesi del il sistema ERTMS/ETCS è effettivamente un sistema complesso.
Finora dunque per il sistema ERTMS/ETCS (ed in generale tranne che per poche eccezioni per i sistemi ferroviari tutti) sono stati prodotti lavori relativi alla verifica di specifiche e basati solo su processi di modellazione formale e non anche di analisi formale; inoltre gli approcci proposti lavorano su modelli ad un grado di astrazione tale per cui occorre un ulteriore sforzo nell'ottica della dimostrazione delle proprietà del sistema. C'è da segnalare che nessun tentativo di modellazione ed analisi mediante multiformalismo è stato finora fatto.

Sono dunque qui indicati i problemi che possono essere ascritti all'uso dei metodi formali per l'analisi di sistemi complessi e critici . Essi rappresentano anche le aree di maggiore ricerca nel settore e sono riassumibili nei seguenti punti:

- *composizionalità*: bisogna comprendere quali sono i meccanismi che stanno alla base della composizione di linguaggi formali, modelli e risultati;
- *multiformalismo*: l'uso di un solo formalismo non riesce spesso ad evidenziare tutti gli aspetti di un sistema ma permette di concentrarsi su alcune caratteristiche o di modellare meglio sottosistemi di natura particolare. E' solo attraverso l'uso integrato di più formalismi che è possibile avere una visione d'insieme di un sistema e di poter analizzare comportamenti "emergenti" che altrimenti resterebbero latenti;
- *decomposizione*: una delle più grosse limitazione all'uso dei metodi formali nella verifica automatica di specifiche è il problema dell'esplosione dello spazio di stato. Alcuni algoritmi di analisi evidenziano una necessità di risorse computazionali (tempo e soprattutto memoria) tali da rendere proibitivi questo tipo di approcci nell'applicazione a problemi di grosse dimensioni. Occorre capire in che termini è possibile scomporre un modello in sottomodelli più facilmente risolvibili;
- *integrazione nei processi:* non è possibile supporre che, soprattutto per sistemi safe-critical, i processi di sviluppo e di V&V possano essere sostituiti ex-abrupto da processi di verifica formale. La complessità di tali processi e la loro dipendenza dagli standard e dai procedimenti di assessment e certificazione impone ai metodi formali la necessità di integrarsi all'interno dei processi preesistenti. Una tendenza può essere quella di preferire sistemi orientati all'individuazione di controesempi per particolari proprietà (model-checking) rispetto a quelli verificanti la "correttezza" del modello (theorem-proving); ciò

perchè un controesempio è più facilmente verificabile da un test, strumento semplice e di maggiore utilità nell'individuazione di bug (vedi la già citata tesi di Dijkstra).

- *riuso dei componenti*: invece di reinventare da zero nuovi modelli, è necessario, al fine di migliorare il time-to-market, poter contare su una "libreria" di modelli di facile instanziazione e parametrizzazione.

I primi due (composizionalità e multiformalismo) svolgono un ruolo cruciale nella definizione degli scenari futuri d'applicazione dei metodi formali a tali tipi di sistemi e saranno oggetto di studio in questa tesi.

Approcci alla modellazione ed analisi multiformale

Descriviamo adesso alcuni approcci presenti in letteratura relativi a precedenti metodologie per l'integrazione di modelli in framework per lo sviluppo e l'analisi di sistemi complessi; tale disciplina è anche detta Computer Automated Multiparadigm Modeling (CAMPaM) .
Il progetto SMART integra i seguenti formalismi: Reti di Petri Stocastiche, Catene di Markov a Tempo Continuo e Discreto in un ambiente unico di modellazione. I vari modelli possono scambiare dati durante i processi di analisi.
SHARPE è uno strumento per l'analisi di reliability e performability dei modelli che integra diversi formalismi quali gli Alberi dei Guasti, le Reti di Petri Stocastiche Generalizzate, alcuni tipi di Reti di Code e Processi Markoviani. Anche questo ambiente permette lo scambio di informazioni e risultati tra i vari modelli nell'ambito dell'analisi dei modelli.
L'approccio di Möbius è più generale degli altri due in quanto fornisce strumenti per l'integrazione di differenti tools per la modellazione e l'analisi dei modelli; in particolare i risolutori di Möbius, in un modello multiformale, non fanno ipotesi sul formalismo dei sottomodelli riportando il modello multiformale in un formalismo omogeneo affine a quello delle catene di Markov tempo continuo. La scelta dei possibili formalismi da utilizzare resta comunque confinata ad un insieme di formalismi "compatibili", quali: Stochastic Activity Network (SAN)[7], Bucket and Balls Formalism[8] e Performance Evaluation Process Algebra (PEPA)[9].
AToM[3] implementa un approccio di modellazione basato sul meta-modelling e su tecniche di trasformazione tra grafi. Consente di modellare differenti componenti di un sistema utilizzando diversi formalismi e la sua interfaccia utente fornisce un editor grafico per ogni formalismo definito. I modelli possono essere traslati da un formalismo all'altro grazie ad un'apposita grammatica sui grafi, che definisce le regole di trasformazione. L'analisi dei modelli può essere effettuata sia attraverso tecniche di tipo simulativo che traslando i componenti in un formalismo comune ed applicando successivamente il solver specifico.
L'approccio di DEDS è quello di fornire un framework per la costruzione di modelli di performance di sistemi ad eventi discreti (Discrete Events Dynamic Systems) attraverso un'interfaccia grafica. I formalismi utilizzati sono: Reti di Code, Reti di Petri, Reti di Petri Colorate. L'analisi viene effettuata traducendo tutti i sottomodelli in un formalismo simile a quello delle Reti di Petri: il modello risultante viene analizzato poi attraverso la costruzione dello spazio di stato.

L'approccio OsMoSys

In questa sezione si presenta un approccio al multiformalismo/multisoluzione che è parte di OsMoSys (Object-based multi-formaliSm MOdelling of SYStems) , un framework multiformalismo e multisoluzione per l'analisi basata su modelli di sistemi. OsMoSys è

[7] Estensione delle Reti di Petri Stocastiche
[8] Estensione delle Catene di Markov
[9] Estensione stocastica delle Process Algebra

sviluppato dalle università di Napoli Federico II, Seconda Università di Napoli, Università del Piemonte Orientale ed Università di Torino. Nel seguito verrà descritta la metodologia di modellazione , l'architettura e l'approccio multisoluzione del framework .
La metodologia di modellazione, indicata anche come **OMM** (OsMoSys Multiformalism Metodology), si basa sul principio della metamodellazione e dei metalinguaggi (ci sono tre livelli di linguaggi: modelli, meta-modello, meta-meta-modello, vedi Figura 1).

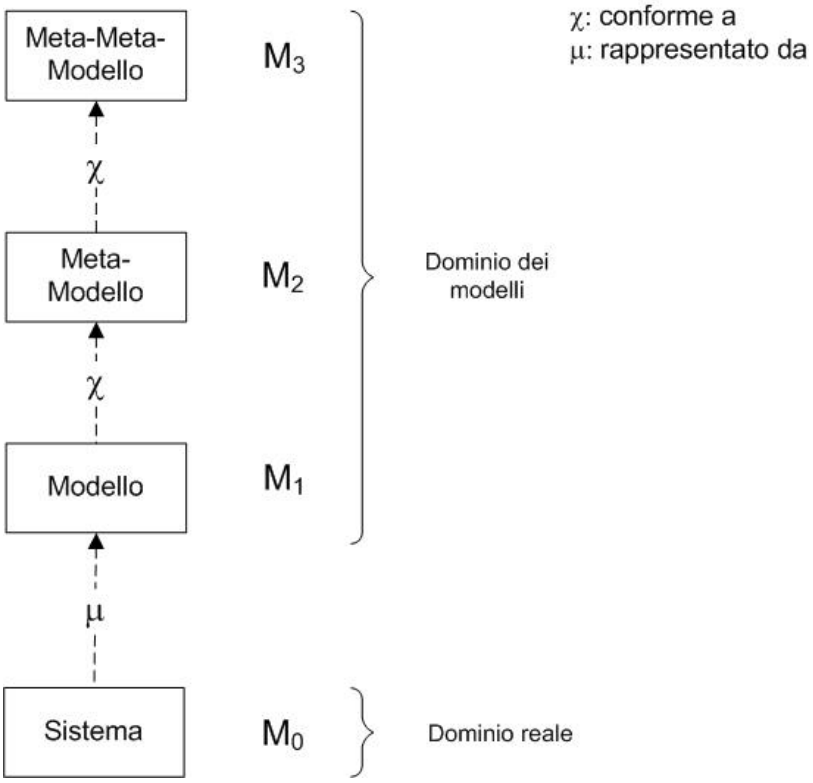

Figura 1: Pila dei modelli a quattro livelli

Tale metodologia è applicabile a formalismi basati su grafo come ad esempio Reti di Petri, Alberi dei guasti e Reti Bayesiane; essa prevede ulteriormente aspetti di object-orientation. Inoltre OsMoSys è fortemente estensibile in quanto lo sviluppatore può creare, per il proprio formalismo, la descrizione in XML del proprio linguaggio ed integrare dunque le proprie risorse sia per quanto riguarda gli aspetti modellativi che per quelli risolutivi.
OsMoSys è anche un framework multisoluzione nel senso che integra non solo diversi tipi di formalismo, ma fornisce libertà di scelta su quale risolutore (numerico, analitico, simbolico, etc…) usare per un determinato tipo di formalismo e per il particolare tipo di analisi da effettuare sul formalismo. Questo aspetto di OsMoSys è anche detto **OMF** (OsMoSys Multisolution Framework). I diversi strumenti di soluzione sono integrati in OsMoSys attraverso l'uso di adapters, entità software costruiti ad-hoc per ogni solver in grado di fornire verso OsMoSys un'interfaccia comune. Il processo di risoluzione si basa sul coordinamento di tali adapters da parte di una workflow engine.
L'architettura (vedi Figura 2) è basata su un approccio a livelli; possiamo distinguere:

- *livello grafico*: contiene le interfacce grafiche che permettono all'utente di disegnare ed analizzare nuovi modelli, cuore di tale livello è il tool grafico DrawNET Xe! ;
- *livello del workflow engine*: la Workflow Engine (WFE) gestisce il processo di risoluzione dei modelli coordinando l'attivita dei vari risolutori seguendo i principi della gestione dei workflow ;
- *livello degli adattatori*: gli adapters incapsulano i risolutori fornendo verso OsMoSys un'interfaccia comune in modo che il loro uso possa essere generale e facilmente formalizzabile;

- *livello database*: diversi repository sono gestiti da OsMoSys nell'ottica di conservare risultati di analisi e fornire una struttura d'appoggio all'intero processo di risoluzione; altra funzione di tale livello è quello di costituire un bacino di modelli dove l'utente possa attingere per costruire nuovi modelli, aggregando e specializzando modelli predefiniti più semplici.

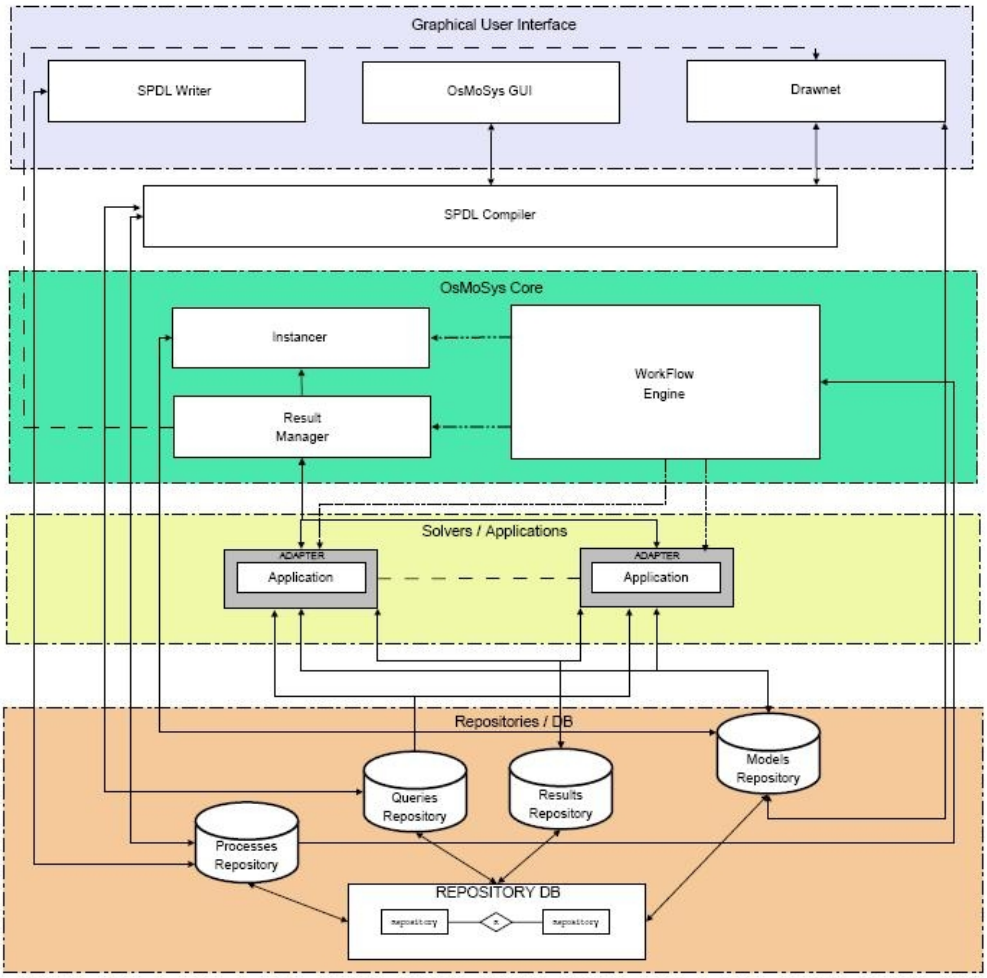

Figura 2: Architettura di OsMoSys

OsMoSys costituisce uno strumento effettivo per realizzare il multiformalismo sia implicito che esplicito. Si intende per *multiformalismo esplicito* quando l'utente finale è cosciente di usare un modello multiformale (egli modella esplicitamente diverse porzioni del sistema con diversi formalismi a lui noti). Per *multiformalismo implicito* s'intende ovviamente quello per cui l'utente non è cosciente di utilizzare un modello multiformale che viene realizzato attraverso l'estensione di un singolo formalismo.
OsMoSys fornisce anche "primitive" di composizione di modelli realizzate mediante formalismi bridge in grado di legare le interfacce delle model class (le cosiddette model interface) tra loro. In viene accuratamente definito tale meccanismo e la sua integrazione nell'architettura e nel processo di risoluzione di OsMoSys.

Model Engineering: una disciplina emergente

Negli ultimi anni è divenuta sempre più impellente la necessità di avere una disciplina ingegneristica in grado di astrarre le caratteristiche ed i benefici dell'ingegneria del software ad un livello di formalizzazione più alto: un approccio degli ultimi anni è stato quello della Model-Driven Engineering[10] (MDE) proposto in principio da Kent . Tale metodologia nasce come

[10] Nel corso di questa tesi si farà riferimento a Model Driven Engineering e Model Engineering come sinonimi

Old one, MDA è il vecchio tipo, adesso c'è la MDE

estensione dell'approccio di analisi e progettazione Model-Driven Architecture (MDA) basato su metodologie object-oriented e sui linguaggi sviluppati dall'Object Management Group (UML, XMI, OCL, etc…) .

La Model-Driven Architecture si sofferma sullo sviluppo di sistemi ed applicazioni informatiche attraverso l'uso massiccio di modelli e trasformazioni tra modelli. Tale paradigma si riflette nella classificazione dei tipi di modelli in tre categorie: il sistema in esame viene modellato attraverso un *Platform Independent Model* (PIM) indipendente dalla piattaforma che provvederà alla sua implementazione e che si concentra esclusivamente sulla logica dell'applicazione; è presente inoltre un *Platform Description Model* (PDM) che descrive essenzialmente l'architettura di realizzazione. Attraverso l'uso di di una trasformazione automatica, il PIM ed il PDM vengono trasformati in un *Platform Specific Model* (PSM) di possibile realizzazione sulla piattaforma hardware/software descritta dal PDM. Esiste dunque una trasformazione per la quale $PSM = f(PIM,PDM) = f_{PDM}(PIM)$. I principi di automazione nelle trasformazioni tra modelli sono spesso referenziati come approcci "correct-by-construction" contrapposti ad alcune pratiche tradizionali denominate "construct-by-correction".

I principi comuni su cui fondano le discipline di Model Engineering sono:

- approccio meta-modellativo: ogni linguaggio di modellazione deve essere ben definito in un apposito meta-modello ed il linguaggio con il quale vengono descritti i linguaggi in un meta-meta-modello;
- definizione di trasformazioni tra modelli e delle metodologie per l'automatizzazione di tali trasformazioni;
- definizione di technological spaces ossia di domini applicativi specifici, caratteristici per la presenza di best practices, tecniche e strumenti ad hoc.

La maggior parte degli approcci di Model-Driven Engineering concordano sul fatto che le trasformazioni da PIM a PSM sono solo una parte delle possibili trasformazioni, ossia che i processi descritti nella MDA sono contenuti in quelli della Model-Driven Engineering. Vi sono ad esempio diversi campi di applicazioni di trasformazioni da PIM a PIM e da PSM a PSM. Occorre pertanto allargare lo spettro d'indagine. Anche una trasformazione può essere vista, secondo il paradigma del Model Engineering, come un modello conforme ad un particolare metamodello, rientrando pertanto in tutti i discorsi relativi ai modelli.

Come si evince dalla Figura 3, una trasformazione è dunque un modello M_T, conforme ad uno specifico metamodello F_T che lavora sui metamodelli sorgente (F_A) e destinazione (F_B). A tali metamodelli sono conformi i modelli rispettivamente M_A ed M_B sui quali agisce l'istanza della trasformazione M_T (chiamata in figura I_T).

Descrizione di Meta-meta-modelli e metamodelli come già viste per ATM

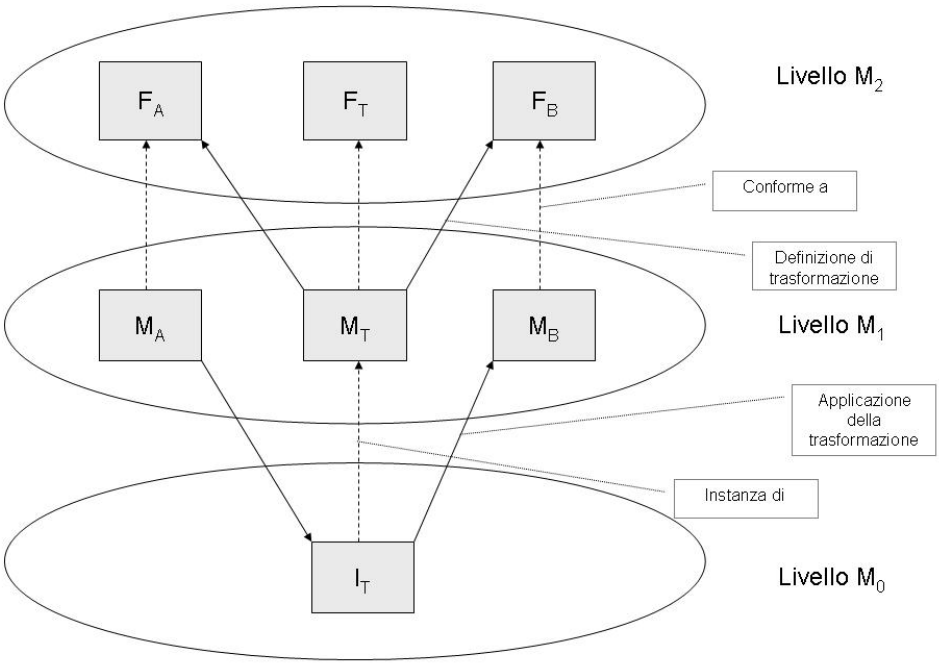

Figura 3: Schema di definizione di una trasformazione

Diversi approcci per la definizione di framework per Model Engineering sono stati proposti negli ultimi anni:

- Alanen et al. propongono processi di integrazione degli strumenti messi a punto dall'Object Management Group all'interno di discipline di Model Engineering basate su quattro distinti livelli di consapevolezza nell'uso di metodologie per la creazione e la manipolazione dei modelli;
- Bevezin et al. propongono approcci similari basati sulla nota "pila dei modelli (OMG)" (vedi Figura 1) dove sono espresse le relazioni tra i sistemi, i modelli ed i metamodelli. Questo punto di partenza viene sviluppato nel framework proposto, AMMA (Atlas Model Management Architecture), che si fonda sui seguenti cardini:
  - trasformazione tra modelli basata su linguaggi ben definiti;
  - model weaving ossia "l'ancoraggio" di più modelli in punti di contatto nell'ottica di generare modelli complessi a partire da modelli semplici;
  - associabilità di caratteristiche tra i vari modelli e metamodelli;
  - apertura dei modelli/metamodelli nei confronti di altri per lo scambio di informazioni.
- altri approcci sono stati presentati da diversi enti nell'ottica di colmare il distacco tra i principi teorici e gli aspetti pratici ed operativi della Model Engineering (IBM Eclipse Modeling Framework, Microsoft Software factories, …).

Diverse sono state le metodologie presentate per le trasformazioni. E' stato possibile per esse individuare alcuni fattori comuni: in particolare , e costituiscono alcuni dei tentativi di classificazione, formalizzazione ed unificazione di tali approcci.

Un possibile approccio è quello per cui è possibile classificare le trasformazioni tra modelli innanzitutto sulla base della natura del dominio e del condominio della trasformazione . Se F_A=F_B allora la trasformazione si dice *riformulazione* altrimenti si dice *traslazione*. Per quanto riguarda le prime, esse si dividono in *normalizzazione* (quando un modello è tradotto in un sotto linguaggio del sorgente), *rifattorizzazione* (un modello è ristrutturato in modo da migliorarne la qualità interna senza cambiare l'interfaccia dello stesso all'esterno), *correzione* (usato per eliminare gli errori) ed *adattamento* (usato per adattare il modello a nuove esigenze). I sottotipi di traslazione sono invece: *migrazione* (se F_A e F_B hanno lo stesso livello di astrazione), *sintesi*

Ad esempio classificando insieme tutte le trasf. effettuate con da o in UML, oppure PETRI-net

(se F_A è più astratto di F_B) e *reingegnerizzazione* (F_A è meno astratto di F_B).

Altro approccio alla classificazione delle trasformazioni è considerare la natura del formalismo di espressione delle stesse : ricordiamo tra essi l'approccio basato su trasformazioni di grafi come in , quello relazionale in cui vengono definite relazioni (vincoli) tra gli elementi dei metamodelli sorgente e destinazione , quello basato sulla marcatura di dei modelli come ad esempio l'approccio MDA .

Molte di queste metodologie hanno visto la costruzione di veri e propri ambienti di sviluppo per la definizione di trasformazioni tra modelli. QVT rappresenta certamente un riferimento col quale confrontarsi: esso rappresenta la confluenza di diversi contributi del mondo della ricerca in quello che è una proposta di standard OMG. Tale metodologia è relativa alla definizione di trasformazioni tra modelli nell'ambito del Technical Space definito dal Meta Object Facility (MOF) ossia del dominio dei modelli UML. Il suo approccio si basa sull'ibridazione tra quelli relazionali e quelli imperativi. Più in dettaglio lo standard prevede tre diversi livelli di linguaggio chiamati *Relations*, *Core* ed *Operational Mappings*. I primi due rappresentano la parte dichiarativa della metodologia mentre Opertional Mappings arricchisce il potere espressivo del linguaggio con costrutti imperativi.

Lo standard QVT definisce solo un framework di riferimento per la trasformazioni di modelli; esistono in letteratura diversi linguaggi e strumenti relativi all'implementazione di questa e di altre metodologie:

- ATL (ATLAS Trasformation Language): linguaggio sviluppato all'INRIA che permette di sviluppare trasformazioni tra modelli. ATL segue le indicazioni di QVT in quanto implementa sia costrutti dichiarativi che imperativi. Il cuore della parte dichiarativa è la *matched rule* che consiste nella specifica di un pattern sorgente sul modello sorgente che, una volta verificato, crea un pattern sul modello destinazione. La parte imperativa è costituita da *called rule* (costrutti simili alle procedure) e dai *block rule* (sequenze di istruzioni). L'esecuzione di una trasformazione specificata in ATL si appoggia ad una ATL Virtual Machine;
- VIATRA2 (VIsual Automated model TRAnsformations): è un approccio basato su trasformazione di grafi e sulla specifica matematica di tali trasformazioni in termini di Macchine Astratte a Stati (Abstract State Machine). Il concetto di base è il riconoscimento di pattern di grafi da parte del motore di esecuzione della trasformazione; in questo senso è più espressivo della metodologia di QVT ma non è conforme ad esso ;
- GReAT (Graphical Rewrite and Translation): anch'esso basato sulla trasformazione di grafi, ha un linguaggio di specifica della trasformazione distinto in tre parti :
  - linguaggio di specifica dei pattern;
  - linguaggio di trasformazione grafi, che si occupa di stabilire vincoli all'interno dei grafi sorgente e destinazione;
  - linguaggio controllo di flusso, che gestisce le modalità di esecuzione di una trasformazione tra modelli.
- IBM Rational Systems Developer: basato di QVT;
- Borland Together Architect 2006: anch'esso basato su QVT.

Al di là dei diversi approcci al problema della formalizzazione delle trasformazioni tra modelli spesso polarizzati dai domini specifici in cui tali metodologie nascono, individuiamo in essi alcuni elementi comuni. Tali elementi di unione vanno ricercati nell'uso combinato di *precondizioni* e di *conseguenze*. Le precondizioni esprimono la validità logica del modello sorgente per l'applicabilità della regola mentre la conseguenza rappresenta la postcondizione che è necessario avere sul modello di destinazione dopo l'applicazione della regola stessa. In d'altronde, tra le proprietà relative alle trasformazioni è menzionata proprio la specifica di condizioni LHS/RHS.[11]

[11] LHS/RHS è l'acronimo di Left Hand Side/Right Hand Side (sorgente/destinazione)

A colmare il divario tra la teoria della metamodellazione e la pratica dell'ingegneria dei processi di sviluppo software c'è il concetto di Technological Spaces . Un *Technological Space* (TS) è infatti un dominio "tecnologico" dove i modelli ed i formalismi in esso definiti sono omogenei o comunque riconducibili allo stesso insieme di best practices (metodologie consolidate, processi, strumenti, etc…). Essi sono rappresentati dai meta-meta-modelli che sono in cima alla pila dei modelli; alcuni esempi di domini tecnologici sono: MDA, i linguaggi di programmazione (il cosiddetto *Grammarware*) oppure il dominio dei DBMS. A titolo di esempio, in Tabella 1 è definita la corrispondenza tra i livelli astratti della pila deli modelli ed i concetti relativi al Grammarware.

Tabella 1: Corrispondenza software-pila dei modelli

Livelli	**TS Grammarware**
Metametamodello	EBNF
Metamodello	Grammatica del linguaggio C
Modello	Programma C specifico
Sistema	Specifica esecuzione di uno specifico programma

Contributo originale

Il contributo originale apportato da questa tesi è relativo alla formalizzazione di un approccio Model Enginnering per l'analisi di sistemi complessi e critici. In particolare si farà riferimento al framework OsMoSys e si definiranno in esso gli elementi fondamentali di un approccio di Model Engineering; tale formalizzazione consiste nella definizione delle seguenti caratteristiche:

- technological space per OsMoSys con la definizione dei livelli e la mappatura tra tali livelli e quelli relativi alla metodologia MDE;
- inquadramento della definizione formale dei linguaggi di OsMoSys nel contesto del Model Engineering;
- framework teorico per la trasformazione e composizione di modelli.

Successivamente verrà data una classificazione di diversi operatori di trasformazione applicabili ai diversi tipi di modelli di OsMoSys. Attraverso tale classificazione e formalizzazione è possibile inquadrare alcuni esempi di trasformazioni notevoli di forte interesse nell'analisi di sistemi complessi (multiformalismo, composizionalità, etc…). Per tali casi, oltre alla formalizzazione, verranno forniti esempi applicati ai metodi formali basati su grafo (Reti di Petri, Alberi di Guasti, Reti Bayesiane).

Altro contributo originale della tesi è contenuto nella definizione di una metodologia di modellazione ad oggetti e composizionale basata su reti di Petri ad alto livello (Stochastic Well-formed Nets) e paradigmi di comunicazione client-server.

Quanto introdotto in questa tesi è applicato allo studio di un sistema di segnalamento ferroviario corrispondente all'architettura di riferimento dello standard ERTMS/ETCS che costituisce un caso di sistema critico e complesso. Per tale sistema verrà descritta un'analisi multiformale e multirisoluzione.

La tesi è strutturata nel seguente modo: il Capitolo I inquadra il contesto in cui questo lavoro di ricerca si svolge evidenziando le problematiche e descrivendo lo scopo del lavoro. Nel Capitolo II viene definito il Technological Space di OsMoSys e vengono definite le trasformazioni tra modelli con la successiva classificazione delle trasformazioni stesse e della sottoclasse delle composizioni. Il Capitolo III formalizza i concetti di traslazione, riformulazione e decomposizione tra modelli; inoltre viene descritto il caso del multiformalismo implicito come specializzazione del concetto di traslazione. Il Capitolo IV formalizza il concetto di composizione di modelli e di operatore di composizionalità, definendo per questi ultimi una possibile tassonomia sulla base dei paradigmi di interazione tra modelli; sono inoltre formalizzati alcuni operatori utilizzati nelle metodologie di modellazione multiformale di sistemi. Il Capitolo

V si concentra sulla categoria di operatori di composizione legata a paradigmi di interazione basati su scambio di messaggi: in tale contesto, viene descritta un'interpretazione di tali operatori nell'ambito di modelli ad oggetti basati su reti di Petri ad alto livello (SWN) e viene fornita in tale contesto l'applicazione all'analisi di un sistema di Contact Center. Il Capitolo VI descrive il caso di studio industriale del sistema di segnalamento ferroviario ERTMS/ETCS fornendo per esso l'applicazione di una metodologia di modellazione ed analisi multiformale e composizionale. Completa la tesi l'Appendice A relativa ad una breve digressione sui benefici dell'uso dei metodi formali in ambito industriale.

Capitolo II:

Model Engineering per sistemi critici e complessi

In questo capitolo verrà definito il processo di formalizzazione dei linguaggi e delle trasformazioni di OsMoSys secondo il paradigma del Model Engineering. Dopo aver inquadrato i linguaggi di OsMoSys secondo tale approccio e dopo aver definito quindi un Technological Space per tale framework, si definirà un approccio formale alle trasformazioni tra modelli in OsMoSys fornendo una classificazione di tali operazioni.

Un approccio Model Engineering per OsMoSys

L'obiettivo è quello di inquadrare i principi ed i concetti dei formalismi di OMM all'interno della Model Driven Engineering; inoltre si svilupperà all'interno del framework per la risoluzione dei modelli (OMF) una formalizzazione dei meccanismi di trasformazione di modelli.
Verranno seguiti a tale fine i seguenti passi:

- definizione di un Technological Space attraverso la formalizzazione dell'intera pila dei linguaggi di OsMoSys (formalizzazione dei livelli M_3, M_2 ed M_1). Questo passo verrà realizzato riprendendo i concetti espressi in e riformulando gli stessi nell'ottica di un approccio orientato al Model Engineering;
- classificazione dei tipi di modelli possibili in OsMoSys (vedi);
- definizione di trasformazione tra modelli in OsMoSys e caratterizzazione delle loro proprietà;
- classificazione delle trasformazioni di OsMoSys;
- definizione di composizione tra modelli: formalizzazione del concetto di composizione tra modelli e classificazione degli operatori di composizione; formalizzazione delle loro proprietà.

Il dominio tecnologico

La definizione di un *Technological Space* passa attraverso quella della pila dei modelli e quelle dei modelli stessi. Con riferimento a , definiamo in Tabella 2 le relazioni tra i livelli della pila dei modelli in ambito OMG e la definizione degli stessi in OsMoSys.

Tabella 2: La "pila dei modelli" in OsMoSys

MDE	OsMoSys
Meta-meta-modello	Metaformalismo (Meta formalism)
Meta-modello	Metaclasse (Model MetaClass)
Modello	Classe (Model Class)
Sistema	Oggetto (Model Object)

Adotteremo nella nostra formalizzazione la seguente notazione per l'espressione delle seguenti relazioni:

μ (*rappresenta*): m μ S indica che il modello m rappresenta il sistema S;
χ (*conforme a*): m χ F indica che il modello m è conforme al formalismo F.

Descriviamo adesso, sempre in relazione a quanto formalizzato in , i vari livelli della pila di OsMoSys.

Meta-meta-modello (Metaformalismo)
Il meta-meta-modello sul quale si basa OsMoSys è quello dei formalismi basati su grafo. Notiamo con F un generico formalismo e sia ε_F l'insieme dei tipi degli elementi di F. Sia inoltre definito $\forall$ e $\in$ ε_F l'insieme P_e come l'insieme unione delle proprietà caratteristiche del tipo di elemento e del suo nome. A titolo di esempio per un formalismo quale le reti di Petri, l'insieme dei tipi di elementi è pari a ε_F={posto, transizione, arco}; in base a ciò avremo che: P_{posto} = {"Posto", numero di token} e che P_{arco}={"Arco", cardinalità}.
In OsMoSys è definita anche la relazione di "*is-a*" tra formalismi al fine di definire gerarchie di linguaggi:

$$D \text{ eredita da } B \Leftrightarrow \varepsilon_B \subset \varepsilon_D$$

Meta-modello (Metaclasse)
Dato F un formalismo, definiamo un modello (model class):
MC_F = (N,S,SM) tale che:
- MC_F χ F;
- N: singleton rappresentante il nome del modello;
- S: insieme rappresentante la struttura della classe. Esso contiene elementi i cui tipi possono appartenere ad SM o a ε_F. Più formalmente: $\forall$ e $\in$ S, Type(e) $\in$ ε_F U SM;
- SM: insieme dei sottomodelli contenuti in MC_F.

La struttura del model class S può essere partizionato in due insiemi:

$$S = External_S \cup Internal_S$$

dove $External_S$ definisce l'interfaccia, ossia l'insieme di elementi visibili all'esterno mentre $Internal_S$ definisce gli elementi non visibili esternamente della model class.
Inoltre, definito l'insieme P_S come l'insieme delle proprietà di tutti gli elementi di S, esso può essere partizionato in:

$$P_S = EP_S \cup IP_S$$

Dove EP_S è l'insieme delle proprietà del modello modificabili nel loro valore per ogni istanza della model class mentre IP_S sono le proprietà non modificabili.

Modello (Model Class)
Data la definizione di model class, ne è possibile una loro classificazione in diversi tipi .
Definiamo innanzitutto la classe dei *Flat Model Class*. A questa classe appartengono tutti i model class tali che SM = $\varnothing$, ossia i modelli "omogenei", costituiti cioè da un solo linguaggio. Un semplice esempio più essere quello di una rete di Petri.
Definiamo adesso la classe dei *Composed Single Formalism Model Class*. A tale classe appartengono i modelli contenenti sottomodelli conformi allo stesso formalismo cui il modello principale è conforme. Più formalmente supponiamo FF = {F, F_1, F_2, …, F_n} insieme di formalismi e MC_F χ F tale che SM={M_1, M_2, …,M_n} con M_i χ F_i. MC_F è un Composed Single Formalism Model Class se e solo se $\forall$ i $\in$ {1,..,n} F_i=F

Supponiamo come sopra che FF = {F, F_1, F_2, …, F_n} sia l'insieme dei formalismi e MC_F χ F tale che SM = {M_1, M_2, …,M_n} con M_i χ F_i. MC_F è un *Composed Explicit Multiformalism Model Class* se e solo se ∃ i ∈ {1,..,n}: $F_i \neq$ F. Questa classe corrisponde a quella dei modelli che fanno un uso esplicito del multiformalismo.
Definiamo infine la categoria dei *Composed Implicit Multiformalism Model Class*. Questa classe corrisponde a quella dei modelli che fanno un uso implicito del multiformalismo attraverso la definizione di Implicit Model Class. Consideriamo la model class $MC_{F'}$ χ (F′, ε′), essa è una Implicit Model Class se e solo se ∃ (F′, ε′) : F′ eredita da F e ε′= ε ∪$IN_{F'}$ ∪ $IE_{F'}$. $IN_{F'}$ è l'insieme dei nodi impliciti ed $IE_{F'}$ quello degli archi impliciti tale che uno dei due nodi connessi da uno dei suoi elementi deve appartenere ad $IN_{F'}$. Si ha che $MC_{F'}$ è un Composed Implicit Multiformalism Model Class se e solo se $MC_{F'}$ è un Implicit Model Class ed ∃ c ∈ $IN_{F'}$: a c è associato un modello MC_L con L≠ F.

Una visione d'insieme
Attraverso la Figura 4 diamo una visione d'insieme del Technological Space per OsMoSys. Notiamo che si è usata per notare la relazione di "is-a" tra modelli o formalismi la notazione:
ε (*is-a*): m ε n indica che il modello/formalismo m "è-un" modello/formalismo n.

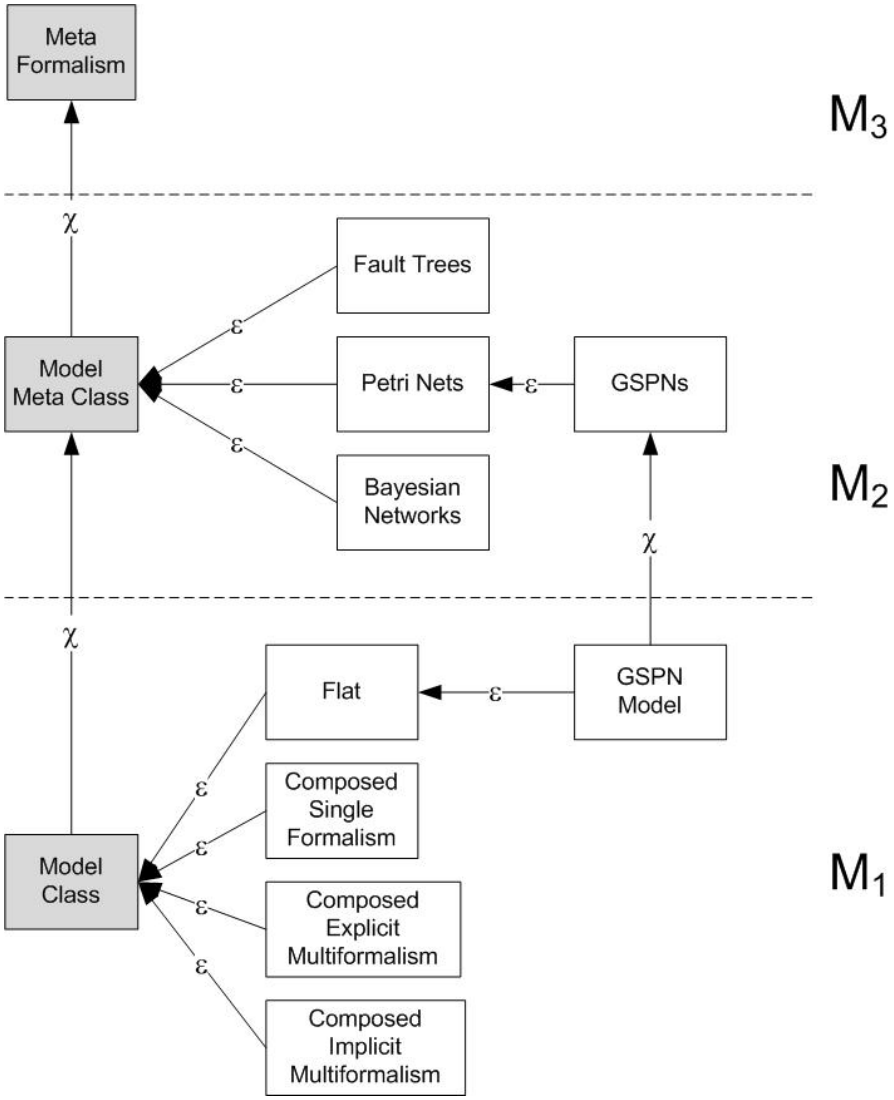

Figura 4: Technological Space per OsMoSys

La definizione di un Technological Space non preclude però all'interoperabilità tra modelli e formalismi afferenti a meta-meta-modelli differenti. Infatti come evidenziato in Aksit et al. ,

esiste la possibilità nonché l'opportunità nel costruire bridge tra i Technological Space. Un semplice esempio della convenienza di tali strutture è illustrata in Figura 5. Si vuole passare da un programma espresso in un linguaggio di programmazione A ad un altro programma espresso in B. Tale operazione può essere ad esempio effettuata attraverso una trasformazione diretta "intra-TS" dal linguaggio A a quello B. Un'alternativa è quella di definire tale trasformazione nell'ambito del bridging tra il Grammarware (Technological Space dei linguaggi di programmazione) e l'ambito della MDA (attraverso ad esempio il formalismo UML) componendo in tal modo una trasformazione di "reverse-engineering" dal linguaggio A ad UML con una di "code-generation" da UML a B.

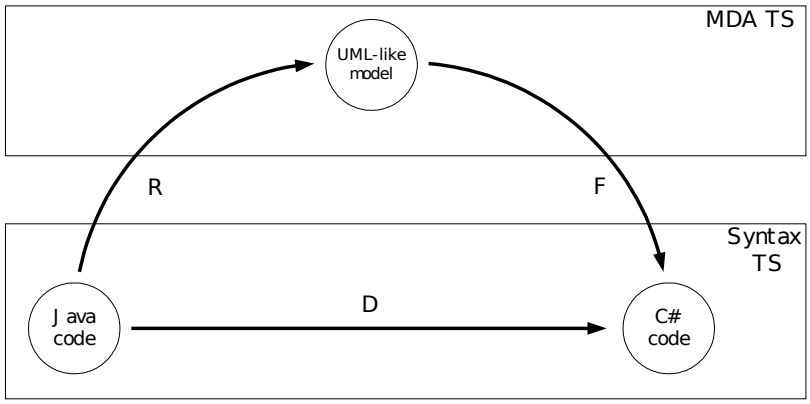

Figura 5: Esempio di bridging tra Technological Space (da)

Diversi approcci sono presenti in letteratura nell'ambito della definizione di diversi bridge tra formalismi: ricordiamo Staikopoulos et al. con le trasformazioni dal formalismo Business Process Execution Language (BPEL) a Petri Nets e Florescu et al. con la costruzione di bridge tra il dominio XML e quello dei database relazionali.
Quello di OsMoSys d'altro canto, per la natura generale del suo meta-meta-modello, ben si presta a diventare un Technological Space capace di facilitare la definizione di bridge verso gli altri TS. Questo aspetto dell'inquadramento di OsMoSys all'interno delle discipline del Model Engineering non verrà trattato in questa tesi costituendo spunto per ulteriori approfondimenti in merito.

Trasformazioni tra modelli

Introdurremo adesso una formalizzazione del concetto di trasformazione tra modelli. Per trasformazione tra modelli intendiamo una funzione che, dato un modello m_A conforme al metamodello F_A, generi un nuovo modello m_B conforme al metamodello F_B sotto il vincolo di conservazione della proprietà di rappresentazione del sistema da parte dei modelli.
Definiamo, dato un formalismo F, MM_F l'insieme dei modelli possibili conformi a tale formalismo ossia $MM_F = \{m \in M_1 : m \chi F\}$[12]. Definiamo una trasformazione tra i modelli rispetto al sistema S, la funzione

$$T : m_S \in MM_{F_S} \rightarrow m_D \in MM_{F_D}$$

soggetta al vincolo:

$$m_S \ \mu S \Rightarrow m_D \ \mu S$$

dove m_S e F_S sono detti modello e formalismo sorgente e m_D e F_D modello e formalismo destinazione. Il vincolo cui è soggetta la funzione garantisce che il modello destinazione

[12] Ricordiamo che M_1 è l'insieme dei possibili modelli conformi a qualsiasi formalismo.

rappresenti ancora il sistema in esame. In questa tesi tale vincolo verrà inteso sempre presente nella definizione delle trasformazioni, tranne ove espressamente specificato.
Consideriamo il caso generale di trasformazioni k ad n. Estendiamo prima il concetto di rappresentazione di sistema ad una t-pla di modelli. Considerato l'insieme di formalismi FF = $\{F_1,..,F_t\}$ e considerati i modelli m_i con $i \in \{1,..,t\}$ tali che $m_i \chi F_i \ \forall\ i \in \{1,..,t\}$, diciamo che:

$$\overline{m} = (m_1,\ldots,m_t) \in MM_{F_1} \times \cdots \times MM_{F_t},\ \overline{m}\ \mu\ S \Leftrightarrow \forall i \in \{1,\ldots,t\}\ m_i\ \mu\ S$$

In tal caso ogni m_i rappresenta una "vista" diversa del sistema nel senso che rappresenta un aspetto diverso del sistema stesso. Un esempio chiarificatore ci viene dalla modellazione di un software attraverso il linguaggio UML dove diversi diagrammi modellano aspetti diversi del sistema in esame (strutturali, di interazione, evoluzione interna, di deploy, etc…).
Date queste premesse definiamo trasformazione di k in m modelli rispetto al sistema S, la funzione:

$$T : \overline{m_S} = \left(m_S^1,\ldots,m_S^k\right) \in MM_{F_S^1} \times \cdots \times MM_{F_S^k} \rightarrow$$
$$\overline{m_D} = \left(m_D^1,\ldots,m_D^m\right) \in MM_{F_D^1} \times \cdots \times MM_{F_D^m}$$

soggetta al vincolo:

$$\overline{m_S}\ \mu S \Rightarrow \overline{m_D}\ \mu S$$

La formalizzazione di una trasformazione avverrà spesso attraverso la specifica di precondizioni (*condizioni*) sul modello sorgente e delle postcondizioni (*azioni*) su quello destinazione come precedentemente indicato. Sarà tenuta però una notazione non orientata verso nessuna delle diverse soluzioni presenti in letteratura ma puramente basata su proprietà matematiche dei casi sotto esame. Possiamo pertanto dire che essa è esprimibile attraverso un insieme di coppie di precondizioni e postcondizioni:

$$T = \{(Cond, Act)_1,\ldots,(Cond, Act)_n\}$$

dove per ogni coppia, *Act* viene attivata se e solo se la proposizione *Cond* omologa viene soddisfatta. Aggiungiamo per quanto riguarda la notazione l'assunzione per convenzione di: $m_S = (N_S, S_S, SM_S)$ e $m_D = (N_D, S_D, SM_D)$. Di conseguenza a quanto appena fissato vale che:

$$S_D = External_S^D \cup Internal_S^D$$

e che:

$$P_S^D = EP_S^D \cup IP_S^D$$

Possiamo procedere in modo del tutto analogo per S_S.

Operatori di trasformazione

Diamo adesso una classificazione delle trasformazioni tra modelli in OsMoSys. Una prima classificazione ci viene dalla cardinalità della trasformazione: possiamo avere pertanto trasformazioni uno-uno, uno-molti e molti-uno.
Viene inoltre proposta, sulla base del tipo di trasformazione, la seguente classificazione:

- **riformulazione**: in accordo con quanto detto in , viene definita riformulazione una trasformazione uno-uno da un modello in un altro quando i entrambi i modelli sono conformi allo stesso formalismo. Possiamo avere a tale riguardo due tipi di riformulazione: *riduzione*, che occorre quando eliminiamo degli elementi dalla struttura di un modello e la *normalizzazione*, quando ossia esprimiamo un modello limitando il formalismo cui esso è conforme ad un suo sottoinsieme;
- **traslazione**: come in , viene definita traslazione una trasformazione uno-uno quando i formalismi sorgente e destinazione sono diversi. Tale trasformazione può essere intra-gerarchica se i due formalismi sono uno erede, anche indiretto, dell'altro o extra-gerarchica, se i due formalismi non sono collegati da vincoli di ereditarietà;
- **decomposizione**: trasformazione uno-molti caratterizzata dalla suddivisione di un modello in più modelli dello stesso formalismo al fine di separare aspetti particolari del modello stesso. A seconda della sovrapposizione tra i modelli destinazione, possiamo distinguere la sottocategoria delle *partizioni*;
- **composizione**: trasformazione molti-uno caratterizzata dall'accorpamento di diversi modelli sorgenti in un modello più largo secondo un determinato paradigma di composizione.

Descriveremo in dettaglio tali operazioni nei prossimi capitoli fornendo per essi un esempio inerente ai metodi formali.

Capitolo III:

Operatori di Trasformazione

In questo capitolo definiremo le varie tipologie di riformulazione e di traslazione di modelli definendo per ciascun tipo le loro proprietà formali in OsMoSys. Si focalizzerà l'attenzione sull'interpretazione del multiformalismo implicito come traslazione tra modelli portando come esempio il caso del formalismo dei Repairable Fault Trees. Verrà infine discusso l'operatore di decomposizione di modelli.

Riformulazione

Definiamo più formalmente il concetto di riformulazione. Data una trasformazione rispetto al sistema S,

$$T : m_S \in MM_{F_S} \rightarrow m_D \in MM_{F_D}$$

soggetta al vincolo di conservazione della proprietà di rappresentazione del sistema, essa si dice *riformulazione* di m_S in m_D rispetto ad S se e solo se $F_S = F_D = F$ e quindi se T è una endofunzione in quanto:

$$F_S = F_D \Rightarrow MM_{F_S} = MM_{F_D}$$

Distinguiamo due casi notevoli.

Riduzione
Si parla di riduzione quando il modello viene semplificato nella sua struttura. Quindi, considerate due model class m_S e m_D, una riformulazione T si dice riduzione se e solo se

$$S_D \subset S_S$$

La struttura infatti rappresenta l'insieme delle istanze dei tipi di elementi contemplati dal formalismo e pertanto una riduzione consiste nell'eliminazione di alcune di queste istanze dal modello sorgente. Si noti che non è detto che le proprietà del sottoinsieme della struttura di m_S presente anche in m_D restino conservate. Può esistere cioè un elemento di S_D tale che alcune proprietà dell'elemento siano diverse dall'analogo elemento di S_S.
Dato dunque che:

$$T = \{(Cond, Act)_1, \ldots, (Cond, Act)_n\}$$

vale pertanto che una generica postcondizione della trasformazione (*Act*) è composta da una funzione operante sulla struttura (Σ) ed una sui valori delle proprietà (Φ):

$$Act = (\Sigma, \Phi)$$
$$\Sigma : S_S \rightarrow S_D$$
$$\Phi : P_S^S \rightarrow P_S^D$$

Vale dunque la seguente:

$$\exists S_{RID} \subset S_S : \Sigma(s) = \begin{cases} \varnothing & s \in S_{RID} \\ s & s \notin S_{RID} \end{cases}$$

Ulteriore sottoinsieme delle riduzioni è costituito dalle rifattorizzazioni che si contraddistinguono dal fatto di lasciare inalterata l'interfaccia di un modello e di alterare solo la sua descrizione interna. Una riformulazione T è dunque una rifattorizzazione se e solo se è una riduzione ed avviene che:

$$\left(External_S^D = External_S^S \wedge S_D \subset S_S\right) \Rightarrow Internal_S^D \subset External_S^S$$

Un esempio di riduzione ci viene dalle regole di sintesi delle reti di Petri. Come riferito in , sono presenti il letteratura diversi approcci alla riduzione di Reti di Petri attraverso l'applicazione di regole di sintesi. Consideriamo a titolo di esempio la rete di Petri P/T illustrata in Figura 6.

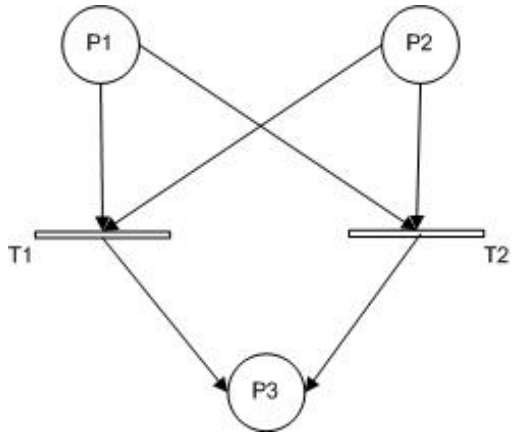

Figura 6: Applicazione della riduzione alle Petri Nets (modello sorgente)

Tale esempio rappresenta la riduzione di una porzione di rete di Petri dove siano presenti più transizioni collegate agli stessi posti. Più formalmente quindi la riduzione è applicabile se:[13]

$$\exists T_{EQ} = \{t_1, \ldots, t_n\} \subseteq T_S : \forall t_i, t_j \in T_{EQ} \quad t_i^\bullet = t_j^\bullet \wedge {}^\bullet t_i = {}^\bullet t_j$$

dove T_S è l'insieme delle transizioni del modello sorgente e T_D quello del modello destinazione. In tal caso l'applicazione della riduzione trasforma il modello sorgente in quello destinazione raffigurato in Figura 7. In termini matematici la trasformazione produce il seguente effetto:

$$\exists\, t_k \in T_{EQ} : t_k \in T_D \wedge \forall\, t_i \in T_{EQ} - \{t_k\},\, t_i \notin T_D$$

[13] Si presume che il lettore conosca il formalismo delle reti di Petri .

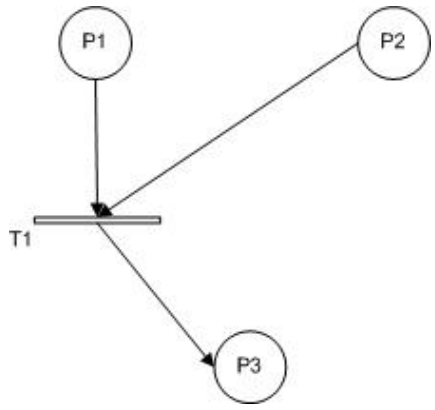

Figura 7: Applicazione della riduzione alle Petri Nets (modello destinazione)

Specializziamo dunque la caratterizzazione della nostra trasformazione T = (*Cond*,*Act*) a questo esempio. Per *Cond* avremo che: [14]

$$Cond : P(T_S) \rightarrow \{true,\ false\} \mid Cond(T_{EQ}) = \begin{cases} true & se\ \forall t_i, t_j \in T_{EQ} : \quad t_i^{\bullet} = t_j^{\bullet} \wedge {}^{\bullet}t_i = {}^{\bullet}t_j \\ false & altrimenti \end{cases}$$

Notiamo che per una model class conforme al linguaggio delle reti di Petri P/T vale che S = (P,T,A) rispettivamente insieme dei posti, delle transizioni e degli archi della rete; *Act* sarà invece descritta da:[15]

$$Act = (\Sigma, \Phi)$$

$$\Sigma(s) = \begin{cases} \bar{s} & se\ s \in T_{EQ} : Cond(T_{EQ}) = true \wedge \bar{s} \in T_{EQ} \\ \varnothing & se\ s \in A_{EQ} : s = (p,t) \vee (t,p) : t \in T_{EQ} - \{\bar{s}\} \\ s & altimenti \end{cases}$$

$$\Phi : P_S^S \rightarrow P_S^D \mid \Phi = 1$$

Possiamo trovare nell'ambito delle reti di Petri un esempio che mostri l'alterazione dell'insieme delle proprietà di uno o più elementi del modello destinazione. Sempre con riferimento alla Figura 6, supponiamo che la rete di Petri in esame sia conforme al formalismo delle Generalized Stochastic Petri Nets (GSPN) e che le transizioni T_1 e T_2 siano transizioni temporizzate stocastiche con parametri di firing rate pari rispettivamente a λ_1 ed λ_2. In tal caso, essendo la regola "strutturale" rispetto alle reti di Petri, essa sarà applicabile ma, affinché possa valere il vincolo di conservazione della proprietà di rappresentazione del sistema, deve valere che:[16]

$$(\exists T_{EQ} = \{t_1, \ldots, t_n\} \subseteq T_S : \forall t_i, t_j \in T_{EQ} \quad t_i^{\bullet} = t_j^{\bullet} \wedge {}^{\bullet}t_i = {}^{\bullet}t_j)$$

$$\Downarrow$$

$$(\exists t_k \in T_{EQ} : t_k \in T_D \wedge \forall t_i \in T_{EQ} - \{t_k\}, t_i \notin T_D) \wedge \left(\lambda_k^D = \sum_{i=1}^{n} \lambda_k^S \right)$$

Per l'esempio in Figura 7 vale che $\lambda_1^D = \lambda_1^S + \lambda_2^S$.

Questo ulteriore caso richiede nella formalizzazione della trasformazione, un'opportuna modifica alla funzione Φ.

[14] Dato l'insieme X notiamo con P(X) l'insieme dei sottoinsiemi di X.

[15] Notiamo con 1 la funzione identità.

[16] La generalizzazione della regola alle GSPN è giustificata dalla formula del calcolo di permanenza medio in una marcatura .

Normalizzazione

Si parla di normalizzazione quando una riformulazione di modello restringe il formalismo in cui il modello sorgente è espresso ad un sottoinsieme dello stesso. In altre parole ciò corrisponde ad esprimere il modello, che comunque conserva la conformità al linguaggio, in un sottoinsieme dello stesso formalismo. Dato ε'_F il sottoinsieme di ε_F al quale vogliamo restringere il modello sorgente, possiamo scrivere che:

$$\forall\, e \in \varepsilon_F - \varepsilon'_F, \exists\ e' \in \varepsilon'_F : \forall s \in S_S : Type(s) = e, \exists s' \in S_D : Type(s') = e$$

Ciò vuol dire che, per tutti gli elementi della struttura S tali da avere come tipo un elemento del sottolinguaggio che si vuole eliminare, si deve provvedere ad un'opportuna sostituzione con altri elementi di tipi diversi:

$$Cond : S_S \to \{true,\ false\} \mid Cond(s) = \begin{cases} true & se\ Type(s) \in \varepsilon_F - \varepsilon'_F \\ false & se\ Type(s) \in \varepsilon'_F \end{cases}$$

$$Act = (\Sigma)$$

$$\Sigma : S_S \to P(S_D) \mid \Sigma(s) = \begin{cases} S'_D & se\ Cond(s) = true, Type(S'_D) \subseteq \varepsilon'_F \\ s & se\ Cond(s) = false \end{cases}$$

dove S'_D è un nuovo sottoinsieme di elementi generato in S_D in sostituzione degli elementi non appartenenti a ε'_F.

Un esempio di applicazione di tale trasformazione ci viene ancora dalle reti di Petri. Nel corso degli anni molte estensioni sono state fatte alle reti di Petri al fine di potenziarne l'espressività, di poter cioè esprimere comportamenti sempre più complessi in modo sempre più conciso. In vengono menzionate diverse di queste estensioni tra cui scegliamo quella dell'arco pulitore (*clean arc* o *reset arc*). Tale arco, orientato da un posto ad una transizione (vedi Figura 8), ha come compito quello di ripulire il posto da tutti i token ogni qualvolta la transizione scatti.

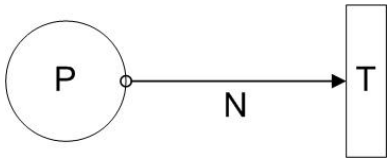

Figura 8: Clean Arc

Riferendoci alla Figura 8, la caratterizzazione di un tale modello in termini di abilitazione della transizione e di regola di sparo è:

$$\begin{cases} T\ è\ abilitata \Leftrightarrow M(P) \geq n \\ T\ scatta \Rightarrow M'(P) = 0 \end{cases}$$

Riduciamo tale costrutto al sottolinguaggio (rispetto a questo esempio) delle reti di Petri con archi inibitori. In tale contesto possiamo ricorrere a quello che in letteratura è definito come un pattern per reti di Petri *Complete Removal* descritto in ed illustrato in Figura 9.

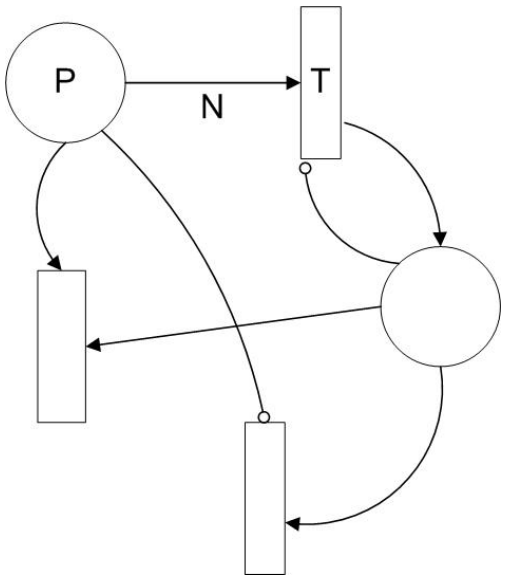

Figura 9: Complete Removal Pattern

Nei termini della nostra normalizzazione otteniamo dunque che ε'_F = {clear arc} e che S'_D è la struttura descritta dal pattern.

Traslazione

Si chiama traslazione una trasformazione uno-uno rispetto al sistema S:

$$T : m_S \in MM_{F_S} \rightarrow m_D \in MM_{F_D}$$

soggetta al vincolo di conservazione della proprietà di rappresentazione del sistema e tale che $F_S \neq F_D$. Possiamo distinguere tra traslazioni *intra-gerarchiche* ed *extra-gerarchiche*.
Le prime avvengono tra due modelli conformi a formalismi di cui uno eredita, anche in via indiretta, dall'altro. Un esempio a tal riguardo può essere quello della trasformazione di un modello a reti di Petri con archi multipli in un modello P/T. Molte traslazioni intra-gerarchiche possono essere viste come normalizzazioni di linguaggio dalla definizione di ereditarietà dei formalismi. Dato che l'eredità in OsMoSys è intesa sia nel senso di arricchimento dell'insieme ε_F con nuovi elementi che in termini di ampliamento delle proprietà di un elemento di tale insieme, dobbiamo fare un'ulteriore distinzione. Qualora i due formalismi sorgente e destinazione differiscano esclusivamente di nuovi elementi, ossia se:

$$\varepsilon = \varepsilon_{F_S} - \varepsilon_{F_D} : \forall e \in \varepsilon, \forall es \in \varepsilon_{F_S}, e = es \Rightarrow P_e = P_{es}$$

la traslazione intra-gerarchica può essere vista come una normalizzazione, altrimenti deve essere trattata diversamente con la definizione di una funzione Φopportuna. Nel caso sopra detto, per la trasformazione di un modello da formalismo PN ad archi multipli a modello P/T sarà necessario definire una funzione Φ che trasformi in uno tutte le molteplicità degli archi maggiori di uno.
Concentriamoci adesso sulle riformulazioni extra-gerarchiche. Definiamo per semplicità di notazione:

$$\varepsilon = \varepsilon_{F_S}, \varepsilon' = \varepsilon_{F_D}$$

avremo che:

$$\forall\, e \in \varepsilon\ \exists\, E' \subseteq \varepsilon' : \forall s \in S_S : Type(s) = e, \exists\ S'_D : \forall s' \in S'_D\ Type(s') \in E' \wedge S'_D \subseteq S_D$$

Illustriamo ad esempio il caso della trasformazione di un modello di alberi di guasti in una rete Bayesiana. In sono definite alcune regole per la trasformazione di un fault tree in una bayesian network (vedi Tabella 3):

Tabella 3: Corrispondenza Fault Tree - Bayesian Network

Fault Tree	Bayesian Network
Basic Event	Node
(Gate, Output Event)	Node
Arc	Arc
OR Gate	CPT con OR logico
AND Gate	CPT con AND logico

Notiamo che tutti i nodi possono assumere come possibili valori solo *true* e *false*.
Consideriamo un semplice Fault Tree come illustrato in Figura 10.

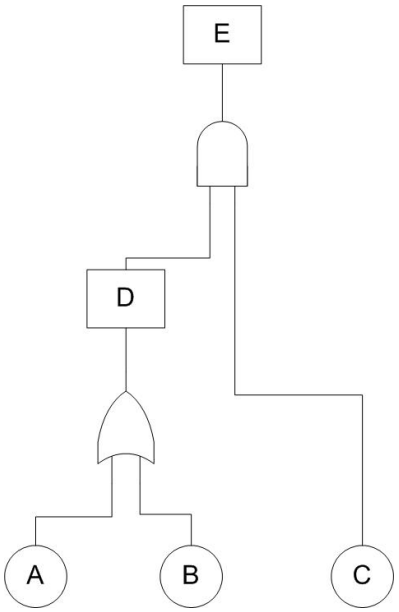

Figura 10: Esempio di traslazione (fault tree sorgente)

Applicando le regole di trasformazione presenti in , otteniamo la rete bayesiana illustrata in Figura 11.

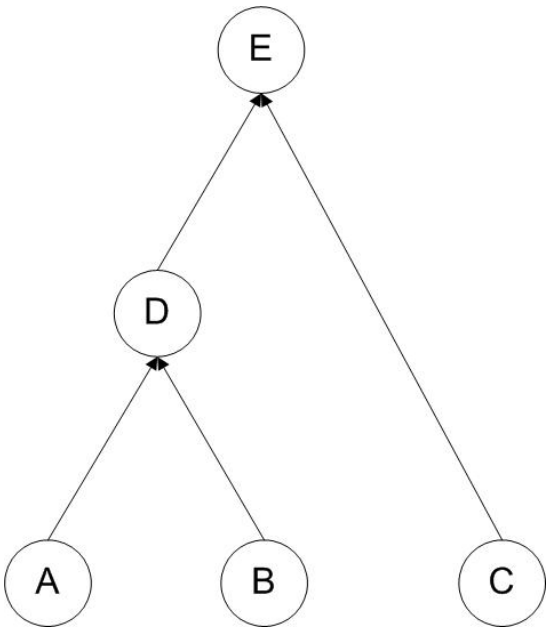

Figura 11: Esempio di traslazione (bayesian network destinazione)

Le tabelle di probabilità condizionata (Conditional Probability Table – CPT) descrivono come le probabilità degli ingressi si combinano e determinano la distribuzione di probabilità del nodo stesso. Le tabelle relative all'esempio considerato sono successivamente indicate: la Tabella 4 indica la struttura della CPT dei nodi A, B e C dove λ è il rate associato al basic event nel fault tree; la Tabella 5 indica la CPT del nodo D (OR logico); la Tabella 6 descrive la CPT del nodo E (AND logico).

Tabella 4: CPT basic event

True	*False*
λ	1- λ

Tabella 5: CPT nodo OR logico

A	B	True	False
True	True	1.0	0.0
True	False	1.0	0.0
False	True	1.0	0.0
False	False	0.0	1.0

Tabella 6: CPT nodo AND logico

D	C	True	False
True	True	1.0	0.0
True	False	0.0	1.0
False	True	0.0	1.0
False	False	0.0	1.0

Definiamo formalmente la trasformazione. Siano F_{FT} e F_{BN} i formalismi dei Fault Trees e delle Bayesian Networks[17]. Siano dunque considerati gli insiemi dei tipi di elementi del linguaggio:

$$\varepsilon_{FT} = \{basic\ event, OR\ gate, AND\ gate, event, arc\}$$
$$\varepsilon_{BN} = \{node, arc\}$$

[17] Non verranno considerati tutti gli elementi possibili dei due formalismi. Ci si limiterà, in questa sede, a descrivere questa riformulazione relativamente agli elementi dei linguaggi inerenti ai modelli considerati.

e le proprietà di alcuni di questi tipi:

$$P_{basic\ event} = \{nome, \lambda\}$$
$$P_{node} = \{nome, cpt\}$$
$$P_{arc} = \{sorg, dest\}$$

quest'ultima tale che:

$$P_{arc}(sorg) \in S$$
$$P_{arc}(dest) \in S$$

La trasformazione è dunque caratterizzata da

$$Act = (\Sigma, \Phi)$$
$$\Sigma: S_{FT} \rightarrow S_{BN}$$
$$\Sigma(s) = s': \begin{cases} Type(s') = arc & se\ Type(s) = arc \wedge Type(P_s(dest)) \neq event \\ Type(s') = node & altrimenti \end{cases}$$

Sia inoltre valida la seguente proprietà:

$$\forall s \in S_{FT} : Type(s) = arc, \exists! s' \in S_{BN} : Type(s') = arc \wedge$$
$$\wedge \Sigma(P_s(sorg)) = P_{s'}(sorg) \wedge \Sigma(P_s(dest)) = P_{s'}(dest)$$

Inoltre valgono le seguenti proprietà degli elementi della struttura:

$$P_{s'}(cpt) = \begin{cases} tipo\ basic & se\ \exists s : \Sigma(s) = s' \wedge Type(s) = basic\ event \\ tipo\ or & se\ \exists s : \Sigma(s) = s' \wedge Type(s) = OR\ gate \\ tipo\ and & se\ \exists s : \Sigma(s) = s' \wedge Type(s) = AND\ gate \end{cases}$$

dove i tipi delle cpt sono quelle definiti rispettivamente in Tabella 4, in Tabella 5 ed in Tabella 6.

Il multiformalismo implicito

Sviluppiamo adesso un esempio notevole di traslazione: il caso del multiformalismo implicito, la cui definizione è stata precedentemente data nell'ambito della metodologia di OsMoSys attraverso la formalizzazione dei *Composed Implicit Multiformalism Model Class*. Il meccanismo del multiformalismo implicito è molto potente in quanto permette di potenziare l'espressività di un linguaggio. Tali formalismi non sono però facilmente risolvibili in quanto, dato che è teoricamente possibile "ibridare" un qualsiasi formalismo con un qualsiasi altro, i risolutori attuali si concentrano quasi esclusivamente su *Flat Model Class*. Uno dei possibili modi per analizzare un modello conforme ad un tale linguaggio può essere dunque quello di considerare trasformazioni che operino su Composed Implicit Multiformalism Model Class nell'ottica della loro traslazione in Flat Model Class. Chiameremo F_B, F_I ed F_O rispettivamente il formalismo *base*, ossia quello dal quale si eredita il multiformalismo implicito, il formalismo *implicito* stesso ed il formalismo *oggetto*, ossia quello al quale il flat model class di destinazione è conforme:

$$(F_B, \varepsilon_B), \quad (F_O, \varepsilon_O), \quad (F_I, \varepsilon_I) : \varepsilon_I = \varepsilon_B \cup IN \cup IE$$

Di particolare interesse è la definizione di una traslazione tra F_I ed F_O quando ne esiste una già definita tra F_B ed F_O. Con riferimento alla Figura 12, l'obiettivo è caratterizzare la trasformazione T* a partire da T.

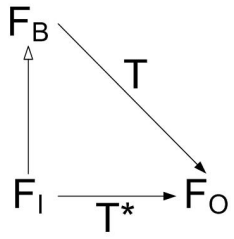

Figura 12: Relazioni tra traslazioni nel multiformalismo implicito

Analizzeremo tale problema in relazione all'esempio del formalismo dei Repairable Fault Trees. Questo formalismo nasce dalla necessità di preservare la semplicità d'uso del formalismo dei Fault Tree permettendo a tale formalismo di modellare politiche di riparazione dei guasti, necessari per la corretta modellazione ed analisi di sistemi riparabili . Tale miglioramento si quantifica nell'introduzione di un nuovo elemento del formalismo detto Repair Box (RB) che, viene collegato all'albero dei guasti attraverso due diversi tipi di archi. Il primo tipo arco lo collega agli eventi trigger il cui scatto attiva il meccanismo di riparazione della Repair Box, il secondo tipo lo collega ai basic event su cui la riparazione si attua. L'astrazione della Repair Box permette dunque di tenere in considerazione:

- la condizione d'attivazione della riparazione;
- l'algoritmo di riparazione dell'evento;
- l'insieme degli elementi riparabili.

Partendo dal modello a Fault Tree raffigurato in Figura 10, è possibile costruire un RFT aggiungendo ad esempio un Repair Box che, innescato dall'evento E, produca la riparazione della causa di guasto tra quelle possibili (A, B e C) come illustrato in Figura 13.

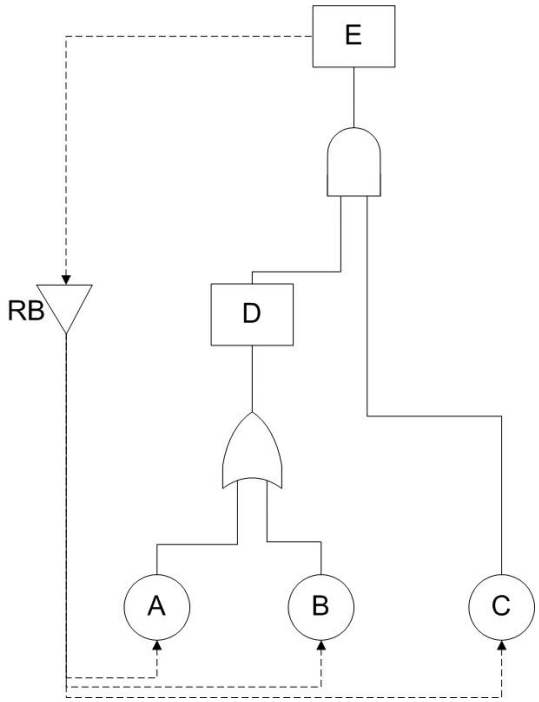

Figura 13: Estensione RFT del modello di Figura 10.

Attualmente l'unico modo di analizzare un modello conforme agli RFT è quello di tradurre tale modello in un modello GSPN. In tale caso il linguaggio ε_B è costituito da quello dei Fault Tree,

ε_I dagli RFT ed ε_O è quello delle GSPN. Caratterizziamo dunque la traslazione T* tra RFT e GSPN a partire da quella T tra FT e GSPN.
Partendo dalle considerazioni tra le possibili trasformazioni tra Fault Trees e Reti di Petri, consideriamo il seguente approccio alla traslazione tra FT e GSPN conforme a quanto già presente in letteratura . Siano dunque considerati gli insiemi dei tipi di elementi del linguaggio:

$$\varepsilon_B = \{basic\ event, OR\ gate, AND\ gate, event, arc\}$$
$$\varepsilon_O = \{place, stochtastic, immediate, arc, inhibitor\ arc\}$$

La trasformazione T è caratterizzata dalla regole raffigurate in Figura 14.

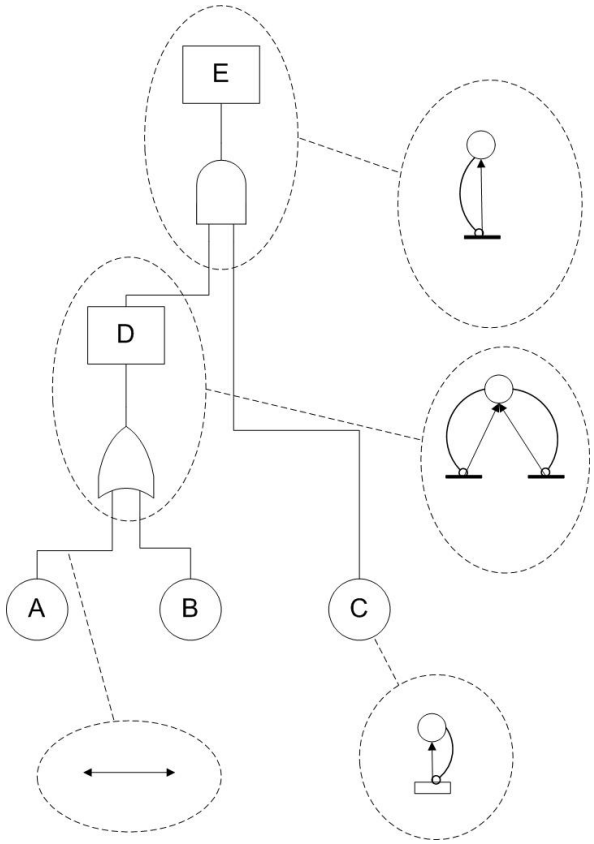

Figura 14: Trasformazione di FT in GSPN

Sia dunque

$$\Sigma : S_B \rightarrow P(S_O)$$

la funzione che trasforma la struttura del modello sorgente dai Fault Tree alle GSPN. Applicando T al modello sorgente, il fault tree può essere riscritto in termini di GSPN come in Figura 15.

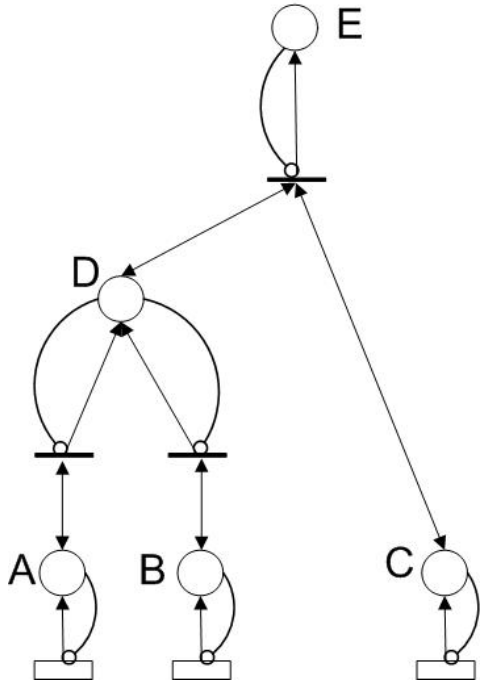

Figura 15: Traduzione GSPN del modello ad FT in Figura 10.

Consideriamo adesso il modello RFT descritto in Figura 13. Caratterizziamo la traduzione di tale modello descrivendo la traslazione dei soli tipi di elementi presenti in IN e IE. L'insieme IN contiene i tipi di Repair Box possibili nel formalismo degli RFT. Diverse semplificazioni sono fatte in relazione alle ipotesi ed ai modelli di Repair Box descritti in ; considereremo per semplicità una politica di riparazione con MTTR = 0 in cui il riparatore non è sempre disponibile. Quindi avremo che:

$$IN = \{RB\}$$

La traduzione di una tale Repair Box in linguaggio GSPN è raffigurata in Figura 16.

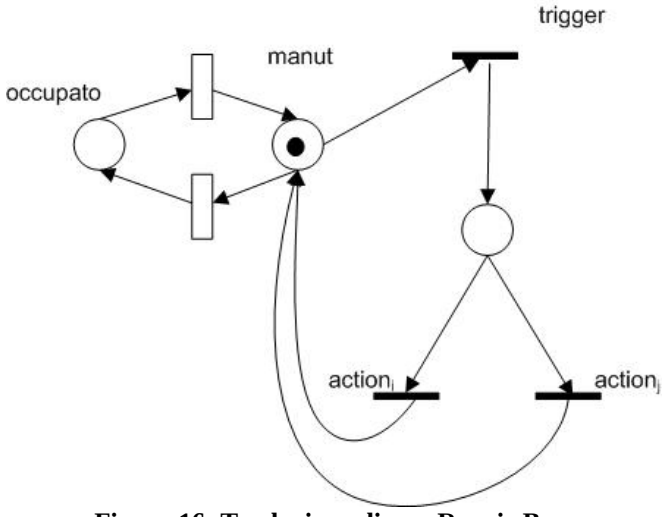

Figura 16: Traduzione di una Repair Box.

Per quanto riguarda l'insieme IE:

$$IE = \{trigger, action\}$$
$$Type(P_{trigger}(sorg)) = event$$
$$Type(P_{trigger}(dest)) = RB$$
$$Type(P_{action}(sorg)) = RB$$
$$Type(P_{action}(dest)) = basic\ event$$

La traduzione di un arco di tipo trigger si traduce con un doppio arco tra il posto relativo alla traduzione GSPN dell'evento e la transizione di trigger della RepairBox, un arco di tipo action si traduce invece in un arco dal posto della traduzione GSPN del basic event alla transizione di action della Repair Box.
La rete descritta in Figura 16 può avere diverse transizioni di tipo action. Lo scopo di una transizione di tale tipo è quello di svuotare i posti che si sono riempiti nell'attivazione dell'evento trigger in modo da "riparare" il sistema. Esiste una relazione tra il numero di queste transizioni ed il concetto di insieme di taglio minimo (minimal cut set). Ricordiamo che si definisce insieme di taglio minimo il più piccolo insieme di basic event una cui occorrenza contemporanea provoca lo scatto del top event .
Indichiamo, come in , con $\bullet RB$ l'insieme dei basic event collegati da un arco action alla Repair Box (tale insieme è anche detto insieme di copertura elementare) ed indichiamo con $MCS=\{mcs_1, \ldots, mcs_N\}$ l'insieme degli insiemi di taglio minimo rispetto all'evento trigger. Deve valere che:

$$\forall i \in \{1,\ldots,n\}\ mcs_i \subseteq \bullet RB$$

Possiamo dire che il numero delle transizioni di action relative alla traduzione di GSPN della Repair Box considerata è pari al numero degli insiemi di taglio minimi del fault tree. Inoltre la generica transizione di action considerata avrà il compito di svuotare dai token tutti i posti relativi ai basic event compresi nell'insieme di taglio minimo associato nonché tutti gli eventi non basici implicati dall'insieme di taglio stesso.
Rispetto al nostro fault tree di riferimento (Figura 10) ci saranno due insiemi di taglio minimi: MCS = {{A, C}, {B, C}}. Pertanto avremo due transizioni di tipo action:

- $action_1$, avente archi entranti dai posti relativi alle traduzioni degli eventi A, C, D ed E;
- $action_2$, avente archi entranti dai posti relativi alle traduzioni degli eventi B, C, D ed E.

Traduciamo formalmente le seguenti affermazioni. Sia T_A l'insieme delle transizioni di action della traduzione GSPN della Repair Box, definiamo la relazione R:

$$R \subseteq MCS \times T_A : \forall t \in T_A\ \exists! mcs \in MCS \wedge \forall\, mcs \in MCS\ \exists! t \in T_A : (mcs, t) \in R$$

Definito inoltre il seguente insieme:

$$\forall mcs \in MCS, \overline{mcs} = mcs \cup \{e : Type(e) = event \wedge mcs \Longrightarrow e\}$$

corrispondente ad un insieme di taglio non minimo, comprendente oltre che ai basic event anche gli eventi intermedi implicati dall'insieme di taglio minimo nonché l'evento trigger.
Sia inoltre definito:

$$P_{\overline{mcs}} = \left\{\Sigma(e) \cap P_{FT} : e \in \overline{mcs}\right\}$$

Ossia l'insieme dei posti relativi agli eventi dell'insieme di taglio non minimo della rete di Petri corrispondente alla traduzione del Fault Tree secondo la trasformazione T da FT a GSPN.
Vale dunque la seguente:

$$\forall (mcs,t) \in R,\ {}^\bullet t \supseteq P_{\overline{mcs}} \wedge \nexists p \in ({}^\bullet t - P_{\overline{mcs}}) \cap \Sigma(e) : Type(e) \in \varepsilon_{FT}$$

cioè che ogni transizione di action ha come archi d'ingresso quelli provenienti dai posti corrispondenti alle traduzioni degli eventi contenuti all'insieme di taglio non minimo associato alla transizione stessa. Inoltre ulteriori archi di ingresso alla transizione riguardano esclusivamente l'implementazione della Repair Box.

Notiamo che questo discorso vale sotto le seguenti ipotesi:

- tempi di riparazione nulli;
- politica di riparazione della Repair Box semplice.

Rilassando tali ipotesi la traduzione GSPN della Repair Box deve tener conto di ulteriori fattori che porterà a complicare tale traduzione.
Sommando le regole della traduzione dal Fault Tree a GSPN (da F_B ad F_O) e quelle dei nuovi elementi del formalismo RFT verso GSPN (da F_I-F_B ad F_O), otteniamo la traslazione del modello RFT di Figura 13 in quello GSPN illustrato in Figura 17.

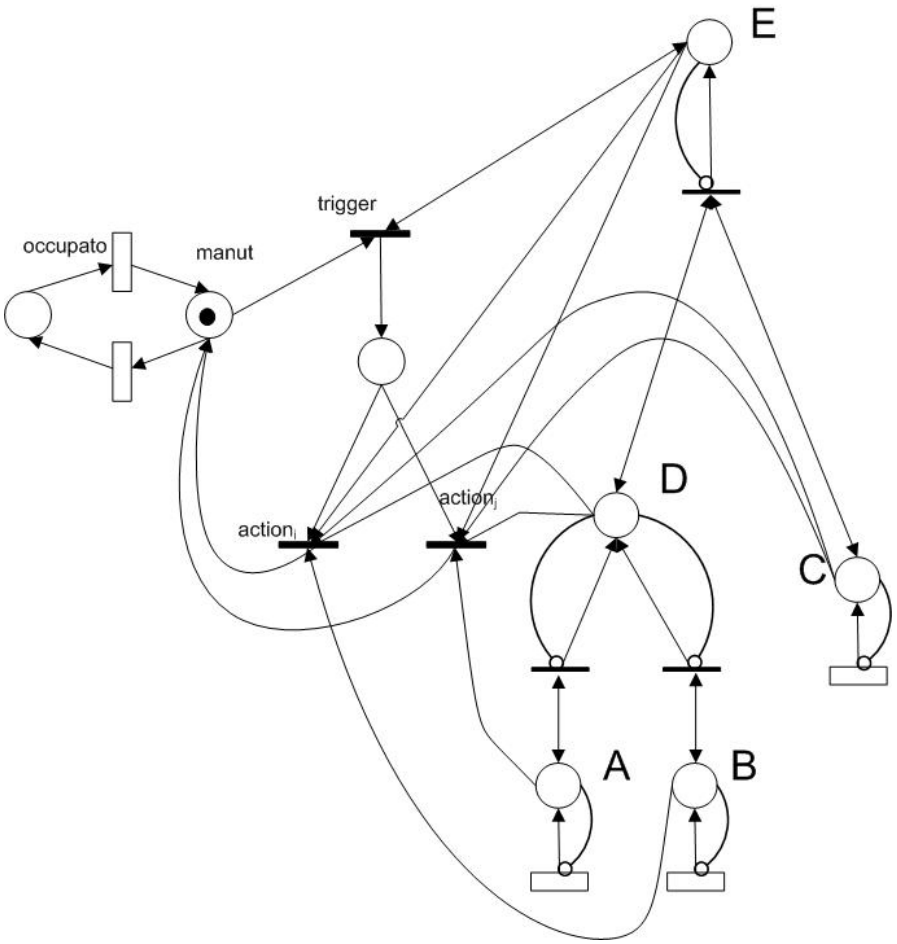

Figura 17: Traslazione GSPN del modello RFT di Figura 10.

Decomposizione

Introduciamo adesso la classe di trasformazioni denominata decomposizione. S'intende per decomposizione una trasformazione uno-molti caratterizzata dalla suddivisione di un modello in più modelli dello stesso formalismo al fine di separare rappresentazioni di aspetti distinti del modello stesso.
Dunque una decomposizione è innanzitutto:

$$T : m_S \in MM_F \rightarrow \overline{m_D} = \left(m_D^1, \ldots, m_D^k\right) \in MM_F^k$$

soggetta al vincolo:

$$m_S \ \mu S \Rightarrow \overline{m_D} \ \mu S$$

Quindi in generale avremo per le decomposizioni che:

$$\forall i \in \{1, \ldots, k\} \ m_D^i = (N_D^i, S_D^i, SM_D^i) \wedge S_D^i \subseteq S_S \wedge \bigwedge_{i=1}^{k} S_D^i = S_S$$

Un esempio di applicazione della decomposizione ci può venire dallo studio delle reti Bayesiane. Tratteremo in questo caso le reti Bayesiane con riferimento alla teoria dei grafi orientati aciclici (DAG). Ciò non vuole essere una trattazione esaustiva della decomposizione di grafi; tale argomento è di forte interesse sia in ambiti accademici che per le forti ricadute sulle applicazioni e, allo stato dell'arte attuale, sono possibili per un grafo diverse decomposizioni in relazione sia allo studio delle reti Bayesiane che per altre applicazioni.
Una di queste decomposizioni, di interesse nell'ambito dell'analisi delle reti bayesiane, è quella di dividere il grafo in componenti tali da ridurre la complessità dello stesso eliminando le variabili irrilevanti rispetto ad una query . Ciò si può ottenere individuando i nodi di cui si vuole conoscere la distribuzione di probabilità ed isolando i nodi che hanno un'influenza su esso, ossia quelli per cui esiste un percorso che arrivi al nodo di interesse. Tale processo è raffigurato in Figura 18 (modello sorgente) ed in Figura 19 (modello destinazione) dove ovviamente è stato scelto il nodo F come "base" della decomposizione.

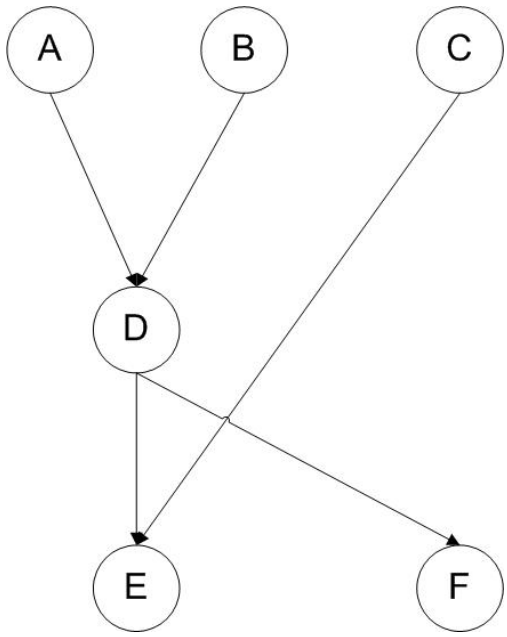

Figura 18: Decomposizione di Reti Bayesiane (modello sorgente)

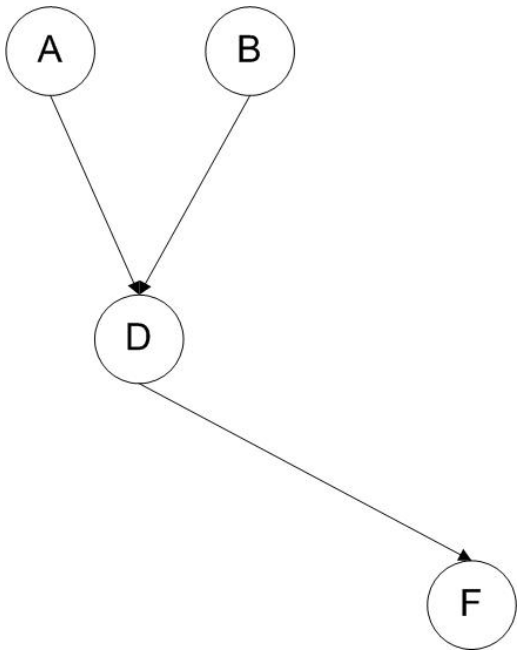

Figura 19: Decomposizione di Reti Bayesiane (modello destinazione)

E' possibile selezionare nel nostro processo di decomposizione anche un nodo intermedio.
Più formalmente detto G=(N,A) il grafo sotteso alla rete Bayesiana con N insieme dei nodi ed A quello degli archi, notato n→m l'arco che va dal nodo m a quello n, definiamo la decomposizione sopra descritta con:

$$T : m_S \in MM_F \rightarrow \overline{m_D} = \left(m_D^1, \ldots, m_D^k\right) \in MM_F^k$$

soggetta al vincolo di rappresentazione del sistema, dove:

$$m_S \equiv G = (N_S, A_S), \quad \forall i, j \in \{1,\ldots,k\}\ m_D^i \equiv G_i = (N_i, A_i):$$

$$(1) \quad \forall i \in \{1,\ldots,k\}\ \exists! n \in N_i : \nexists n \rightarrow m \in A_i; n = root(G_i)$$

$$(2) \quad \forall i \in \{1,\ldots,k\}, \forall m \in N_i, \exists n \in N_i, \exists a \in A_i : a = n \rightarrow m \vee a = m \rightarrow n$$

$$(3) \quad \forall i \in \{1,\ldots,k\}, n \in N_i \Rightarrow \forall a = m \rightarrow n \in A_S, a \in A_i \wedge m \in N_i$$

$$(4) \quad \forall i, j \in \{1,\ldots,k\}\ i \neq j \Rightarrow root(G_i) \neq root(G_j)$$

La proprietà (1) vuol dire che ogni modello destinazione può contenere un solo nodo terminale (senza archi uscenti); la proprietà (2) determina la connessione del grafo di ogni modello destinazione; la proprietà (3) implica che per ogni nodo terminale, siano compresi nel grafo del modello destinazione tutti i nodi e gli archi da cui è possibile raggiungere tale nodo; la proprietà (4) implica che non esistono due modelli uguali tra la k-pla dei modelli destinazione.

Formalizzando la decomposizione per tale esempio, supponiamo l'insieme:

$$\bar{S}_S : \left\{n_1,\ldots,n_k\right\} \subseteq S_S : \forall i, j \in \left\{1,\ldots,k\right\}\ i \neq j \Rightarrow n_i \neq n_j$$

che chiameremo base della decomposizione. Possiamo pertanto definire la decomposizione T rispetto alla base $\bar{S}_S$ come un vettore:

$$T = (T_1, \ldots, T_k) : \forall i \in \{1, \ldots, k\}\ T_i = (Cond_i, Act_i)$$

tale che:

$$Cond_i : S_S \rightarrow \{true,\ false\} : Cond_i(s) = \begin{cases} true & \text{se sono soddisfatte le regole (1) (2) (3) e (4)} \\ false & \text{altrimenti} \end{cases}$$

$$Act_i = (\Sigma_i, \Phi_i)$$

$$\Sigma : S_S \rightarrow S_D^i$$

$$\Sigma_i(s) = \begin{cases} s & se\ Cond_i(s) = true \\ \varnothing & altrimenti \end{cases}$$

$$\Phi_i = 1 \quad dove \quad \Sigma_i = 1$$

Tra le decomposizioni distinguiamo ulteriormente la classe delle partizioni che si caratterizzano dalle prime per il fatto che i modelli destinazione non hanno nessun elemento in comune:

$$\forall i, j \in \{1, \ldots, k\}\ i \neq j \Rightarrow S_D^i \cap S_D^j = \emptyset$$

Capitolo IV:

Composizione di modelli in ambito multiformale

In questo capitolo si affronterà il problema della composizione tra modelli. Dopo aver definito il concetto di composizione di modelli nell'ambito della metodologia di OsMoSys ed inquadrato il problema secondo i paradigmi del Model Engineering, si fornirà una possibile classificazione di operatori di composizione in relazione ai diversi paradigmi di interazione tra modelli. Successivamente si analizzerà un esempio di multiformalismo esplicito attraverso l'applicazione di alcuni operatori di composizionalità.

Composizione di modelli

Una categoria molto importante di trasformazioni di modelli è quella della composizione. Diversi approcci sono stati dati al principio di composizionalità. La maggior parte di questi approcci possono ricondursi al principio di Frege:

"Il significato di un'espressione complessa è determinata dalla sua struttura e dal significato dei componenti."

Gottlob Frege

Diversi approcci sono stati proposti al fine di formalizzare il concetto di composizionalità. Una pietra miliare in tale contesto è rappresentata dal lavoro di Montague che sviluppa una teoria generale di sintassi e semantica sia per i linguaggi formali che per quelli naturali. Oltre a tale aspetto, Montague formalizzò anche il concetto di composizionalità all'interno di una teoria che legasse gli aspetti sintattici di tale composizione con quelli semantici dei componenti. Tale interpretazione è detta *composizionalità forte* al contrario di altri approcci che rilassano il legame tra sintassi e semantica detti appunto *composizionalità debole*.
Il nostro principio guida sarà quello dell'approccio della composizionalità forte in quanto ci proponiamo di fornire un inquadramento teorico all'interno della metodologia e del framework di OsMoSys tale da poter definire, con uno stesso insieme di modelli sorgente, diversi modelli destinazione con diverse semantiche in relazione ai diversi operatori di composizionalità utilizzati.
Ricordiamo innanzitutto il modo in cui viene implementata in OsMoSys la composizionalità tra modelli . Viene definito in OsMoSys il *formalismo bridge* come il meta-modello capace di "collegare" diversi modelli conformi a formalismi diversi. Più formalmente, dati m modelli, ognuno dei quali conforme ad un particolare formalismo, definiamo formalismo bridge:

$$(B, \varepsilon_B) : \varepsilon_B = A_B \cup OP_B \cup Ext_B$$

tali che A_B è l'insieme degli archi verso un operatore (elemento di OP_B) e/o verso un elemento di Ext_B. Questo ultimo insieme è caratterizzato da:

$$Ext_B = \bigsqcup_{i=1}^{m} Ext(\varepsilon_i)$$
$$\forall i \in \{1, \ldots, m\},\ Ext(\varepsilon_i) = \left\{ et : et = Type(e) \wedge e \in External_{S_i} \right\}$$

Un formalismo bridge contiene dunque un operatore che determina la semantica della composizione ed il riferimento alle interfacce dei modelli che può comporre.
Riprendendo adesso la formalizzazione data in questa tesi, definiamo composizione di modelli

rispetto al sistema S una trasformazione molti-uno:

$$T : \overline{m_S} = \left(m_S^1, \ldots, m_S^k\right) \in MM_{F_S^1} \times \cdots \times MM_{F_S^k} \rightarrow m_D \in MM_{F_D}$$

soggetta al vincolo di conservazione della rappresentatività del sistema:

$$\overline{m_S}\ \mu S \Rightarrow m_D\ \mu S$$

Consideriamo in tale contesto la seguente (k+1)-pla di modelli sorgenti:

$$\overline{m_S} = \left(op, m_S^1, \ldots, m_S^k\right) \in MM_{OP} \times MM_{F_S^1} \times \cdots \times MM_{F_S^k}$$

dove il modello *op* è un modello conforme ad un particolare formalismo ed è detto operatore di composizionalità. Tale formalismo s'inquadra nel contesto della composizionalità classica attraverso l'interpretazione dei modelli m^i_S come i componenti della nostra "espressione" composizionale e dell'operatore *op*, conforme al formalismo bridge OP, come alla struttura della composizione.
Un modo alternativo di esprimere la composizione come trasformazione di modelli è dunque la seguente:

$$T : \overline{m_S} = \left(op, m_S^1, \ldots, m_S^k\right) \in MM_{OP} \times MM_{F_S^1} \times \cdots \times MM_{F_S^k} \rightarrow m_D \in MM_{F_D}$$
$$\Updownarrow$$
$$T_{op} : \overline{m_S} = \left(m_S^1, \ldots, m_S^k\right) \in MM_{F_S^1} \times \cdots \times MM_{F_S^k} \rightarrow m_D \in MM_{F_D}$$

Paradigmi di interazione e operatori di composizione

A partire dalla definizione formale di composizione, diamo adesso una classificazione degli operatori di composizione tra modelli in OsMoSys. Per classificare correttamente tali operatori non è possibile prescindere dal meccanismo di interazione tra i modelli.
In letteratura sono stati effettuati diversi tentativi di classificazione di tali modalità di interazione in ambiti più o meno generali. Citiamo a tal fine il lavoro di Andrews che definisce e classifica i meccanismi di interazione tra processi in ambito distribuito, quello di Salaün et al , dove vengono definiti modalità di composizione per formalismi eterogenei e quello di Abadi et al. dove viene definito un approccio alla composizionalità tra modelli conservativo rispetto alle proprietà di sicurezza(safety) e vivezza(liveness).
Partendo da tali riflessioni, definiamo innanzitutto i due diversi aspetti di un modello: quello orientato ai dati, più legato ad aspetti "statici", e quello orientato al comportamento, legato invece all'evoluzione "dinamica" del modello. Tali aspetti coesistono nello stesso istante in un generico modello e determinano in parti eguali la sua semantica. Abbiamo quindi una prima distinzione tra dati e comportamento. Inoltre i modelli di accoppiamento tra processi (ed in senso più lato tra oggetti) possono riassumersi in stretto, relativo alla tecnica della memoria condivisa, e lasco, in cui la comunicazione avviene per scambio di messaggi. La seconda divisione è dunque relativa alla coppia condivisione/comunicazione. Ponendo queste coppie (dati/comportamento e condivisione/scambio) lungo i lati di una tabella possiamo definire quattro famiglie di operatori di composizionalità (vedi Tabella 7):

Tabella 7: Classificazione degli operatori di composizionalità

	Dati	**Comportamento**
Condivisione	Risorse	Stato/Evento

Comunicazione	Copia dato	Scambio di messaggi

La prima categoria è rappresentata dagli operatori di **condivisione di risorse** che definiscono una semantica di composizione di più modelli quando essi concorrono all'utilizzo di dati o strutture dati comuni. Un semplice esempio ci viene dall'ambito delle reti di Petri. Con riferimento al modello raffigurato in Figura 20, consideriamo la composizione di tale modello con se stesso attraverso la condivisione di risorse (in realtà componiamo due istanze distinte dello stesso modello): in particolare relativamente al posto *risorsa*. Operatori tipici di questa famiglia sono quelli relativi alla gestione di risorse condivise in ambiente a memoria comune: i semafori.

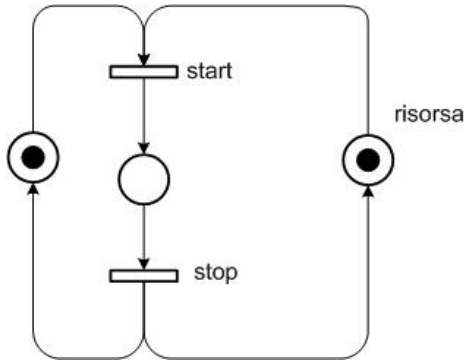

Figura 20: L'operatore di composizionalità di condivisione risorse: un esempio a reti di Petri

Seconda categoria di operatore di composizione è quella relativa alla **condivisione di comportamenti**. A tale categoria fanno capo i seguenti operatori:

- *state sharing***,** i due modelli condividono uno stato locale in quanto tali stati rappresentano uno stesso stato fisico nel sistema reale;
- *event sharing***,** i due modelli condividono una transizione tra stati in quanto tali attività rappresentano una stessa attività nel sistema reale.

La terza categoria di operatori di composizionalità è rappresentata dalla **copia dato**, relativa a paradigmi di interazione basati sullo scambio unidirezionale di dati tra modelli.[18] Molti degli operatori definiti in , rientrano in questa categoria: *Copy Properties Operator* (CPP), *Copy Elaborated Properties Operator* (CEP), *Copy Results Operator* (CPR), *Copy Elaborated Results Operator* (CER).

Infine enunciamo la quarta categoria concernente lo **scambio di messaggi** tra modelli, tipico di paradigmi di comunicazione asincroni. Rientrano in tale categoria:

- *simple message*, il modello di comunicazione prevede l'invio di un messaggio da una sola sorgente ad un solo destinatario,
- *broadcast message*, vi è una sola sorgente ma più destinatari ognuno dei quali riceve il messaggio,
- *anycast message*, vi è una sola sorgente ma più destinatari; solo uno di questi però, scelto in modo casuale, riceve il messaggio,
- *deterministic unicast*, vi è una sola sorgente ma più destinatari; solo uno di essi però, scelto in modo esplicito, riceve il messaggio.

Descriveremo in dettaglio alcuni di questi operatori in questo e nel prossimo capitolo fornendo per essi un esempio inerente ai metodi formali.

[18] Ciò corrisponde ad avere un modello che fornisce il dato ed un altro che lo riceve e lo elabora come in una sorta di modello produttore-consumatore.

Operatori di copia dato

Formalizziamo adesso la categoria degli operatori di copia. Sia dunque definito un formalismo OP relativo agli operatori di composizionalità e sia definito il formalismo COP, relativo agli operatori di composizionalità di copia, come sottolinguaggio di OP:

$$\varepsilon_{COP} \supseteq \{CPP, CEP, CPR, CER\}$$

Consideriamo l'operatore cop χ COP. Definiamo composizione di modelli attraverso un operatore di composizionalità di tipo copia la seguente:

$$T_{cop} : \left(m_S^i, m_S^o\right) \in MM_{F_S}^i \times MM_{F_S}^o \rightarrow m_D \in MM_{F_D}$$

dove i tipi di modelli sorgente (i "fattori" della composizione) sono due: uno avente il ruolo di input (fornitore del dato condiviso) e l'altro quello di output (utente dello stesso dato). Tale notazione può essere estesa a casi k-input, n-output:

$$\overline{m_S} = \left(m_{S,1}^i, \ldots, m_{S,k}^i, m_{S,1}^o, \ldots, m_{S,n}^o\right)$$
$$MM_S = MM_{F_{S,1}}^i \times \ldots \times MM_{F_{S,k}}^i \times MM_{F_{S,1}}^o \times \ldots \times MM_{F_{S,n}}^o$$
$$T_{cop} : \overline{m_S} \in MM_S \rightarrow m_D \in MM_{F_D}$$

Faremo riferimento per semplicità alla composizione 1-input 1-output.
Sia per un generico modello sorgente:

$$m_S^x = \left(N_S^x, S_S^x, SM_S^x\right) : S_S^x = External_{SS}^x \cup Internal_{SS}^x$$

L'operatore *cop* è dunque caratterizzato dalle seguenti equazioni:

$$\exists\, s \in External_{SS}^i \wedge \exists\, p \in P_s,$$
$$\exists\, s' \in External_{SS}^o \wedge \exists\, p' \in P_{s'} :$$
$$p' = f(p)$$

dove *f* è una funzione di elaborazione della proprietà di input in proprietà di output la cui natura è relativa allo specifico operatore considerato.
Prima di caratterizzare la funzione *f*, esprimiamo l'operazione di risoluzione di un modello attraverso una trasformazione tra il modello stesso ed un modello "soluzione":

$$R : m_S \in M_1 \rightarrow m_D \in M_1$$

Consideriamo adesso le diverse caratterizzazioni della funzione *f* in relazione ai diversi operatori di copia:

- *Copy Properties Operator (CPP)*, l'operatore copia il valore della proprietà dell'elemento di interfaccia di input in quello di output; ciò comporta che: $f = 1$;

- *Copy Elaborated Properties Operator (CEP)*, l'operatore copia una trasformazione del valore della proprietà dell'elemento di interfaccia di input in quello di output; ciò comporta che:

$f \neq 1;$

- *Copy Results Operator (CPR)*, l'operatore copia il valore della proprietà dell'elemento di interfaccia di input in quello di output dopo che esso è stato risolto; ciò comporta che: $f = 1 \circ R;$

- *Copy Elaborated Results Operator (CER)*, l'operatore copia una trasformazione del valore della proprietà dell'elemento di interfaccia di input in quello di output dopo che esso è stato risolto; ciò comporta che: $f = f' \circ R : f' \neq 1.$

Applicazione degli operatori di composizionalità al multiformalismo

Un esempio relativo all'applicazione di tali tipologie di operatori di composizionalità è quello di un sistema di monitoraggio per applicazioni domotiche di gestione allarmi in presenza di condizioni pericolose per la vita dell'uomo. Una di queste applicazioni prende in considerazione un sistema per il controllo antincendio distribuito.
Per tale sistema definiamo l'architettura hardware illustrata in Figura 21.

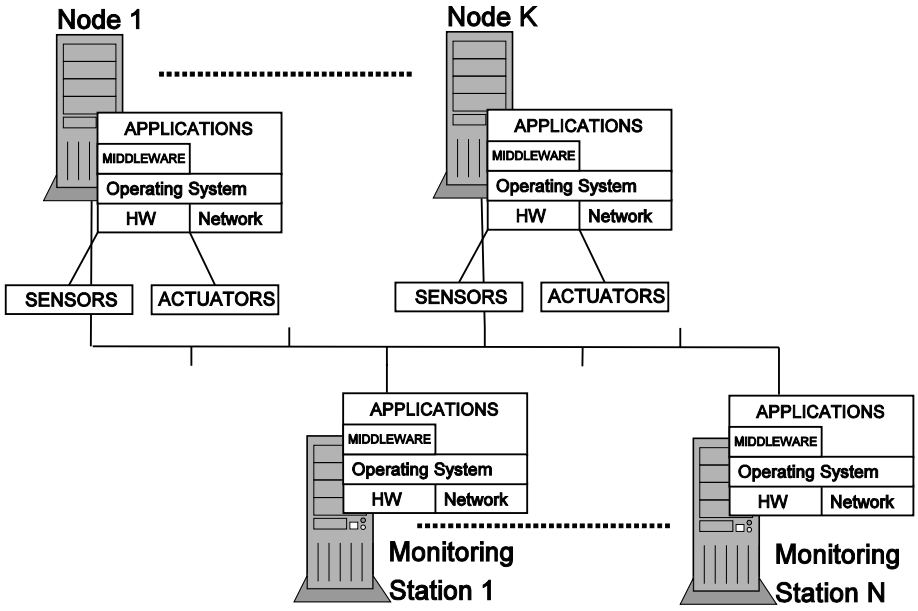

Figura 21: Architettura hardware di un sistema di controllo domotico

L'insieme dei nodi di elaborazione è partizionato in due sottoinsiemi: quello dei K nodi di applicazione, su cui sono attivi software di controllo per sensori ed attuatori, e quello degli N nodi di monitoraggio, che ospita il software relativo al controllo delle condizioni di allarme. L'eterogeneità delle piattaforme hardware e software costringono il progettista a ricercare soluzioni interoperabili e adattabili a qualsiasi soluzione tecnologica. Una buona scelta in tale contesto, è costituita dalle architetture software basati su agenti mobili, tecnologia emergente per la costruzione di software distribuito su piattaforme eterogenee.
Vi sono dunque, dal punto di vista dell'architettura software, una serie di agenti specializzati in specifici compiti che monitorano lo stato del sistema leggendo dai sensori e che interagiscono tra loro al fine di valutare in sicurezza un'eventuale condizione di allarme.
Uno di questi possibili scenari è quello che vede un agente mobile (*Nomad*) controllare i vari nodi dove sono presenti sensori termici al fine di rilevare un'anomalia: in caso positivo, occorre

valutare la possibilità che nell'ambiente siano presenti delle persone attraverso l'interrogazione di un sensore di pressione da parte di un altro agente. Nomad quindi invoca i servizi di tale agente (*Nimrod*) che si sposta verso il nodo in questione ed interroga il sensore di pressione. Qualora l'ambiente sia libero, Nomad avvia la procedura di blocco porte.
Un approccio ad agenti per tali funzionalità deve essere però conciliato con la necessità di rispettare gli stringenti requisiti sui tempi di risposta. Occorre dunque valutare tali tempi dell'applicazione nell'ottica di garantire la sicurezza (safety) del sistema.

L'approccio di modellazione seguito è dettato da considerazioni sulla necessità di:

- valutare quantitativamente i tempi di risposta dei singoli agenti in relazione alla piattaforma hw/sw con la quale interagiscono;
- analizzare quantitativamente scenari di interazione complessi tra agenti.

Tale caratterizzazione del problema in termini di criticità dei requisiti e di complessità dell'applicazione richiede dunque un approccio di modellazione ed analisi multiformale e multirisoluzione. In particolare verranno utilizzati i seguenti formalismi:

- Timed Petri Nets (TPNs) : formalismo derivato da quello delle Petri Nets dove è possibile avere transizioni temporizzate con tempi di scatto costanti e determinati. Tale formalismo è indicato in quanto permette di valutare facilmente le performance di un modello ma è poco scalabile in quanto, per sistemi di grosse dimensioni, i modelli diventano di difficile analisi a causa delle risorse computazionali richieste dagli algoritmi di risoluzione;
- Timed Automata (TA) : formalismo diventato uno standard nella modellazione di sistemi real-time asincroni, ideale per la valutazione di sistemi complessi ma poco indicato per modellare effetti di accodamento e per ottenere informazioni complete sulle performance del sistema.

Tali linguaggi sono stati scelti per modellare il primo, gli effetti di "basso livello" come l'interazione degli agenti con il sistema operativo e con quello di comunicazione, l'altro, pur con un'estensione per tenere conto degli effetti degli accodamenti, per valutare i comportamenti relativi alle interazioni tra gli agenti.

Pertanto la metodologia di modellazione è chiaramente multiformale e può essere qui riassunta. Il primo passo è la valutazione dei tempi di esecuzione del singolo agente quando interagisce con un solo altro agente su uno stesso nodo di esecuzione ed il sistema operativo (schedulatore). Questa caratterizzazione è effettuata attraverso modelli conformi alle Timed Petri Nets parametrizzate con valori relativi a misure effettuate attraverso instrumentazione del software. Il secondo passo consiste nel modellare le interazioni complesse tra gli agenti attraverso un modello Timed Automata e verificando l'aderenza del sistema ai requisiti di performance richiesti attraverso la formulazione di espressioni in Real-Time Computational Tree Logic (RTCTL). E' necessario estendere questo step con i risultati del primo in modo da considerare nel modello Timed Automata gli effetti degli accodamenti tra task che vengono eseguito sullo stesso nodo di elaborazione. Pertanto i risultati del primo passo sono utilizzati come input per il modello TA

Si profila quindi l'uso dell'operatore CER (Copy Elaborated Results) all'interno del modello (vedi Figura 22). Tale operatore copia i valori delle metriche di performance del modello TPN risolto in parametri del modello TA.

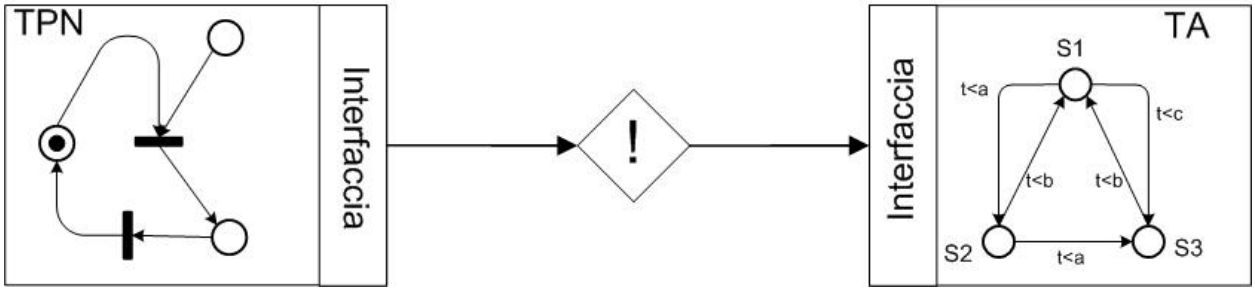

Figura 22: Applicazione dell'operatore Copy Elaborated Results (CER)

Per ulteriori dettagli sull’applicazione e sullo studio effettuato per tale sistema fare riferimento a Moscato et al. .

Capitolo V:

Composizione client-server per modelli a reti di Petri

Nell'ambito della metodologia di composizione già enucleata nel capitolo precedente, si descriveranno e formalizzeranno in questo capitolo gli operatori di composizionalità relativi al caso di scambio di messaggi. Tali operatori verranno formalizzati ed applicati a modelli ad oggetti basati sul formalismo delle Stochastic Well-Formed Nets (SWN), una tipologia di reti di Petri ad alto livello, e per paradigmi di interazione client-server. Verrà illustrato infine un'applicazione di tali operatori al caso di studio del contact center.

Operatori di composizione basati su scambio di messaggi

Formalizziamo adesso, rispetto a quanto già detto nel capitolo precedente, gli operatori di composizionalità basati sul paradigma dello scambio di messaggi. Definiamo, come nel caso precedente, il formalismo MSOP sottolinguaggio di OP:

$$\varepsilon_{MSOP} \supseteq \{SM, BC, AC, DU\}$$

ossia formalismo contenete gli operatori: *simple message* (SM), *broadcast message* (BC), *anycast message* (AC) e *deterministic unicast* (DU).
Consideriamo l'operatore msop χ MSOP. Definiamo dunque composizione di modelli attraverso un operatore di composizionalità basato su scambio di messaggio:

$$T_{msop} : \left(m_S^s, m_S^r\right) \in MM_{F_S}^s \times MM_{F_S}^r \rightarrow m_D \in MM_{F_D}$$

Ci sono due tipi di modelli sorgente: uno avente il ruolo di mittente (*sender*) del messaggio e l'altro del ricevitore (*receiver*). Anche in questo caso la trasformazione può essere estesa a casi k-n:

$$\overline{m_S} = \left(m_{S,1}^s, \ldots, m_{S,k}^s, m_{S,1}^r, \ldots, m_{S,n}^r\right)$$
$$MM_S = MM_{F_{S,1}}^s \times \ldots \times MM_{F_{S,k}}^s \times MM_{F_{S,1}}^r \times \ldots \times MM_{F_{S,n}}^r$$
$$T_{msop} : \overline{m_S} \in MM_S \rightarrow m_D \in MM_{F_D}$$

Rinominato il vettore dei modelli sorgente:

$$\overline{m_S} = \left(m_{S,1}^s, \ldots, m_{S,k}^s, m_{S,1}^r, \ldots, m_{S,n}^r\right) = \left(m_{S,1}, \ldots, m_{S,k+n}\right)$$

valgono per tale tipologia di operatori di composizionalità le seguenti.

$$\forall j \in \{1, \ldots, k+n\},\ m_{S,j} :$$
$$External_{S,j} = \text{Send}_{S,j} \cup \text{Receive}_{S,j} \wedge \text{Send}_{S,j} \cap \text{Receive}_{S,j} = \varnothing$$

In tale contesto specializziamo quanto detto per le due categorie di modelli sorgente. Per i mittenti vale che:

$$\forall i \in \{1,\ldots,k\},\ m^s_{S,i}:$$
$$\forall s_i \in \mathrm{Send}^s_{S,i},\ \exists \mathrm{msgS}_i \in P_{s_i} : \mathrm{msgS}_i = (\mathrm{nome}, \mathrm{dest}, \mathrm{dato})$$

mentre per i destinatari vale che:

$$\forall j \in \{1,\ldots,n\},\ m^r_{S,j}:$$
$$\forall s_j \in \mathrm{Receive}^r_{S,j}, \exists \mathrm{msgR}_j \in P_{s_j} : \mathrm{msgR}_j = (\mathrm{nome}, \mathrm{dato})$$

I diversi operatori specializzano la relazione tra i k messaggi spediti e gli n ricevuti:

- *Simple Message (SM)*, si caratterizza dall'essere una composizione tra un solo mittente ed un solo destinatario dove ogni messaggio inviato dal mittente viene consegnato al destinatario: $\mathrm{msgR} = (\mathrm{msgS.nome}, \mathrm{msgS.dato})$

- *Broadcast Message (BC)*, tale operatore k- n si caratterizza per il fatto che ogni messaggio inviato da uno dei modelli mittenti viene consegnato ad ognuno dei modelli destinatari:
  $\exists i \in \{1,\ldots,k\} : \mathrm{msgS_i} \neq \varnothing \Longrightarrow$
  $\forall j \in \{1,\ldots,n\},\ \mathrm{msgR_j} = (\mathrm{msgS_i.nome}, \mathrm{msgS_i.dato})$

- *Anycast Message (AC)*, tale operatore k-mittenti n-destinatari si caratterizza per il fatto che ogni messaggio inviato da uno dei modelli mittenti viene consegnato ad uno ed un solo dei destinatari attraverso politiche determinate esclusivamente dall'operatore (es. casualmente):

  $$\exists i : i \in \{1,\ldots,k\} : \mathrm{msgS}_i \neq \varnothing \Longrightarrow$$
  $$\exists! x : x \in \{1,\ldots,n\} : \forall j : j \in \{1,\ldots,n\}, \mathrm{msgR}_j = \begin{cases} (\mathrm{msgS}_i.\mathrm{nome}, \mathrm{msgS}_i.\mathrm{dato}) & j = x \\ \varnothing & j \neq x \end{cases}$$

- *Deterministic Unicast (DU)*, l'operatore, anch'esso k-mittenti n-destinatari, invia il messaggio trasmesso ad uno solo dei destinatari attraverso politiche basate su informazioni contenute nello stesso messaggio (ad esempio il campo *dest* del messaggio stesso):
  $\exists i \in \{1,\ldots,k\} : \mathrm{msgS_i} \neq \varnothing \Longrightarrow$
  $$\forall j \in \{1,\ldots,n\},\ \mathrm{msgR_j} = \begin{cases} (\mathrm{msgS_i.nome}, \mathrm{msgS_i.dato}) & \eta(\mathrm{msgS_i.dest}) = \mathrm{true} \\ \varnothing & \eta(\mathrm{msgS_i.dest}) = \mathrm{false} \end{cases}$$
  dove la funzione η è opportunamente definita in relazione all'identità delle istanze dei modelli considerate.

Estendiamo adesso tale formalizzazione a paradigmi di interazione client-server. Ci sono due tipi di modelli sorgente, uno avente il ruolo di client del servizio e l'altro del server:

$$\overline{m_S} = \left(m^c_{S,1},\ldots,m^c_{S,k},m^s_{S,1},\ldots,m^s_{S,n}\right)$$
$$MM_S = MM^c_{F_{S,1}} \times \ldots \times MM^c_{F_{S,k}} \times MM^s_{F_{S,1}} \times \ldots \times MM^s_{F_{S,n}}$$
$$T_{csop} : \overline{m_S} \in MM_S \rightarrow m_D \in MM_{F_D}$$

Rinominato il vettore dei modelli sorgente:

$$\overline{m_S} = \left(m^c_{S,1}, \ldots, m^c_{S,k}, m^s_{S,1}, \ldots, m^s_{S,n}\right) = \left(m_{S,1}, \ldots, m_{S,k+n}\right)$$

valgono per tale tipologia di operatori di composizionalità le seguenti.

$$\forall j \in \{1, \ldots, k+n\},\ m_{S,j}:$$
$$External_{S,j} = \mathrm{A}sk_{S,j} \cup Supply_{S,j} \wedge Ask_{S,j} \cap \mathrm{S}upply_{S,j} = \varnothing$$

In tale contesto specializziamo per le due categorie di modelli sorgente. Per i client vale che:

$$\forall i \in \{1, \ldots, k\},\ m^s_{S,i}:$$
$$\forall s_i \in \mathrm{A}sk^s_{S,i},\ \exists \mathrm{s}rv\mathrm{S}_i \in P_{s_i} : \mathrm{s}rv\mathrm{S}_i = (msgR_i, msgG_i),$$
$$msgR_i = (\mathrm{nome}, \mathrm{id}, \mathrm{dest}, \mathrm{dato}) \wedge msgG_i = (\mathrm{nome}, \mathrm{id}, \mathrm{dato})$$

esistono cioè per ogni elemento di interfaccia un servizio associato ad esso tale che questo servizio preveda due messaggi: uno di *retrive* e l'altro di *get*. Per i server vale che:

$$\forall i \in \{1, \ldots, k\},\ m^s_{S,i}:$$
$$\forall s_i \in Supply^s_{S,i},\ \exists \mathrm{s}rv\mathrm{S}_i \in P_{s_i} : \mathrm{s}rv\mathrm{S}_i = (msgA_i, msgP_i),$$
$$msgA_i = (\mathrm{nome}, \mathrm{id}, \mathrm{dato}) \wedge msgP_i = (\mathrm{nome}, \mathrm{id}, \mathrm{dest}, \mathrm{dato})$$

Il servizio caratteristico di ogni elemento dell'interfaccia del server è costituito dunque da una parte di *accept* ed una di *provide*. E' possibile estendere la caratterizzazione formale dei quattro operatori già definiti per il message-sending al paradigma client-server, modificando opportunamente la semantica delle operazioni:

- *Simple Message*[19] *(SM)*, un client richiede un servizio ad uno ed un solo server;
- *Broadcast (BC)*, un client, collegato ad un pool di n server, richiede un servizio a tutti i server presenti. L'operazione è conclusa quando tutti i server hanno risposto al client;
- *Anycast (AC)*, un client, collegato ad un pool di server, richiede il servizio ad uno qualsiasi dei server. Tale server è scelto attraverso politiche determinate esclusivamente dall'operatore (es. casualmente);
- *Deterministic Unicast (DU)*, l'operatore, anch'esso multi server, richiede il servizio al server scelto sulla base di informazioni contenute nello stesso messaggio (ad esempio il campo *dest* del messaggio stesso).

Modelli message-sending e client-server basati sulle reti di Petri

Prima di applicare quanto formalizzato, definiamo un modello di interazione client-server per il formalismo delle reti di Petri e più in particolare per quello delle Stochastic Well-Formed Nets . Diverse estensioni alle reti di Petri sono state proposte in letteratura nell'ottica di dotare tali formalismi delle caratteristiche di object-orientation. Ricordiamo tra essi: le CO-OPN , le OOPN e CLOWN ; tutti questi approcci ricorrono a formalismi di alto livello per le reti di Petri. L'approccio qui proposto parte da quanto sviluppato da Sibertin-Blanc con le Comunicative Nets, basati su paradigmi di comunicazione message sending, e con le Cooperative Nets, basate invece su paradigmi client-server.
Entrambi questi approcci verranno inquadrati all'interno della metodologia di OsMoSys.
Denotiamo una struttura SWN la seguente:

[19] Conserveremo tale dicitura per questo tipo di operatore anche se, a rigore, esso dovrebbe essere rinominato in *Simple Service*.

$$Net = (P, T, C, cd, \mathrm{Pr}e, Post, Inh, \pi, \phi, \vartheta)$$

ed un SWN la composizione tra una struttura ed una marcatura iniziale:

$$(Net, M_0)$$

In relazione al paradigma message-sending, definiamo la model class[20] dei modelli mittenti conformi al formalismo delle SWN:

$$Sender_{SWN} = (N, S, SM):$$
$$(1)\ N = SwnSender$$
$$(2)\ S = (Net, M_0)$$
$$(3)\ External_S \subseteq T$$

e le model class dei modelli destinatari:

$$\mathrm{Re}ceiver_{SWN} = (N, S, SM):$$
$$(1)\ N = Swn\,\mathrm{Re}ceiver$$
$$(2)\ S = (Net, M_0)$$
$$(3)\ External_S \subseteq P$$

Per i mittenti valgono le seguenti proprietà:

$$t \in External_S \Rightarrow cd(t) = \prod_i C_i:$$
$$(1)\ \exists C_x = NOME$$
$$(2)\ \exists C_y = DEST$$
$$(3)\ \exists C_z = DATO$$

Informalmente ciò comporta che nel dominio dei colori delle transizioni di interfaccia (dette transizioni di *send*) devono essere presenti tre classi elementari di colori tali: NOME, DEST, DATO.
I modelli destinatari godono della proprietà:

$$p \in External_S \Rightarrow cd(p) = \prod_i C_i:$$
$$(1)\ \exists C_x = NOME$$
$$(2)\ \exists C_y = DATO$$
$$p \in External_S \Rightarrow {}^{\bullet}p = \varnothing \wedge M_0(p) = 0$$

Informalmente ciò vuole dire che, oltre alla presenza delle classi elementari NOME e DATO nei posti di interfaccia (detti posti di *receive*), è necessario che tali posti non abbiano archi entranti e che la cui marcatura iniziale è nulla.
Passiamo adesso alla formalizzazione dei modelli basati su paradigma client-server, definendo la model class dei modelli client conformi al formalismo delle SWN:

[20] Tutti i modelli esaminati in questo capitolo si intendono essere flat model class: pertanto SM=∅.

$$Client_{SWN} = (N, S, SM):$$
$$(1)\ N = SwnClient$$
$$(2)\ S = (Net, M_0)$$
$$(3)\ External_S \subset Tr \times Tg : \forall tr \in Tr, \forall tg \in Tg, \exists!(tr, tg) \in External_S$$
$$Tr, Tg \subset T \wedge Tr \cap Tg = \varnothing \wedge |Tr| = |Tg|$$

dove Tr (insieme delle transizioni di *request*) e Tg (insieme delle transizioni di *retrive*) sono sottoinsiemi disgiunti di pari cardinalità dell'insieme delle transizioni. Vale la seguente:

$$(tr, tg) \in External_S \Rightarrow cd(tr) = \prod_i C_i : \exists C_x = NOME \wedge \exists C_y = ID \wedge \exists C_z = DEST \wedge \exists C_w = DATO$$
$$(tr, tg) \in External_S \Rightarrow cd(tg) = \prod_i C_i : \exists C_x = NOME \wedge \exists C_y = ID \wedge \exists C_z = DATO$$

Passiamo adesso alla formalizzazione dei modelli server conformi al formalismo delle SWN:

$$Server_{SWN} = (N, S, SM):$$
$$(1)\ N = SwnClient$$
$$(2)\ S = (Net, M_0)$$
$$(3)\ External_S \subset Pa \times Pp : \forall pa \in Pa, \forall pp \in \mathrm{Pr}, \exists!(pa, pp) \in External_S$$
$$Pa, Pp \subset P \wedge Pa \cap Pp = \varnothing \wedge |Pa| = |Pp|$$
$$\forall pa \in Pa, {}^{\bullet}pa = \varnothing \wedge M_0(pa) = 0$$
$$\forall pp \in Pp, pp^{\bullet} = \varnothing \wedge M_0(pp) = 0$$

dove Pa (insieme dei posti di *accept*) e Pp (insieme delle posti di *provide*) sono sottoinsiemi disgiunti di pari cardinalità dell'insieme dei posti. Vale la seguente:

$$(pa, pp) \in External_S \Rightarrow cd(pa) = \prod_i C_i : \exists C_x = NOME \wedge \exists C_y = ID \wedge \exists C_z = DATO$$
$$(pa, pp) \in External_S \Rightarrow cd(pp) = \prod_i C_i : \exists C_x = NOME \wedge \exists C_y = ID \wedge \exists C_z = DEST \wedge \exists C_w = DATO$$

Un sistema di modelli client-server richiede, a differenza del paradigma di comunicazione message-sending, il rispetto di alcuni vincoli di buona implementazione in relazione agli elementi interni della struttura dei modelli stessi. Questi possono essere descritti come:

- *onestà del client*: un client richiede il risultato da un server solo se vi è stata una precedente richiesta di servizio;
- *discrezione del client*: un client richiede un servizio solo se in grado di processarne il risultato;
- *onestà del server*: un server accetta una richiesta di servizio solo se è in grado di processarla;
- *discrezione del server*: un sever ritorna un risultato solo se è stata fatta una richiesta relativa.

Non formalizzeremo tali aspetti, necessari alla costruzione di un modello che rappresenti effettivamente un sistema, supponendo che tale vincolo sia già rispettato e constatando che questo stesso vincolo non inficia la definizione dei meccanismi di composizionalità, oggetto di questa trattazione. Per ulteriori approfondimenti su questi aspetti si vedano .

Composizionalità per modelli SWN

Rispetto allo stato dell'arte sulla composizionalità di modelli a reti di Petri citiamo gli approcci di Rojas e di Best et al. .

L'approccio seguito, già pubblicato in Franceschinis et al. , si basa sui seguenti passi:

- formalizzazione di modelli ad oggetti basati su SWN;
- definizione degli operatori di composizione e loro implementazione attraverso modelli SWN;
- labellazione degli elementi di interfaccia dei k+n elementi sorgenti al modello e del modello

di un operatore;

- generazione incrementale della rete per sovrapposizione di posti o transizioni ugualmente labellati tra un modello sorgente ed il modello dell'operatore di composizione.

Tale processo automatizzabile permette dunque di implementare una trasformazione di composizione tale da generare un modello destinazione SWN analizzabile.[21]
Partiremo dai modelli evidenziati nella sezione precedente e ci concentreremo sul secondo punto del processo lasciando come riferimenti per ulteriori approfondimenti sull'implementazione ed automatizzazione delle fasi di labellazione e generazione incrementale dei modelli.
Per quanto riguarda gli operatori di composizione definiamo le loro traduzioni SWN relative sia per il paradigma message-sending che per quello client-server. Partiamo da quello message-sending illustrando rispettivamente queste transizioni come nelle seguenti figure: *Simple Message* (Figura 23), *Broadcast Message* (Figura 24) *Anycast Message* (Figura 25) e *Deterministic Unicast* (Figura 26).

Figura 23: Modello SWN dell'operatore di Simple Message (message-sending)

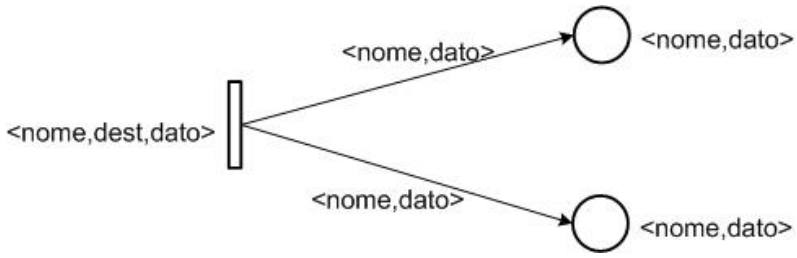

Figura 24: Modello SWN dell'operatore di Broadcast (message-sending)

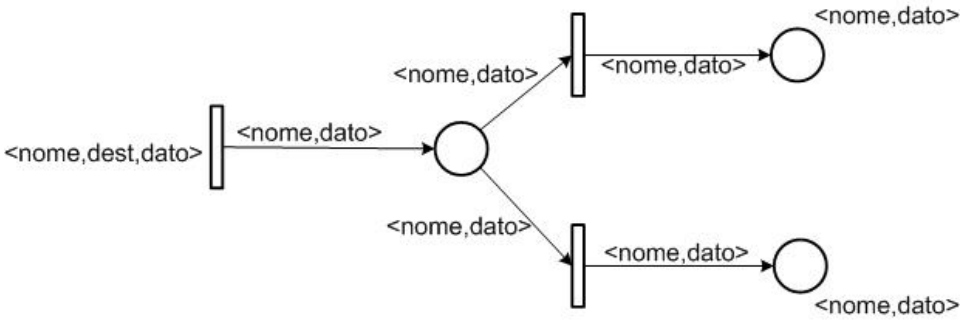

Figura 25: Modello SWN dell'operatore di Anycast (message-sending)

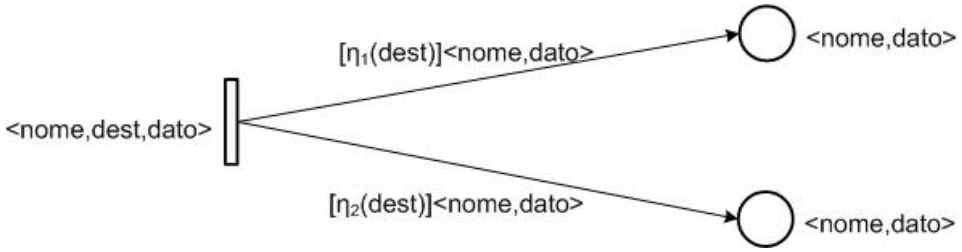

Figura 26: Modello SWN dell'operatore di Deterministic Unicast (message-sending)

Diamo evidenza attraverso la Figura 27 lo schema di principio attraverso il quale vengono poi fuse le transizioni ed i posti tra i modelli SWN sorgenti alla trasformazione di composizione e il modello SWN dell'operatore.

[21] Un esempio di automatizzazione di tale processo si basa sugli strumenti GreatSPN per l'analisi delle reti e Algebra per le caratteristiche di fusione delle reti.

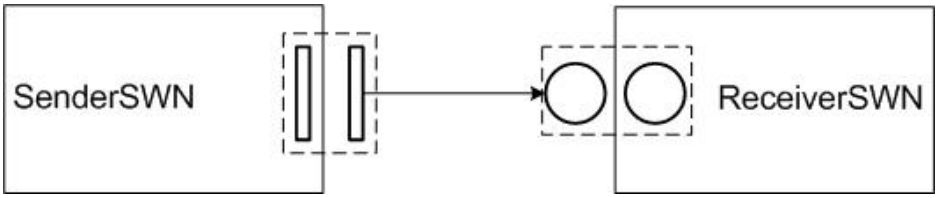

Figura 27: Composizione di modelli in ambito message-sending SWN (operatore Simple Message)

I riquadri tratteggiati indicano gli elementi delle reti che andranno poi fusi tra loro.
Passiamo adesso alla descrizione grafica delle implementazioni degli stessi operatori in ambito client-server: Simple Message (Figura 28), Broadcast (Figura 29), Anycast (Figura 30) e Deterministic Unicast (Figura 31).

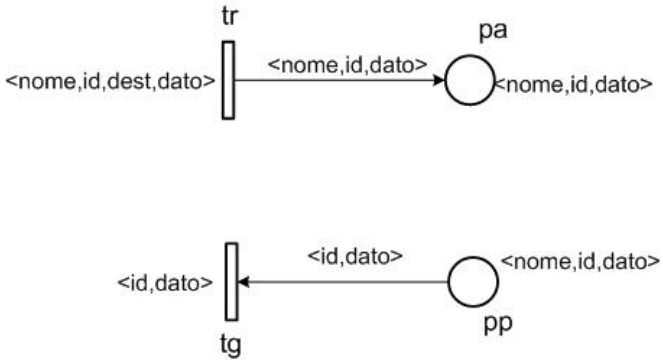

Figura 28: Modello SWN dell'operatore Simple Message (client-server)

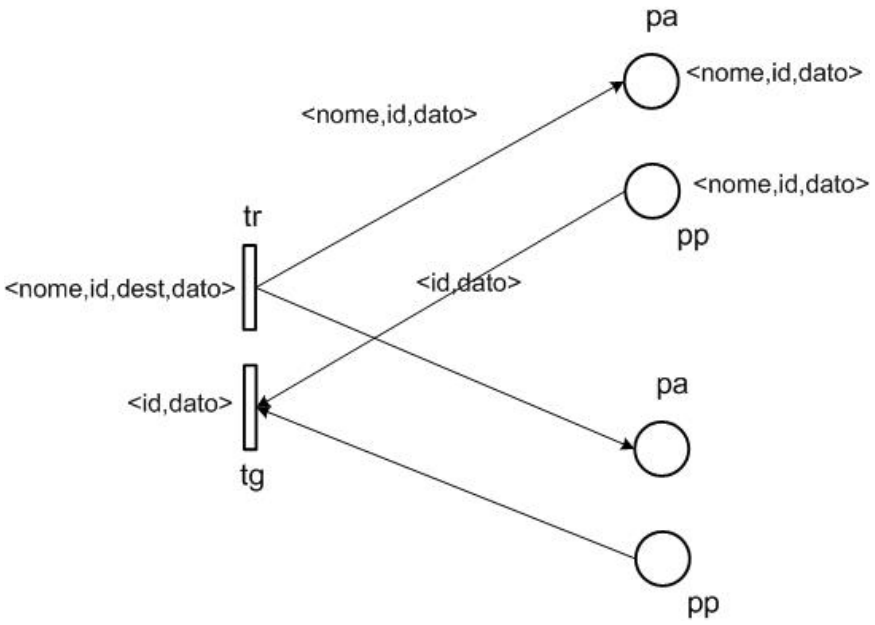

Figura 29: Modello SWN dell'operatore Broadcast (client-server)

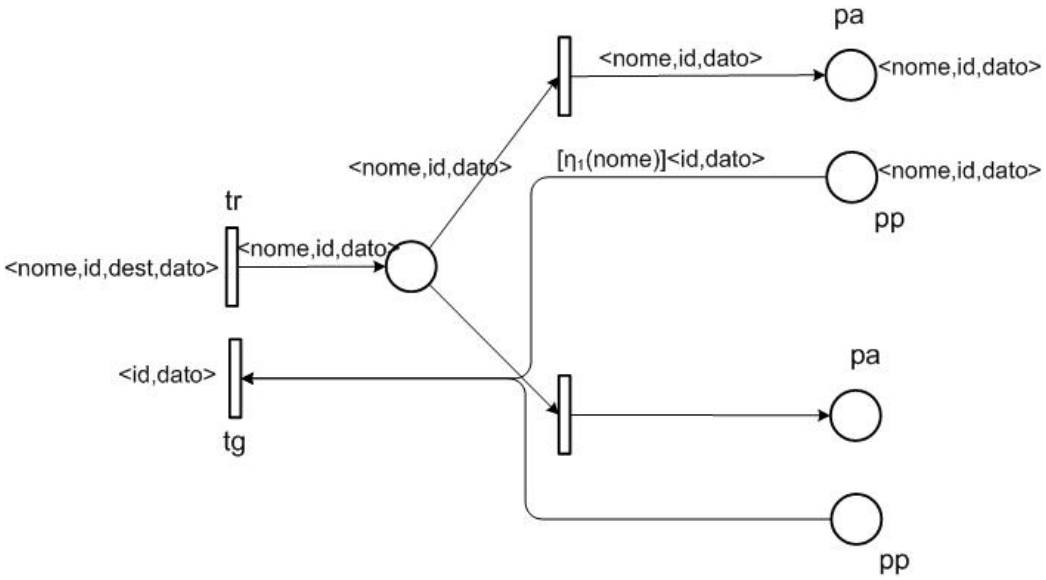

Figura 30: Modello SWN dell'operatore Anycast (client-server)

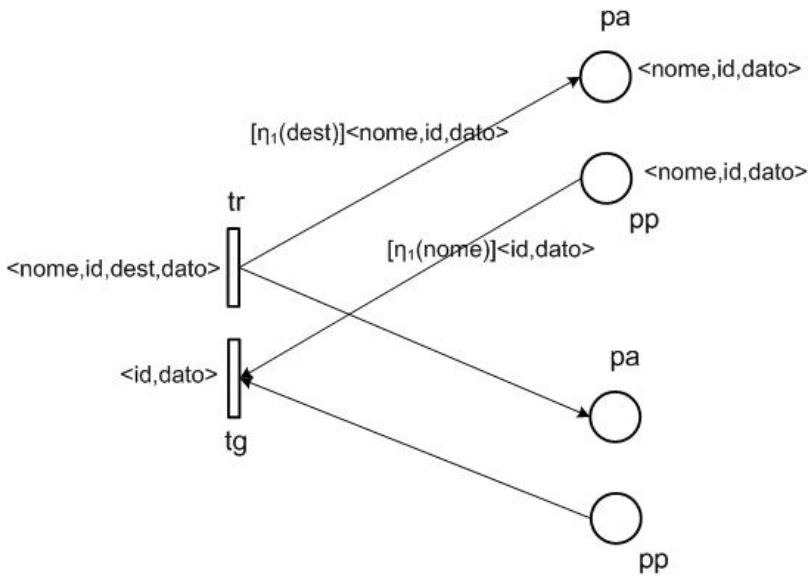

Figura 31: Modello SWN dell'operatore Deterministic Unicast (client-server)

Diamo evidenza, come fatto per il caso precedente, dello schema di principio della fusione tra le reti per modelli SWN (Figura 32).

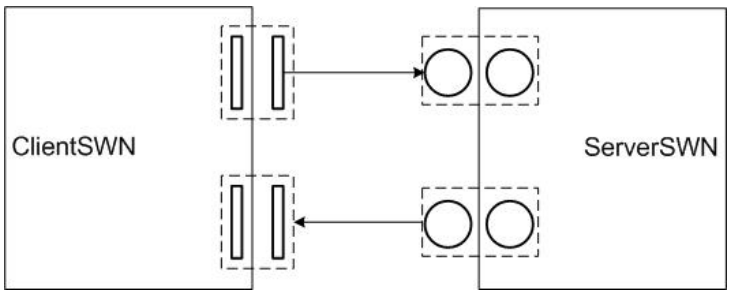

Figura 32: Composizione di modelli in ambito client-server SWN (operatore Simple Message)

Il Contact Center

Mostriamo adesso un'applicazione delle metodologie e dei concetti presentati in questo capitolo ad un caso di studio reale quale quello del Contact Center . I contact/call center sono oggigiorno

un modo efficace per le aziende di gestire le relazioni con i clienti: essi permettono la fornitura di servizi di customer care diversi tra loro ed utilizzando tecnologie informatiche (telefono, email, internet) eterogenee. Tali sistemi sono annoverabili tra i sistemi quality-critical e complessi. E' necessario dunque ottimizzare i costi e l'efficienza di un contact center attraverso la definizione delle giuste politiche di gestione e il giusto dimensionamento delle risorse.
Un modello del nostro sistema può essere dunque definito sulla base dei seguenti componenti:

- client (pool di clienti): agente attivo che chiama il contact center al fine di farsi erogare un servizio;
- site (singolo contact center): che risponde alle chiamate dei clienti smistando eventualmente su altri site un eccesso di carico temporaneo.

Modelleremo tale sistema attraverso le tecniche appena viste in quanto:

- le reti di Petri sono strumento prediletto per la valutazione delle prestazioni di sistemi distribuiti;
- le SWN riescono a modellare situazioni complesse ed analizzare reti di grosse dimensioni attraverso lo sfruttamento delle classi di equivalenza ed il compattamento dello spazio di stato attraverso tecniche simboliche , inoltre l'uso dei colori permette la modellazione sulla base della stessa struttura di rete di dinamiche complesse di gestione di tipi di clienti differenti attraverso politiche di QoS distinte;
- la buona definizione dell'interfaccia dei client e dei site rende agevole la modellazione di tali sistemi attraverso paradigmi ad oggetti;
- gli agenti sono immersi in un contesto distribuito, pertanto il paradigma di comunicazione è assimilabile a quello basato su scambio di messaggi;
- infine la possibilità di analizzare diverse configurazioni (in termini trasformazionali diverse semantiche) del modello globale a partire da modelli dei componenti scritti *una tantum*, impone la scelta composizionale e quindi l'uso dei diversi operatori di composizionalità.

L'approccio seguito è stato precedentemente pubblicato in .

Innanzitutto verranno chiariti, attraverso alcuni diagrammi di sequenza di UML le interazioni tra gli attori del sistema. Ciò verrà ottenuto attraverso la specifica del protocollo di interazione tra cliente e server C2SP e quello tra server e server (S2SP).
Descriveremo, per brevità, solo il protocollo C2SP, esso è illustrato in Figura 33, Figura 34, Figura 35 e Figura 36.

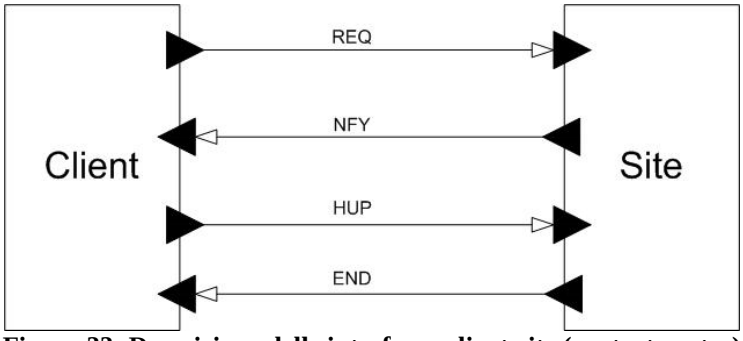

Figura 33: Descrizione delle interfacce client-site (contact center)

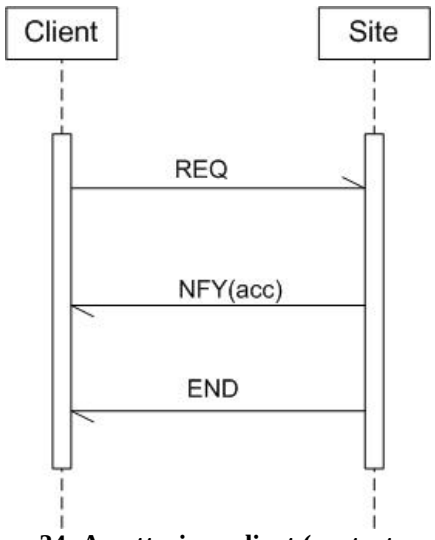

Figura 34: Accettazione client (contact center)

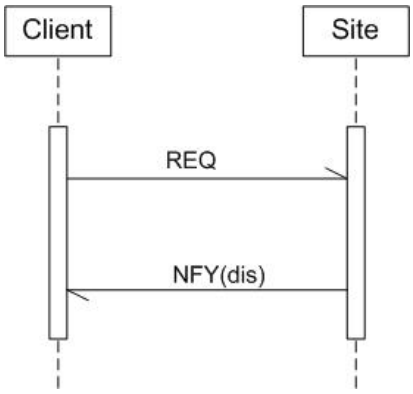

Figura 35: Rifiuto del client da parte del site (contact center)

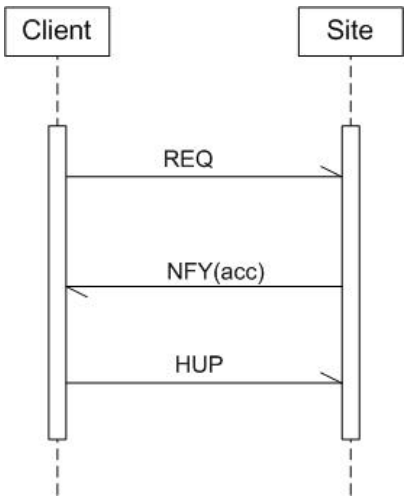

Figura 36: Riaggancio del client (contact center)

Due tipi di messaggi possono essere inviati dal client al server: *REQ* è usato dal client per iniziare una richiesta e *HUP* per riagganciare prima del termine del servizio (a causa di un'attesa troppo alta). Il server può rispondere in due modi diversi: *NFY* in cui comunica al cliente l'accettazione o il rifiuto della chiamata e *END* per terminare la comunicazione al termine dell'erogazione del servizio. E' possibile definire quindi due modelli ad oggetti SWN basati su paradigmi a scambio di messaggi. Tali modelli dei componenti verranno composti all'interno di due scenari: il primo utilizza semplicemente operatori di Simple Message (Figura 37), l'altro utilizza una versione più sofisticata dell'Anycast (illustrato in Figura 38).

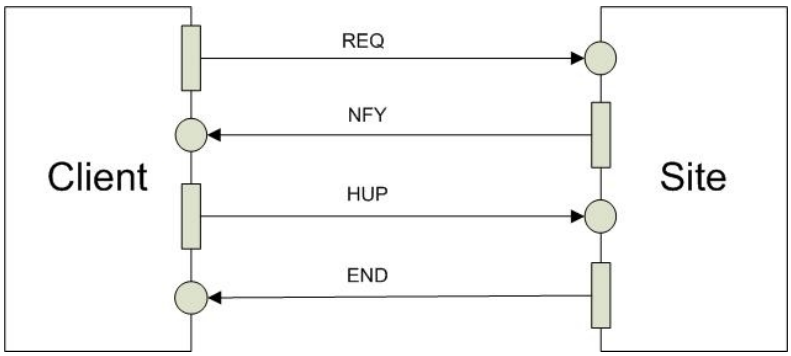

Figura 37: Applicazione del Simple Message al Contact Center

La particolarità di tale operatore rispetto a quelli finora presentati, è che implementa funzionalità di comunicazione complesse e, al fine di fornire tali funzionalità, necessita della definizione di uno stato.

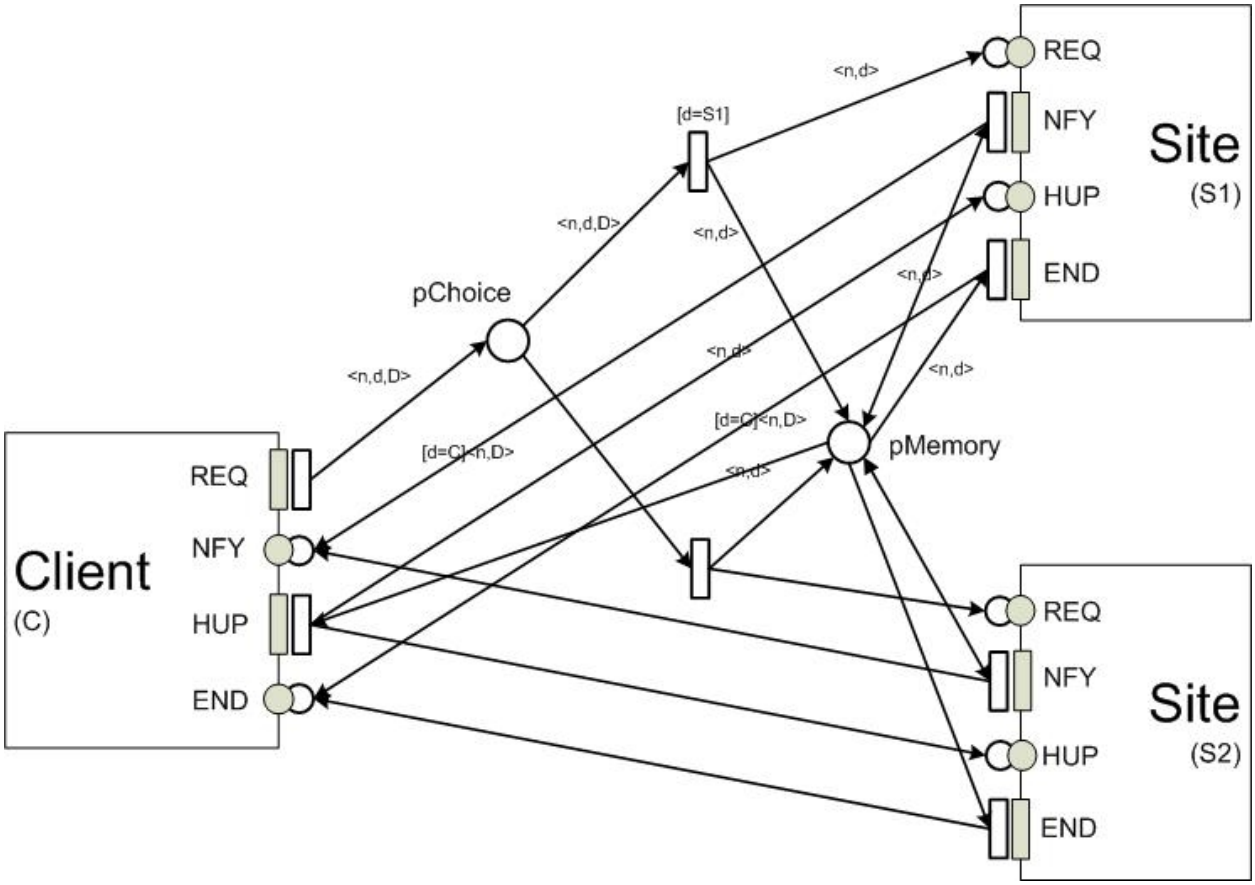

Figura 38: Operatore di Anycast con memoria applicato al Contact Center

Infatti dopo aver scelto uno qualsiasi dei Site cui il Client è collegato, l'operatore tiene memoria dell'associazione tra il Client ed il Site attraverso un token nel posto *pMemory* che rappresenta lo stato dell'operatore. Ciò permette, attraverso un feedback da tale posto verso l'interfaccia di hangOut, di inviare l'eventuale messaggio di HUP al site corretto. Volendo fare un paragone con il mondo delle reti di comunicazione, ciò corrisponde alla creazione di un "circuito virtuale" tipico ad esempio delle reti ATM. Quando la sessione di comunicazione viene interrotta, a causa di un HUP o di un END, lo stato relativo alla comunicazione viene cancellato.
In Figura 38 è stata scelta come convenzione la tripletta *<n,d,D>* come abbreviazione di *<nome,dest,dato>*. Inoltre non sono state esplicitate per semplicità le relazioni sugli archi relativamente alla comunicazione con il Site S2: esse sono, mutata mutandis, simili a quelle relative ad S1.

Capitolo VI:

Un caso di studio: il sistema ERTMS/ETCS

In questo capitolo si applicheranno le metodologie e le tecniche presentate al caso di studio del sistema di segnalamento ferroviario ERTMS/ETCS che costituisce un caso esemplare di sistema complesso e critico. Dopo una descrizione dell'architettura di riferimento del sistema e l'enunciazione dei requisiti RAMS cui il sistema è soggetto, si descriverà la metodologia di modellazione. Essa verrà basata sull'uso di metodi composizionali e multiformali (sia espliciti che impliciti); per tale architettura di riferimento si fornirà un modello formale del sistema e si provvederà alla sua analisi.

Descrizione del sistema

Il compito principale dello standard ERTMS/ETCS è quello di fornire le specifiche per la realizzazione di un sistema di segnalamento ferroviario che si propone di integrare le regole ed i sistemi nazionali esistenti di interlocking ferroviario con la necessità di interoperabilità tra realtà nazionali eterogenee .[22] D'altro canto l'interoperabilità è in questo contesto condizione necessaria alla creazione di prodotti e sistemi con elevate caratteristiche di performance, disponibilità e sicurezza.
Le funzionalità generali che il sistema deve portare a compimento sono raggruppabili nelle seguenti macrofunzionalità:

- gestione della via, il sistema deve garantire l'integrità della via in termini di corretto stato degli enti che gestiscono lo stato del tracciato (segnali, deviatoi, circuiti di binario);
- gestione del distanziamento, il sistema deve distanziare correttamente i treni e garantire che gli stessi abbiano traiettorie non intersecanti;
- protezione della marcia, il sistema deve garantire il rispetto dei limiti di velocità e di spazio percorso massimi in relazione alle diverse condizioni operative in cui esso si trova.

Tale standard è infatti alla base, nei diversi livelli di implementazione possibili, di progetti ferroviari importanti sia in Europa (Alta Velocità Italiana, Francia, Germania, Spagna, …) che extraeuropei (India). Questi diversi gradi di implementazione della specifica sono quantificabili nella caratterizzazione del sistema in livelli. Ci concentreremo sul livello 2 del sistema ERTMS/ ETCS che sta alla base del sistema Alta Velocità Italiana.
Descriviamo adesso l'architettura di riferimento del sistema ERTMS/ETCS (Figura 39) per il livello 2.

[22] Tale interoperabilità si manifesta nella possibilità di un sistema di bordo prodotto in un paese europeo di attraversare il confine con un altro paese, con regole movimentazione ferroviaria diverse, e di poter viaggiare all'interno del nuovo paese senza cambiare locomotore e senza nemmeno doversi fermare (un semplice esempio di tale situazione è il progetto di corridoi europei per il traffico ferroviario).

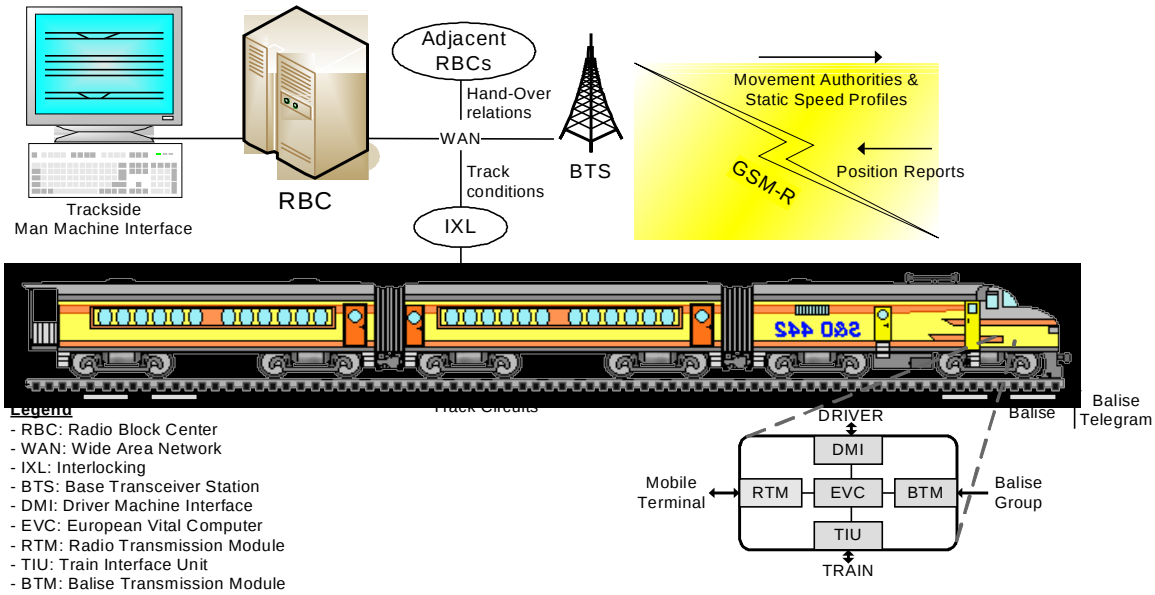

Figura 39: Architettura di riferimento per ERTMS/ETCS

Possiamo evidenziare alcuni elementi fondamentali costituenti il sistema tra loro interagenti:

- il *Radio Block Centre* (RBC): esso è il sottosistema di terra che comunica con il treno ed ha il compito di gestire correttamente il distanziamento tra i treni fornendo al treno stesso un'autorizzazione al movimento (Movement Authority - MA) che rappresenta l'inviluppo spazio-temporale in cui la marcia del treno può essere ritenuta sicura;
- il *SottoSistema di Bordo* (SSB): esso ha il compito di gestire, sulla base di quanto ricevuto dal RBC, la velocità e lo spazio massimo percorribile e di attuare, qualora tali requisiti di movimento non venissero rispettati, la frenatura del mezzo;
- l'*Interlocking* (IXL): è il responsabile dell'integrità della via e provvede a creare le condizioni di marcia (itinerari ed attuazione degli enti fisici della via). Al fine di conservare la compatibilità con i sistemi e le regole di segnalamento ferroviario nazionali, lo standard non copre tali sistemi ma unicamente il protocollo di comunicazione tra il RBC e gli IXL;
- la *rete di comunicazione* (GSM-R): i suoi compiti sono quelli di provvedere alla gestione della comunicazione tra il sottosistema di bordo e RBC attraverso l'adattamento di una normale rete GSM.

Oltre a tali sottosistemi, lo standard prevede altri componenti quali i Punti Informativi (PI) costituiti da gruppi di boe, antenne poste tra i binari che forniscono al treno informazioni di posizionamento assoluto (chilometrica), che permettono al SSB di comunicare al RBC l'esatta posizione del treno lungo il tracciato.
Data la non copertura degli IXL da parte dello standard, tali componenti non verranno analizzati, concentrandoci esclusivamente sui sottosistemi ERTMS.

Oltre ad una descrizione statica provvederemo anche a illustrare i meccanismi di consegna della MA da parte del RBC verso il SSB. Lo standard ERTMS/ETCS non descrive in modo mandatario la dinamica di tale interazione ma mette a disposizione dei fornitori di tecnologie alcuni strumenti utili nella determinazione di tali dinamiche. Il modello che seguiremo sarà basato sullo schema di principio qui descritto. Il RBC invia periodicamente, con periodo T_{MA}, una MA al SSB attraverso la rete GSM-R. Qualora il bordo non riceva tale messaggio per più di un certo periodo, $T_{NV\_CONTACT}$, ritiene di aver perso il collegamento con il sistema di terra e provvede ad una frenatura di servizio. Altra causa di tale frenatura è la perdita di connessione radio, l'instaurazione di una sessione di comunicazione GSM-R, che avviene per motivi legati alla disponibilità delle apparecchiature non vitali della rete in esame (BTS, MSC, Router, …). Il SSB d'altronde, prima di dichiarare la perdita di collegamento con il sottosistema di terra provvede a ricontattare il RBC per un numero di volte fissato. Qualora tutti i tentativi non vanno

a buon fine, esso è costretto a provvedere alla marcia degradata evidenziando così il fallimento del sistema di segnalamento ERTMS/ETCS.

Verifichiamo se questo sistema è da considerare come un ottimo esempio di sistema critico e complesso chiedendoci innanzitutto se esso rientra nella categoria di quelli complessi. In Tabella 8 è contenuta la risposta a tale domanda per ogni caratteristica dei sistemi complessi.

Tabella 8: Il sistema ERTMS/ETCS come sistema complesso

Caratteristiche	Motivazioni
agent-based	Il sistema è decomponibile in diversi sottosistemi (SSB, RBC, GSM-R, PI) ognuno dei quali attivo e proattivo
non-linearità	La marcia del treno sul binario è regolata da dinamiche fortemente non lineari soprattutto nella gestione della frenatura del veicolo
eterogeneità	I componenti (ed altri sottosistemi ed attori) sono molto eterogenei tra loro sia nella loro natura che nelle funzionalità (Operatore RBC, rete di comunicazione GSM-R, processo fisico di marcia del treno, sistemi di elaborazione)
dinamicità	Il comportamento del sistema è fortemente dipendente dalle condizioni a contorno (traffico ferroviario, condizioni operative, condizioni del tracciato …)
retroazione	Con riferimento ai sottosistemi RBC – SSB, il fatto che RBC invii informazioni ad EVC sulla base di quanto questi gli riferisce (e viceversa) è una dimostrazione di tale proprietà
organizzazione	I sottosistemi di terra (RBC, IXL) sono organizzati gerarchicamente con paradigmi e tecnologie di comunicazione a loro volta complesse
emergenza	La proprietà di emergenza è manifesta appena si pensa alle combinazioni che si possono ottenere dalle intersezioni di condizioni nominali (soprattutto di più treni) e di condizioni di degrado dei sistemi

Un’evidenza della criticità del sistema è ottenibile attraverso lo studio dei requisiti RAM cui il sistema stesso è soggetto . Tali requisiti regolano le prestazioni del sistema, soprattutto in termini di affidabilità e di disponibilità. Viene innanzitutto definito nello standard i vari tipi di fallimento cui il sistema può andare in contro:

- *fallimento immobilizzante* (immobilising failure – IF): tipo di guasto in cui il controllore di terra RBC perde il controllo di due o più treni;
- *fallimento di servizio* (service failure – SF): tipo di guasto in cui il controllore di terra RBC perde il controllo di al massimo un treno;
- *fallimento minore* (minor failure – MF): tutti gli altri tipi di fallimento o degrado.

Nella nostra analisi considereremo per semplicità, senza che però si perda di genericità nella nostra metodologia, solo i fallimenti immobilizzanti. E’ importante ai fini della nostra analisi specificare che i requisiti RAM vengono sviluppati in due diverse direzioni: la prima a livello

sistemico con requisiti generali su tutto il sistema di segnalamento, il secondo a livello di componente attraverso il presupposto che utilizzare componenti che soddisfino specifiche "locali" permetta al sistema globale di essere conforme al requisito.
In Tabella 9 sono specificati alcuni dei requisiti RAM di interesse nel corso della nostra analisi

Tabella 9: Requisiti RAM del sistema ERTMS/ETCS (Immobilising Failures)

Parametro	Descrizione	Requisito RAM
MTB_{ONB}	MTBF del sottosistema di bordo	$> 2.7*10^6$ h
$MTBF_{TRK}$	MTBF del sottosistema di terra	$> 3.5*10^8$ h
A_{IF-HW}	Disponibilità di sistema (IF)	> 0.999985
U_{RBC}	Indisponibilità del RBC	$< 10^{-6}$

Modellazione

Si sottolinea che i modelli ed i parametri utilizzati, nonché i risultati ottenuti dalla risoluzione dei modelli stessi, sono basati su considerazioni effettuate esclusivamente sull'architettura di riferimento ERTMS/ETCS. Sono stati utilizzati a tal fine soltanto documenti pubblicamente accessibili (specifiche UNISIG, data-sheets commerciali e pubblicazioni scientifiche) eliminando ogni riferimento a qualsiasi sistema fisico reale implementante tale standard e a qualsiasi prodotto esistente; ciò non pregiudica la validità della metodologia e permette di effettuare considerazioni di massima che devono essere, al fine della sua applicazione a sistemi reali, oggetto di ulteriori indagini scientifiche e tecnologiche.

Il nostro studio prevede innanzitutto l'individuazione, con riferimento ai fallimenti immobilizzanti, delle possibili cause di tale evento. Sono state individuate le seguenti:

- guasto hardware di almeno due SSB;
- guasto hardware di almeno un RBC;
- impossibilità della rete di comunicazione di consegnare le MA dal RBC a SSB.

Pertanto è stato definito un primo modello a livello sistemico che, attraverso un fault tree, individua questi contributi e relaziona il fallimento di sistema ai fallimenti dei singoli sottosistemi.
Tale albero di guasti è illustrato in Figura 40 e, data la semplicità del modello, non verrà ulteriormente commentato.

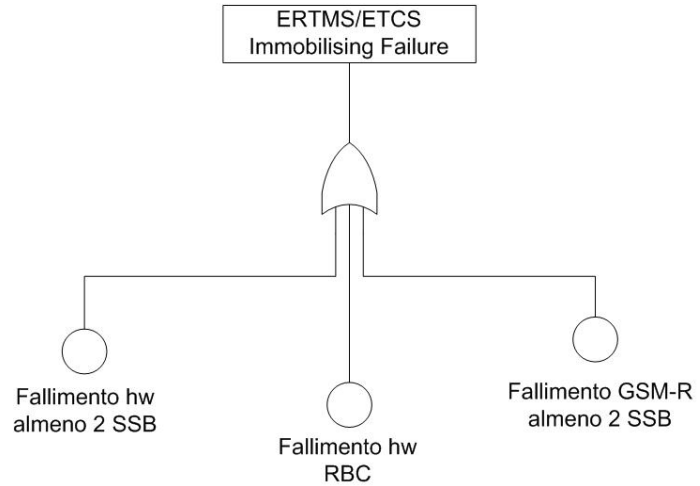

Figura 40: Modello per l'Immobilising Failure globale del sistema ERTMS/ETCS

Successivamente sono stati modellati i singoli sottosistemi attraverso una metodologia multiformale.

Il sottosistema di bordo

Descriviamo l'approccio di modellazione seguito per il sottosistema di bordo. Esso è a sua volta composto da diversi componenti attraverso una struttura gerarchica: tali componenti sono divisi in due categorie. Nella prima rientrano i componenti commerciali tipici di un sistema di elaborazione embedded:

- **TMR**: supponiamo che il centro delle elaborazioni del bordo sia costituito da un sistema TMR (Triple Redundant Module) composto da tre schede di calcolo con votazione a maggioranza 2oo3;
- **Voter:** un voter è un componente che confronta l'output di diverse cpu e decide, in base alla concordanza tra tali output, quale è l'output del sistema TMR e se ci sono schede di elaborazione in errore che devono essere escluse dall'elaborazione a seguito di errori;
- **BUS**: tipico bus di sistema di backplane che collega i diversi sottosistemi e permette lo scambio di dati tra gli stessi;
- **PS** (Power Supply): alimentazione dell'intero sistema di elaborazione.

Nella seconda categoria ci sono i componenti dedicati ERTMS/ETCS alla gestione di specifici compiti:

- **RTM** (Radio Terminal Module): comunicazione GSM-R con il RBC;
- **DMI** (Driver Machine Interface): interfacciamento input/output verso il macchinista;
- **BTM** (Balise Trasmission Module): ricezione dei telegrammi provenienti dai Punti Informativi;
- **TIU** (Train Interface Unit): interfacciamento verso le apparecchiature fisiche del treno (freni, taglio trazione, etc…).

L'ipotesi di base fatta su tale sistema è la sua non riparabilità on-line. Si suppone che, qualora il sistema di bordo si fermi, non sia possibile ripararlo seduta stante a causa della mancanza e del personale qualificato e dei pezzi di ricambio necessari. La riparazione di tali sistema deve essere perciò eseguito off-line, dopo il ricovero del mezzo in apposite strutture. E' stato scelto dunque il formalismo degli alberi di guasti per modellare guasti hardware per sistemi non riparabili. Il modello prodotto per tale sottosistema è illustrato in Figura 41.

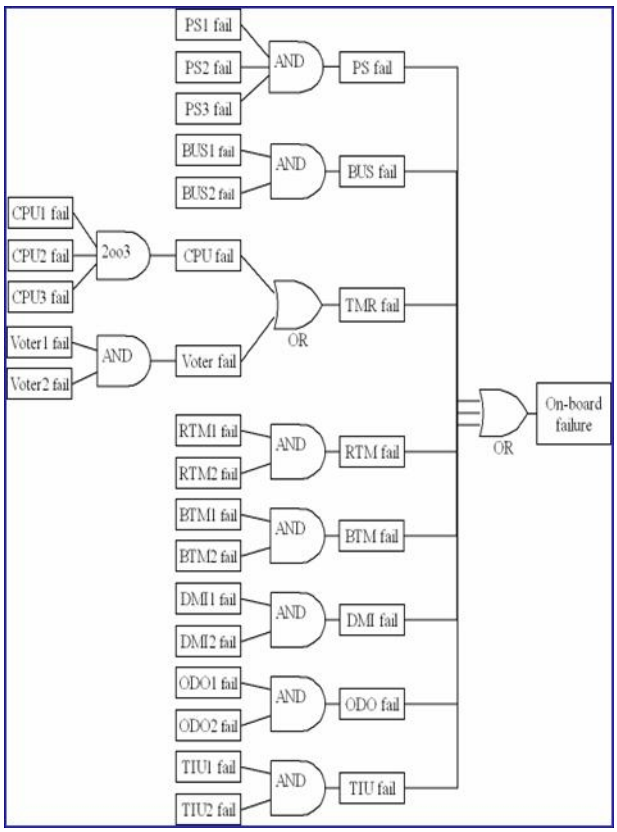

Figura 41: Albero dei guasti del sottosistema di bordo

I parametri scelti per caratterizzare i tassi di guasto dei componenti e quindi dei basic event dell'albero di guasto sono stati scelti con i seguenti criteri:

- per i componenti commerciali, è stato scelto un tasso di guasto medio di mercato estratto da data-sheets disponibili in rete;
- per i componenti ERTMS/ETCS sono stati scelti come riferimento i tassi di affidabilità indicati nei requisiti RAM per i singoli sottosistemi.

Essi sono riassunti in Tabella 10 ed in Tabella 11.

Tabella 10: Parametri MTBF/MTTR dei componenti COTS

Componente	MTBF [h]	MTTR [h]
CPU	$1.35 * 10^5$	0.0833
Bus	$2.25 * 10^5$	0.25
Voter (FPGA)	$3.33 * 10^8$	0.25
Power Supply	$1.35 * 10^5$	0.0833
GSM-R	$1.75 * 10^5$	0.25
WAN	$4 * 10^5$	0.25

Tabella 11: Parametri MTBF dei componenti ERTMS/ETCS

Componente	MTBF [h]
RTM	10^6
BTM	10^8
TIU	10^7
DMI	10^7

Il sottosistema di terra

Per quanto riguarda il sottosistema di terra, esso è composto dagli stessi componenti COTS del sottosistema di bordo. Inoltre sono presenti apparecchiature COTS che gestiscono

l’interfacciamento verso gli altri sottosistemi:

- **GSM-R**: scheda di interfaccia verso la rete GSM-R per la comunicazione con il sottosistema di bordo;
- **WAN**; scheda di interfaccia con rete di comunicazione fissa al fine di permettere ad RBC lo scambio di messaggi con gli IXL e con gli RBC adiacenti.

L’ipotesi fatta per tale sistema è che esso, data la localizzazione fissa di tale sistema in locali presidiati da personale specializzato e fornito di pezzi di ricambio, sia facilmente riparabile on line. E’ dunque necessario l’utilizzo del formalismo RFT per la sua modellazione ed analisi.
In Figura 42 è illustrato tale modello.

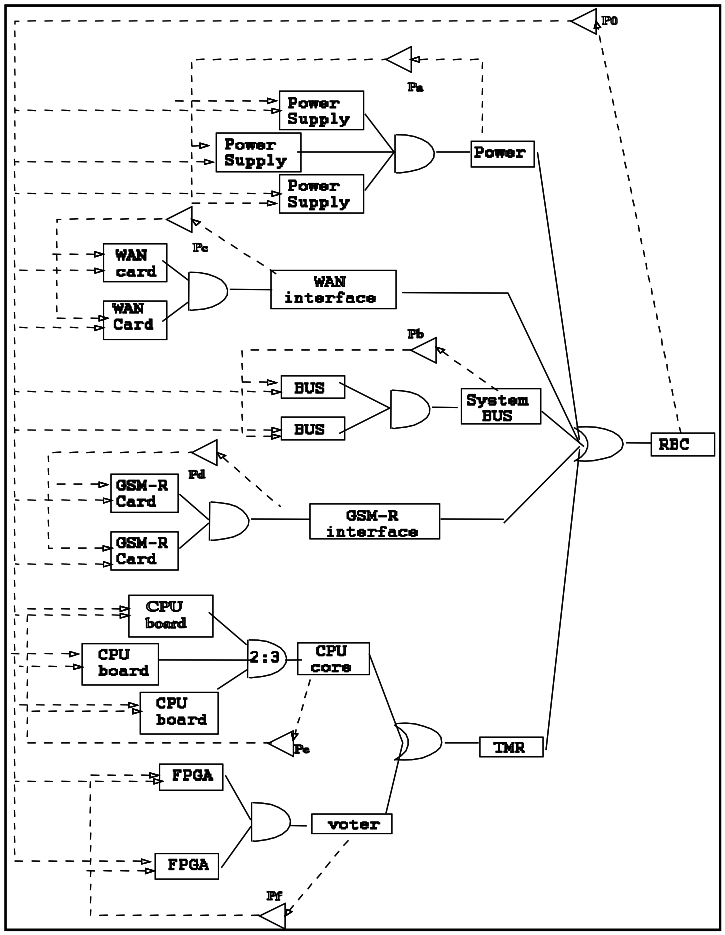

Figura 42: Modello RFT del RBC

Descriviamo brevemente in che modo gli RFT sono stati applicati ed in particolare l’uso di particolari Repair Box che estendono quanto già presentato e descritto in . In questo modello le RB sono state estese in modo da gestire le seguenti caratteristiche:

- tempi di riparazione non nulli: la RB impiega per le diverse riparazioni un MTTR diverso da zero;
- gestione delle risorse: la RB virtualizza diversi manutentori condivisi tra le varie RB: in altre parole qualora ci fosse un solo manutentore ed una RB venisse attivata mentre un’altra è in fase di riparazione, l’inizio della sua riparazione verrebbe posticipata;
- gestione delle priorità: i vari RB non hanno la stessa priorità, esiste una relazione d’ordine tra gli eventi da riparare in modo da massimizzare la funzione di disponibilità del sistema.

Una descrizione più dettagliata dello studio del sottosistema RBC e delle Repair Box utilizzate in questo studio è stata pubblicata in . In Figura 43 è illustrata la traduzione GSPN del modello RFT.

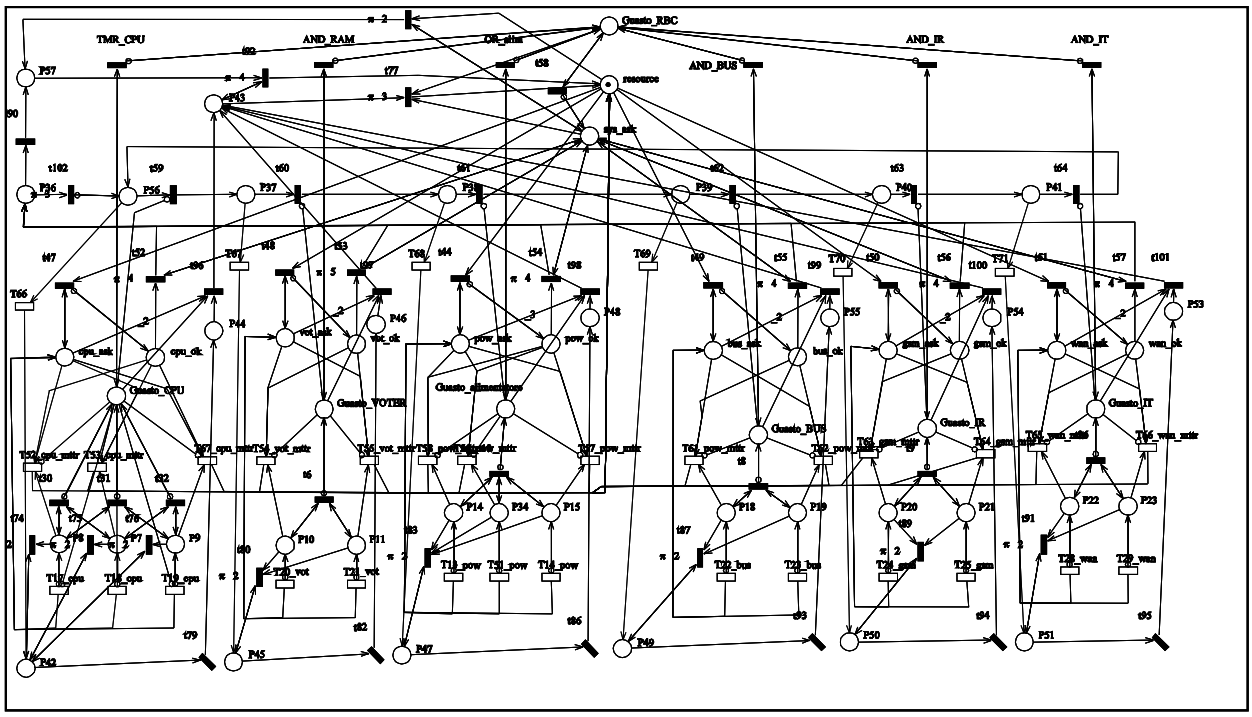

Figura 43: Modello GSPN del RBC

I parametri di tale modello sono illustrati in Tabella 10.

Il sottosistema di comunicazione

L'analisi delle performance della rete di comunicazione è stata effettuata modellando le interazioni tra RBC, la rete di comunicazione GSM-R e il SSB. Questi requisiti hanno richiesto dunque un formalismo quale quello delle reti di Petri Stocastiche Generalizzate al fine di poter modellare le caratteristiche di distribuzione dello stato, di concorrenza e di aleatorietà che tale modello di performance richiede.

Il modello segue lo schema di principio prima illustrato relativo alla consegna della MA al SSB da parte del sistema di comunicazione GSM-R e delle reazioni del SSB alla caduta di connessione o allo scadere di timer di vitalità. Sono state inoltre fatte due ulteriori ipotesi al modello: la prima riguarda l'indipendenza delle performance del sistema rispetto al numero dei treni. In altre parole supponiamo che ogni SSB abbia i suoi canali di comunicazione dedicati e che RBC sia un sistema real-time, ipotesi dunque più che soddisfatta, tale da garantire una schedulazione ciclica e con ciclo di schedulazione costante. La seconda ipotesi è che tutti i tempi siano distribuiti esponenzialmente: tale ipotesi non pregiudica la generalità della metodologia in quanto ha un impatto "locale" al modello GSPN e non sulle caratteristiche di multiformalismo o di composizionalità.

L'approccio seguito estende quanto descritto in con la modellazione aggiuntiva degli effetti di timeout sui sottosistemi SSB ed RBC. In Figura 44 è illustrata la rete GSPN in esame.

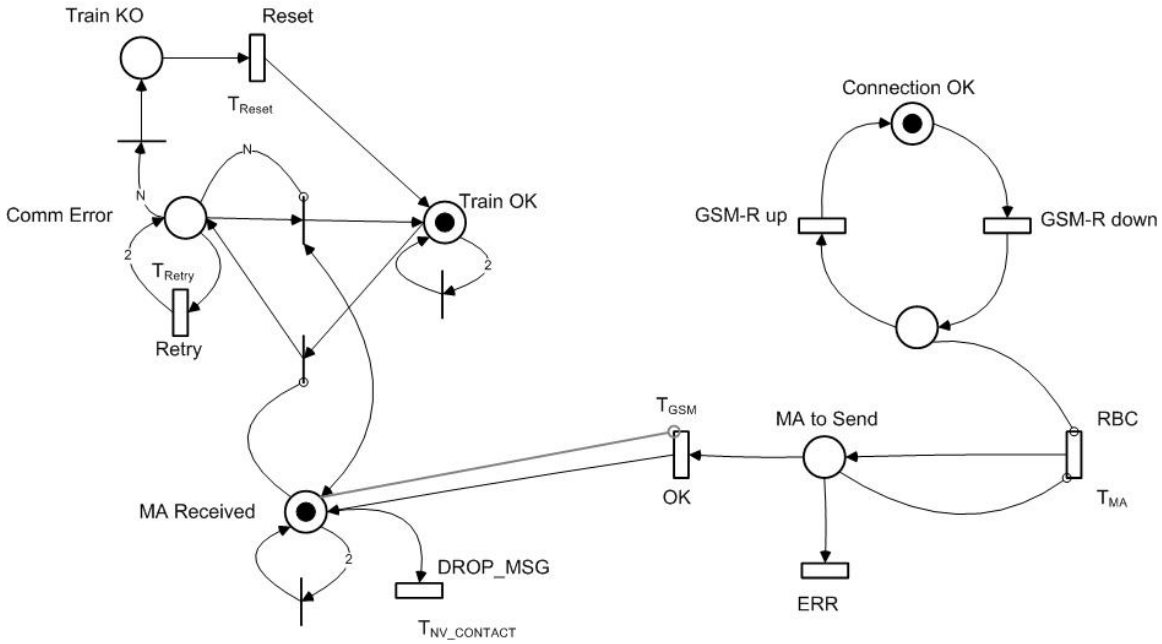

Figura 44: Modello GSPN della rete di comunicazione GSM-R

Evidenziamo le seguenti caratteristiche della rete: il posto *Connection OK* indica che la rete GSM-R è disponibile. Da tale stato è possibile uscire a seguito di un fallimento (transizione *GSM-R down*) verso lo stato di disconnessione dal quale, attraverso la transizione (*GSM-R up*) è possibile riportarsi nello stato di disponibilità. La transizione con *RBC* scatta ogniqualvolta RBC ricalcola un nuova MA e la manda al SSB (rate della transizione T_{MA}). Le transizioni *OK* ed *ERR* indicano il successo della trasmissione o la presenza di errori tali da invalidare il messaggio stesso: esse occorrono con probabilità 1-P_{ERR} e P_{ERR}. Il tempo che il messaggio impiega ad attraversare la rete nel caso positivo è T_{GSM}. La transizione *DROP_MSG* indica che il messaggio ricevuto da SSB viene cancellato in quanto scade la sua validità temporale. In questo caso lo stato del treno passa dal posto *Train OK* a *Comm Error*. In questo posto vendono effettuati N tentativi di riconnessione (transizione *Retry*) e solo dopo tale numero di tentativi senza che il sistema abbia ricevuto un nuovo messaggio, si deve procedere alla marcia senza supervisione: Da tale stato si può tornare nello stato iniziale solo dopo un tempo T_{Reset} oltre il quale vengono ripresi i tentativi di riconnessione.
In Tabella 12 sono descritti i parametri significativi rispetto al modello descritto.

Tabella 12: Parametri del modello GSPN del sottosistema di comunicazione GSM-R

Parametro	Descrizione	Valore nominale
Numero medio di token in Connection OK[23]	Disponibilità della connessione della rete GSM-R	0.9995
T_{MA}	Periodo di invio della MA da parte del RBC	2 sec
P_{ERR}	Probabilità di errore per un messaggio radio	10^{-4}
T_{GSM}	Tempo di attraversamento della rete GSM-R da parte del messaggio radio	200 msec
$T_{NV\_CONTACT}$	Tempo oltre il quale si dichiara muto il canale di comunicazione	7 sec
T_{RETRY}	Tempo di attesa per ritentare la vitalità del canale di comunicazione	30 sec

[23] Tale valore viene alterato agendo sui rate delle transizioni *GSM-R up* e *GSM_R down* in quanto si viene a creare un processo di nascita e morte markoviano tale che: $A = 1 - \frac{\lambda(GSMR\,down)}{\lambda(GSMR\,up)}$

N	Numero di tentativi	3
T_{RESET}	Tempo per il ripristino del canale di comunicazione	10 min

Tali parametri sono stati scelti in base alle indicazioni dello standard oppure a caratteristiche relative alla tecnologia utilizzata.

Una visione d'insieme

La Figura 45 visualizza il modello multiformale ed evidenzia come le tecniche di multiformalismo esplicito, multiformalismo implicito e composizionalità sono state utilizzate nella modellazione di un sistema complesso e critico.

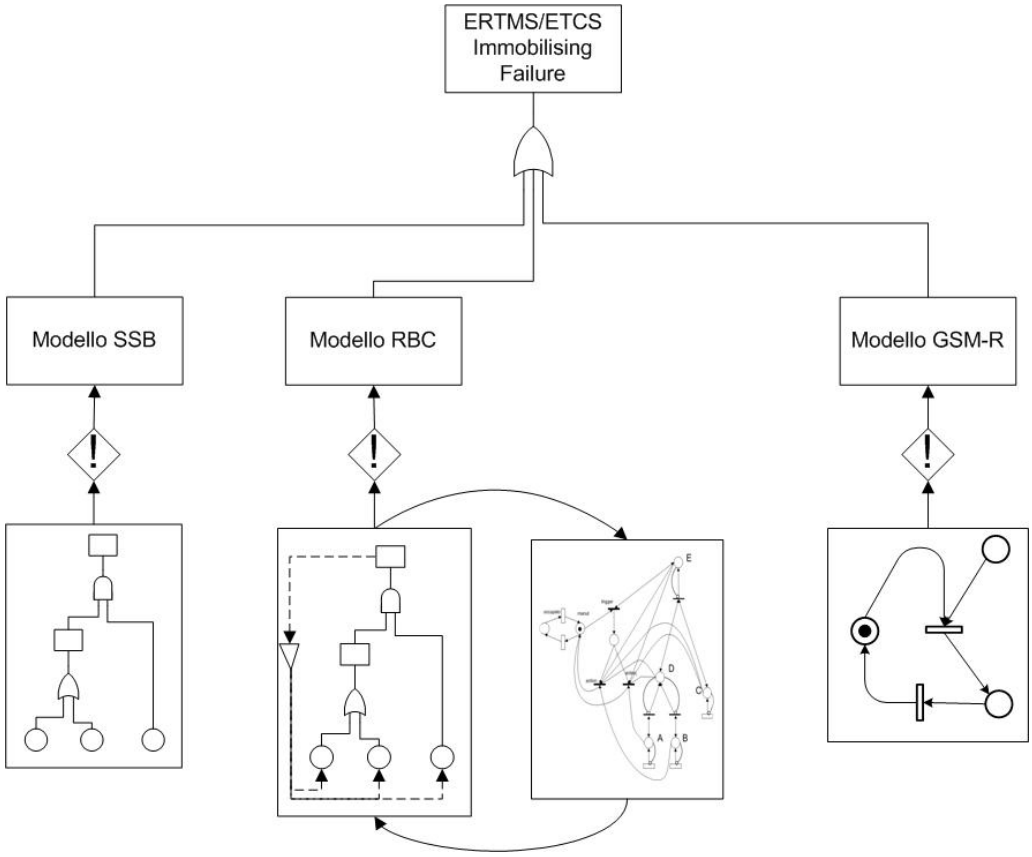

Figura 45: Visione d'insieme del modello ERTMS/ETCS

Tale figura evidenzia le caratteristiche di multiformalismo implicito, con l'utilizzo del modello RFT e la sua traslazione in linguaggio GSPN, il multiformalismo esplicito e di composizionalità, attraverso l'uso dell'operatore CER tra i seguenti attributi dei modelli:

- MTBF del top event del modello SSB per singolo treno → rate del basic event del modello globale;
- numero medio di token nel posto corrispondente al top event del RFT → rate del basic event del modello globale;
- numero medio di token nel posto relativo all'indisponibilità del sottosistema GSM-R per singolo treno → rate del basic event del modello globale.

Analisi e risultati

I sottomodelli sono stati analizzati singolarmente attraverso un'analisi di sensitività condotta nell'intorno dei parametri nominali scelti per il sottomodello; ciò ha permesso di individuare criticità "locali" ai sottosistemi e punti nello spazio dei parametri di compromesso tra affidabilità

dei componenti e risultato complessivo del sottosistema. Tali attività hanno permesso di effettuare, sulla base delle ipotesi fatte sui modelli e dichiarate nella sezione precedente, le seguenti considerazioni:

- sottosistema di bordo:
  - il sistema è poco sensibile a variazioni dell'affidabilità dei componenti ERTMS in quanto il vero collo di bottiglia per l'affidabilità è rappresentato dai componenti commerciali (in primo luogo dagli alimentatori);
  - in secondo luogo è stato notato, per l'architettura considerata, uno sbilanciamento dei requisiti RAM per i componenti ERTMS in quanto lo standard richiede per il componente BTM un MTBF pari a 10^8, per TIU e DMI MTBF pari a 10^7 e per RTM un MTBF pari a 10^6. Per la simmetria dell'architettura utilizzata è necessario avere un MTBF per il RTM di 10^7 e sufficiente un MTBF di BTM pari a 10^7;
- Radio Block Centre:
  - una prima analisi del sottosistema di terra mediante RFT ha portato alla considerazione che il numero delle risorse impegnate è significativo fino a 2 con un variazione significativa, due ordini di grandezza, nell'intervallo tra 0.5 a 1;[24]
  - l'effetto delle politiche di riparazione, le priorità assegnate agli eventi di guasto dei diversi componenti è quantitativamente poco significativo;
  - anche per quest'architettura la causa di guasto più probabile resta il fallimento di uno degli alimentatori;
  - l'affidabilità del sistema è una funzione lineare con i valori delle frequenze di riparazione (inverso del MTTR);
- rete GSM-R:
  - il sottosistema è molto rigido alle variazioni delle probabilità di errore della rete GSM-R (P_{ERR}). Una variazione in un range [10^{-6}; 10^{-2}] produce effetti molto scarsi sul risultato globale;
  - i parametri per cui la funzione dell'affidabilità di sottosistema è più sensibile sono quelli del T_{MA}, del numero di tentativi e dell'attesa tra un tentativo e l'altro;
  - anche con una delle migliori configurazioni parametriche, il risultato di affidabilità di tale sistema è basso.

Successivamente sono stati scelti tre valori significativi risultati dalle analisi dei sottomodelli (Tabella 13).

Tabella 13: Risultati sintetici dell'analisi dei sottomodelli

Sottosistema	Misura	Valore
SSB	MTBF	$6.457 * 10^4$
RBC	U	$5 * 10^{-5}$
GSM-R	U	$3.24 * 10^{-3}$

Tali valori sono stati usati come base per calcolare la probabilità di fallimento immobilizzante dell'intero sistema il cui modello, come già detto esprimibile mediante fault tree, è descritto nel dettaglio in Figura 46.

[24] Dire che per RBC sono allocate 0.5 riparatori significa dire che c'è una risorsa condivisa con altre realtà e che essa può essere considerata presente per RBC solo per il 50% del suo tempo.

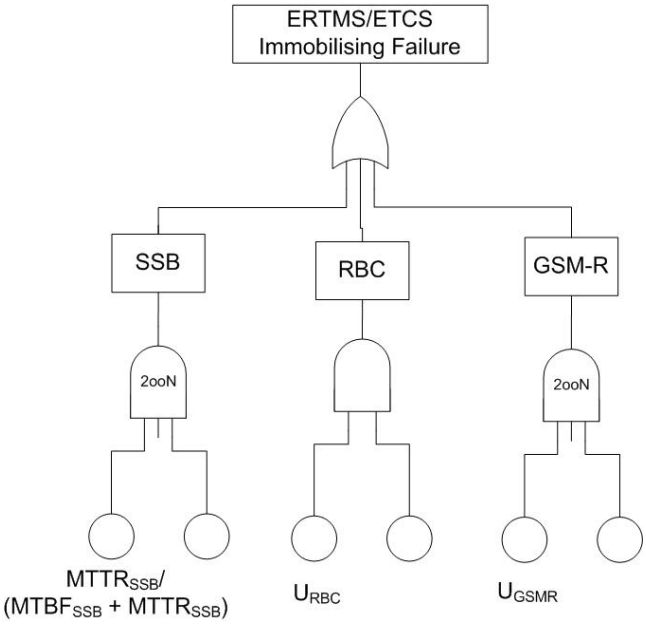

Figura 46: Modello FT globale del ERTMS/ETCS

Sono stati considerati ulteriori parametri di sistema (Tabella 14).

Tabella 14: Parametri del modello ERTMS/ETCS globale

Parametro	Valore
$MTTR_{SSB}$	1 h
N_{Treni}	3
N_{RBC}	2

Per l'architettura di riferimento considerata e per i parametri dei componenti scelti, l'analisi di sensitività su tale modello ha portato alle seguenti considerazioni:

- il requisito globale ERTMS/ETCS sull'immobilising failure, per i risultati ottenuti dalle analisi dei sottosistemi come da Tabella 13, non è soddisfatto;
- il collo di bottiglia per l'affidabilità di sistema resta la rete GSM-R.

Si prospetta come sviluppo possibile l'ottimizzazione automatica della funzione di affidabilità globale rispetto ai costi dettati dall'affidabilità dei singoli componenti.

Bibliografia

[1] W. Ross Ashby. *An Introduction to Cybernetics*. Chapman & Hall Ltd. London, 1957.

[2] A. Avizienis, J. C. Laprie, B. Randel, *Fundamental Concepts of Dependability*. Research Report N01145, LAAS-CNRS. April 2001.

[3] Y. Bar-Yam. *When Systems Engineering Fails --- Toward Complex Systems Engineering*. International Conference on Systems, Man & Cybernetics, 2003, Vol. 2, 2021- 2028, IEEE Press, Piscataway, NJ, 2003

[4] E.W. Dijkstra. *Notes On Structured Programming*. Structured Programming. Academic Press: London, (1972).

[5] A. Gammerman, V. Vovk. *Kolmogorov Complexity: Sources, Theory and Applications*. The Computer Journal, Vol. 42, No. 4, 1999.

[6] *CENELEC EN 50126: Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS).* 2001

[7] *CENELEC EN 50128: Railway Applications - Communication, signalling and processing systems - Software for railway control and protection systems.* 2001

[8] *CENELEC EN 50129: Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling*, 2003

[9] E.M. Clarke, J. M. Wing et al.. *Formal Methods: State of the Art and Future Directions.* ACM Computing Surveys. Vol 28-4: 626—643. 1996

[10] F. Wang. *Formal Verification of Timed Systems: A Survey and Perspective.* Proceedings of the IEEE, Vol. 92-8: 1283-1305. August 2004.

[11] International Electrotechnical Commission: *IEC 61508:2000, Parts 1-7, Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety-Related Systems*. 2000

[12] University of Michigan – Center of the Study of Complex Systems. Home page: http://www.cscs.umich.edu/old/complexity.html.

[13] D.C. Mikulecky. Virginia Commonwealth University. 2003. http://views.vcu.edu/~mikuleck/

[14] R. Gallagher, T. Appenzeller. *Beyond Reductionism*. Science 284, 79 (1999)

[15] *Engineering Safety Management Issue 3, Yellow Book 3* – RailTrack PLC. London, 2000

[16] J. Rushby. *Formal Methods and the Certification of Critical Systems.* Technical Report CSL-93-7. Computer Science Laboratory, SRI International. 1993

[17] C. Joslyn, F. Heylighen. *Welcome to Principia Cybernetica Web.* http://pespmc1.vub.ac.be/DEFAULT.html

[18] Andrew Bradley. Proceedings of the 7th Z User Meeting,London,, December 1992. ISBN 3-540-19818-0. 360.

[19] L. Lavagno, G. Martin, L. Scheffer. *Electronic Design Automation For Integrated Circuits Handbook.* CRC; Slipcase edition. April 13, 2006. ISBN 0-8493-3096-3

[20] E.M. Clarke, O Grumberg, D. A. Peled. *Model Checking*. MIT Press, 1999. ISBN: 978-0-262-03270-4

[21] P. Behm, P. Benoit, A. Faivre, and J.-M. Meynadier. *Meteor: A Successful Application of B in a Large Project*. In FM'99 - Formal Methods, LNCS 1708, pages 369--388. Springer-Verlag, 1999.

[22] G. Guiho, C. Hennebert. *SACEM software validation*. Proceedings of the 12th international conference on Software engineering. Nice, France – 1990. ISBN:0-89791-349-3

[23] S. Schneider. *The B-Method: An Introduction*. Palgrave. 2001. ISBN 0-333-79284-X

[24] J.C. Laprie. *Dependability: Basic Concepts and Terminology, Dependable Computing and Fault-Tolerance.* Springer-Verlag, Vienna, Austria. 1992.

[25] J.F. Meyer. *Performability: a retrospective and some pointers to the future.* Performance Evaluation, vol 14(3&4): 139–156. Elsevier. 1992.

[26] National Communication System Technology & Standard Division. *Federal Standard 1037C - Telecommunications,: Glossary of telecommunication terms.* General Services Administration Information Technology Service. 1996.

[27] A.Papoulis, S.U.Pillai. *Probability, Random Variables and Stochastic Processes.* 4th Ed. McGrawHill. 2002.

[28] Z. Har'El, R. Kurshan. *Software for Analytical Development of Communications Protocols.* AT&T - Technical Journal, pp. 45-59. Jan-Feb 1990.

[29] D.A Duffy. *Principles of Automated Theorem Proving.* John Wiley & Sons. 1991.

[30] S. Abramsky, A. Jung. *Domain theory.* In S. Abramsky, D. M. Gabbay, T. S. E. Maibaum, editors, Handbook of Logic in Computer Science, vol. III. Oxford University Press. 1994. ISBN 0-19-853762-X

[31] K.S. Trivedi. *Probability Theory Refresher*. Course Material. Spring 2006.

[32] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, *Improving the Analysis of Dependable Systems by Mapping Fault Trees into Bayesian Networks.* Reliability Engineering and System Safety Journal. Vol. 71-3:249-260. Elsevier. 2001

[33] E. Charniak, *Bayesian Networks without Tears*, AI Magazine, 1991

[34] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad. *Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications.* In IEEE Transactions on Computers, vol. 42, pp. 1343-1360. 1993

[35] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* J. Wiley and Sons ed., 1995.

[36] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F. Haasl. *Fault Tree Handbook. NUREG-0492.* Systems and Reliability Research Office of Nuclear Regulatory Research U.S. Nuclear Regulatory Commission, 1981

[37] T. Murata. *Petri-Nets: Properties, Analysis and Applications.* Proc. of the IEEE. Vol. 77(4):541-580. 1989.

[38] A. Bobbio, S. Bologna, E. Ciancamerla, G. Franceschinis, R. Gaeta, M. Minichino, L. Portinale. *Comparison of Methodologies for the Safety and Dependability Assessment of an Industrial Programmable Logic Controller*. In Proc. 12th European Safety and Reliability Conference (ESREL). Turin, Italy. September 2001.

[39] J. P. Bowen, M. G. Hinchey. *Ten Commandments of Formal Methods* IEEE Computer. Vol. 28(4):56-63. 1995

[40] S. Kent. *Model Driven Engineering*. In Integrated Formal Methods. Vol. 2335 of LNCS. Springer-Verlag, 2002

[41] M. Alanen, J. Lilius, I. Porres, D. Truscan. *Model Driven Engineering: A Position Paper*. In Proceedings of the 1st International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES). Turku Centre for Computer Science VOl. 29:25-29. 2004.

[42] S.J. Mellor, K. Scott, A. Uhl, D. Weise. *MDA Distilled. Principles of Model Driven Architecture.* Addison-Wesley. ISBN: 0201788918. 2004

[43] J. Bézivin, F. Jouault, D. Touzet. *Principles, Standards and Tools for Model Engineering.* In Proc. of 10th International Conference on Engineering of Complex Computer Systems (ICECCS). June 2005, Shanghai, China. IEEE Computer Society 2005. ISBN 0-7695-2284-X

[44] OMG website. www.omg.org

[45] H.L.M. Vanghekuwe, J. de Lara, P.J. Monsterman. *An Introduction to Multi-Paradigm Modelling and Simulation.* In Proc on the AI, Simulation and Planning in High Autonomy System Conf. (AIS 2002). Lisboa, Portugal. April 2002.

[46] G. Ciardo, R.L. Jones, A.S. Miner, R. Siminiceanu. *SMART: Stochastic Model Analyser for Reliability and Timing*. Tools of International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems. Aachen – Germany. Sept. 2001

[47] F. Bause, P. Bucholz, P. Kemper. *A Toolbox for Functional and Quantitative Analysis of DEDS.* Quantitative Evaluation of Computing and Communication Systems. LNCS 1469. Springer-Verlag. 1998.

[48] D. Deavours, G. Clark, T. Courteney, D. Daly, S. Derisavi, J.M. Doyle, W.H. Sanders, P.G. Webster. *The Möbius Framework and its Implementation*. IEEE Transactions on Software Engineeting, 28(10). 2002.

[49] K.S. Trivedi. *SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator*. In Proceeding of Dependable Systems and Networks. 2002. ISBN 0-7695-1597-5.

[50] J. de Lara, H. Vaunghelewe. *AToM^3: A Tool for Multi-formalism Modelling and Meta-modelling*. In Proc. Of the European Joint Conference on Theory and Practice of Software (ETAPS), Fundamental Approaches to Software Engineering (FASE). Sprinter-Verlag. April 2002. Granoble. France

[51] Jansen L., Meyer zu Horste M., Schneider E. *Technical issues in modelling the European Train Control System (ETCS) using coloured Petri nets and the Design/CPN tools*. In Proc. 1st CPN Workshop, DAIMI PB 532, pages 103--115. Aarhus University, 1998.

[52] L.H. Eriksson, K. Johansson. *Using formal methods for quality assurance of interlocking systems*. Computers in Railways IV, Computational Mechanics publications. 1998.

[53] D. Bjørne. *The FME Rail Annotated Rail Bibliography*. http://citeseer.ist.psu.edu/279277.html

[54] K. Jansen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Volume 1. Springer-Verlag. ISBN: 0387582762. 1995.

[55] R. Esposito, A. Sanseviero, A. Lazzaro, P. Marmo. *Formal Verification of ERTMS Euroradio Safety Critical Protocol*. In Proceedings of FORMS 2003, May 15-16, 2003, Budapest, Hungary.

[56] J. Cullyer, W. Wai. *Application of formal methods to railway signalling-a case study.* Computing & Control Engineering Journal, 4(1):15-22, Feb. 1993.

[57] T. Hlavatý, L. Preucil, P. Stepan, S. Klapka. *Formal Methods in Development and Testing of Safety-Critical Systems: Railway Interlocking System*. In Intelligent Methods for Quality Improvement in Industrial Practice. Prague: CTU FEE, Department of Cybernetics, The Gerstner Laboratory, 2002, vol. 1, p. 14-25. ISSN 1213-3000.

[58] C. Bernardeschi, A. Fantechi, S. Gnesi, S. La Rosa, G. Mongardi, D. Romano. *A Formal Verification Environment for Railway Signalling System Design.* Formal Methods In System Design. ISSN 0925-9856. Vol 12:139-162. 1998

[59] T. Kristoffersen, A. Moen, H.A. Hansen. *Extracting High-Level Information from Petri Nets: A Railroad Case.* Proceedings of the Estonian Academy of Physics and Mathematics, 2003, Volume 52, Number 4, pp. 378 - 393

[60] A. Moen Hagalisletto, J. Bjørk, I. Chieh Yu, P. Enger. *Constructing and Refining Large Scale Railroad Models Represented by Petri Nets.* IEEE Transactions on Systems, Man and Cybernetics, Part C.

[61] V. Vittorini, M. Iacono, N. Mazzocca, G. Franceschinis. *The OsMoSys approach to multiformalism modeling of systems.* In Journal of Software and Systems Modeling. Volume 3(1): 68-81. March 2004.

[62] G. Franceschinis, M. Gribaudo, M. Iacono, V. Vittorini, C. Bertoncello: *DrawNet++: a flexible framework for building dependability models*. In Proc. of International Conference on Dependable Systems and Networks, Washington DC, USA, June 2002

[63] F. Moscato, N. Mazzocca, V. Vittorini. *Workflow Principles Applied to Multi-Solution Analysis of Dependable Distributed Systems*. In Proceedings of 12th Euromico Conference on Parallel, Distributed and Network-Based Processing (PDP'04). 2004

[64] M. Gribaudo, M. Iacono, N. Mazzocca, V. Vittorini. *The OsMoSys/DrawNET Xe! Languages System: A Novel Infrastructure for Multiformalism Object-Oriented Modelling.* In Proc. 15th European Simulation Symposium and Exhibition. Delft, The Netherlands, October 2003

[65] F. Moscato. *Multisolution of Multiformalism Models: Formal Specification of the OsMoSys Framework*. PhD Thesis. Second University of Naples. October 2005.

[66] K. Czarnecki, S. Helsen. *Classification of Model Transformation Approaches*. OOPSLA 2003 Workshop on Generative Techniques in the Context of Model-Driven Architectures. 2003

[67] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Kuske, D. Plump, A. Schürr, G. Taentzer. *Graph Transformation for Specification and Programming.* Technical Report 7/96, Universität Bremen, 1996. http://citeseer.ist.psu.edu/andries96graph.html

[68] D. H. Akehurst, S.Kent. *A Relational Approach to Defining Transformations in a Metamodel.* In J.-M. Jézéquel, H. Hussmann, S. Cook (Eds.): UML 2002 – The Unified Modeling Language 5th International Conference. Dresden - Germany. Proceedings LNCS 2460:243-258, 2002.

[69] E. Visser. *A Survey of Strategies in Program Transformation Systems*. Electronic Notes in Theoretical Computer Science, eds. Gramlich and Lucas, vol. 57. Elsevier. 2001.

[70] M. Meyer zu Hörste. *Modelling and Simulation of Train Control Systems using Petri Nets*. FM'99 Formal Methods. World Congress on Formal Methods in the Development of Computing Systems. Lecture Notes in Computer Science 1709. Springer, 1999.

[71] *Design/CPN: Occurrence Graph Analyser-Manual*. Version 3.0, Aarhus, 1996.

[72] C.W. Janczura. *Modelling and Analysis of Railway Network Control Logic using Coloured Petri Nets.* PhD Thesis in Mathematics. University of South Australia. August 1998.

[73] I. Kurtev, J. Bézivin, M. Aksit. *Technological Spaces: an Initial Appraisal.* CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, USA. 2002

[74] R. Chianese, A. Lazzaro, P. Marmo, N. Mazzocca, D. Papa, V. Vittorini.. *An Experience in Railway interlocking Systems Specification and Formal Verification.* In Formal Methods for Railway Operation and Control Systems. pp.149-157. L'Harmattan Hongrie publisher. ISBN 963-9457-45-0. 2003.

[75] J. F. Meyer, A. Movaghar, W. H. Sanders, *Stochastic activity networks: Structure, behavior and application*. In International Conference on Timed Petri Nets, pp. 106--115, Torino, Italy, 1985.

[76] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambr. Univ. Press, 1996.

[77] A. Zimmermann, G. Hommel. *Toward Modeling and Evaluation of ETCS Real-Time Communication and Operation.*. Journal of Systems and Software archive Vol 77(1) Special issue: Parallel and distributed real-time systems. Pages: 47 – 54. ISSN:0164-1212. 2005. Elsevier Science Inc. New York, NY, USA.

[78] A. Chiappini, A. Cimatti, C. Porzia, G.Rotondo, R. Sebastiani, P.Traverso, A.Villafiorita. *Formal Verification of a Safety Critical Train Management System*. In Proceedings of the 18th International Conference on Computer Safety, Reliability and Security (SAFECOMP'99). Toulouse, FRANCE. September 1999. Lecture Notes in Computer Science series (LNCS) 1698, pages 410-419.

[79] OMG Group. *MOF QVT Final Adopted Specification.* OMG Document 05-11-01

[80] S. Sendall, R. Hauser, J. Koehler, J. Kuster, M. Wahler. *Understanding Model Transformation by Classification and Formalization.* Proceedings of Workshop on Software Transformation Systems (part of 3rd International Conference on Generative Programming and Component Engineering), Vancouver, Canada, October 2004.

[81] F. Jouault, I. Kurtev. *On the Architectural Alignment of ATL and QVT.* In Proceedings of ACM Symposium on Applied Computing (SAC 06), Model Transformation Track. Dijon (Bourgogne, FRA), April 2006

[82] Csertan, G., Huszerl, G., Majzik, I., Pap, Z., Pataricza, A., Varro, D.: *VIATRA - visual automated transformations for formal verification and validation of UML models.* In: Proc. 17th International Conference Automated Software Engineering, IEEE Computer Society (2002) 267–270

[83] A. Agrawal. *GReAT: A Metamodel Based Model Trasformation Language*. Automated Software Engineering 2003. 18th IEEE International Conference.

[84] L. Portinale, A. Bobbio. *Bayesian Network for Dependability Analysis: an Application to Digital Control Reliability.* In Proc. 15th Conference on Uncertainty in Articial Intelligence (1999).

[85] C. Lakos, S. Christensen. *A General Systematic Approach to Arc Extensions for Coloured Petri Nets.* In. Proceedings of 15th International Conference on Application and Theory of Petri Nets. Springer-Verlag, LNCS. 1994.

[86] J.W. Janneck, M. Naedele. *Introducing Design Patterns for Petri Nets.* TIK-Report Nr. 39 1998. http://citeseer.ist.psu.edu/janneck98introducing.html.

[87] D.C. Raiteri, G.Franceschinis, M.Iacono, V.Vittorini. *Repairable fault tree for automatic evaluation of repair policies.* In Proc. of the Performance and Dependability Symposium. July, 2004.

[88] A. Bobbio, G. Franceschinis, L. Portinale, R. Gaeta. *Exploiting Petri net to support fault-tree based dependability analysis.* In 8th International Workshop on Petri Nets and Performance Models (PNPM99), pages 146-155. IEEE Computer Society. 1999.

[89] G. Hura, J. Atwood. *The use of Petri nets to analyze coherent fault-tree.* IEEE Transaction on Reliability, 37. pages 194-203. 1985.

[90] M. Malhotra, K. S. Trivedi. *Dependability modeling using Petri nets.* IEEE Transaction on Reliability, 44. pages 428-440. 1995.

[91] M. Baker, T. E. Boult. *Pruning Bayesian networks for efficient computation.* In Proceedings of the 6th Conference on Uncertainty in Artifcial Intelligence. Cambridge, Massachuttes. Pag 257-264. July, 1990.

[92] S.L. Lauritzen, A.P. Dawid, .B.N.Larsen, H.G.Leimer. *Independence Properties of Directed Markov Fields.* Networks. Vol. 20:491-506. 1990

[93] D. Geiger, T. Verma, J. Pearl. *d-separation: From theorems to algorithms.* In Uncertainty in Artificial Intelligence. Vol 5:139-148. 1990.

[94] I.C. Rojas. *Compositional construction and analysis of Petri nets systems.* PhD Thesis. University of Edimburgh. 1997.

[95] D. Buchs, N. Guelfi. *CO-OPN: a concurrent object-oriented Petri nets approach.* Proceeding of the 12th International Conference on Application and Theory of Petri Nets. 1991.

[96] C. Lakos. *From Coloured Petri nets to Object Petri nets.* Proceedings of the Application and Theory of Petri Nets 1995. LNCS 935. pag 278—297. Springer-Verlag. 1995.

[97] C. Sibertin-Blanc. *A client-server protocol for the composition of Petri nets.* Proceeding of the 14th International Confernce on Application and Theory of Petri nets. LNCS. 1993

[98] C. Sibertin-Blanc. *Cooperative nets.* Proceeding of the 15th International Conference on Application and Theory of Petri nets. LNCS. 1994

[99] R. Montague. *Universal Grammar*. Theoria Vol .36:373-398. 1970.

[100] G. Frege. *Über Sinn und Bedeutung*. Zeitschrift für Philosopic und Philosophische Kritif, 100, 25-50. 1892 tradotto da M. Black come *On sense and reference*, in P. Geach, & M. Black (Eds.) Translations from the Philosophic Writings of Gottlob Frege, Basil Blackwell, Oxford, 1960, 56-78.

[101] G. R. Andrews. *Paradigms for Process Interaction Distributed Programs.* ACM Computing Surveys. Vol. 23(1):49-90. March 1991.

[102] G. Salaün, M. Allemand, C. Attiogbè. *Foundations for a combination of heterogeneous specification components.* FMCI 2002. http://citeseer.ist.psu.edu/664803.html

[103] Mehmet Aksit, Jean Bézivin, Ivan Kurtev. *A Global Perspective on Technological Spaces*. Software Engineering Group (TRESE) University of Twente The Netherlands.

[104] A. Staikopoulos, B. Bordbar. *A Metamodel Refinement Approach for Bridging Technical Spaces, a Case Study*. 4th Workshop in Software Model Engineering (WiSME 2005) - in conjunction with MoDELS/UML 2005

[105] D. Florescu, D. Kossmann. *Storing and Querying XML Data Using an RDBMS.* IEEE Data Eng. Bulletin 22, 1999

[106] BEA, Microsoft, IBM, SAP, Siebel. *BPEL: Business Process Execution Language for Web Services, Version 1.1*. (2003)

[107] M. Abadi, L. Lamport. *Composing Specifications*. ACM Transactions on Programming Languages and Systems, Vol 15, No. 1, January 1993.

[108] M. Iacono, S. Marrone, N. Mazzocca, F. Moscato, V.Vittorini, *Model Analysis of a Distributed Monitoring System using a Multi-Formalism Approach*. In Proc. 7th Workshop on State of the Art in Scientific Computing (PARA), Lyngby (Danemark) - June 2004. Revised Selected Papers, Springer-Verlag 2006, ISBN: 3-540-29067-2

[109] R. Alur, C. Courcoubetis, D.L. Dill. *Model Checking for real-time systems.* In Proc. of the 5th Annual Symposium on Logic in Computer Science. IEEE Computer Society Press, pp. 414-425, 1990.

[110] D.L. Dill. *Timing Assumption and verification of finite-state concurrent systems.* In Proc. of the International Workshop on Automatic Verification Methods for Finite State Systems. LNCS 407, pp. 197-212. Springer, 1989.

[111] P. M. Merlin. *A study of the recoverability of computing systems.* PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA, 1974.

[112] E. Best, A. Lavrov. *Generalised Composition Operations for High-level Petri Nets.* Fundamenta Informaticae. Vol 34. IOS Press. 2000.

[113] S. Marrone. *Modelli ad oggetti basati sulle reti di Petri per l'analisi di sistemi complessi.* Master's Thesis. Second University of Naples. 2002.

[114] G. Franceschinis, S. Marrone, N. Mazzocca, V. Vittorini, *SWN client-server composition operators in the OsMoSys framework*, In Proc. 10th International Workshop on Petri Nets and Performance Models (PNPM), Chicago (USA). IEEE – ISBN: 0-7695-1976-8. September 2003.

[115] G. Chiola, G.Franceschinis, R. Gaeta, M. Ribaudo. *GreatSPN 1.7: GRaphical Editor and Analyser for Timed and Stochastic Petri Nets.* Performance Evaluation special issue on performance modelling tools. Elsevier. 1995

[116] *GreatSPN 2.0.2 User's Manual*. Performance Evaluation Group. University of Turin.

[117] G. Franceschinis, M. Gribaudo, M. Iacono, S. Marrone, N. Mazzocca, V. Vittorini, *Compositional modeling of complex systems: contact center scenarios in OsMoSys*, In Proc. 25th International Conference on Application and Theory of Petri Nets (ATPN), Bologna (Italy). LNCS - ISBN: 3-540-22236-7. June 2004.

[118] G. Franceschinis, C. Bertoncello, G. Bruno, G. Lungo Vaschetti, A. Pigozzi. *SWN models of a contact center: a case study*. Proc. 9th Int. Workshop on Petri Nets and Performance Models. Aachen, Germany. September 2001. IEEE C.S. Press.

[119] UIC. *ERTMS/ETCS class1 System Requirements Specification. Ref. SUBSET-026*. issue 2.2.2,. 2002

[120] UNISIG. *ERTMS/ETCS RAMS Requirements Specification*, Ref. 96s1266.

[121] F. Flammini, M. Iacono, S. Marrone, N. Mazzocca. *Using Repairable Fault Trees for the evaluation of design choices for critical repairable systems*. In Proc. 9th High Assurance System Engineering (HASE), Heidelberg (Germany). IEEE - ISBN: 0-7695-2377-3. October 2005

[122] E. Battiston et al. *Modeling a cooperative development environment with CLOWN*. In Proceedings of the 1st OOPMC workshop (Torino, Italy, June). 1995

Appendice A:

Benefici dell'uso dei metodi formali per i sistemi critici

L'applicazione sistematica dei metodi formali in ambito industriale è limitata dai seguenti fattori:

- costi aggiuntivi e dilatazione dei tempi di sviluppo. A tale riguardo la Tabella 15 illustra la produttività[25] della British Aerospace rispetto ai diversi processi di sviluppo adottati :

Tabella 15: LoC/effort rispetto al processo utilizzato [17]

Processo di sviluppo	**Produttività (loc/m.a.)**
Codice non critico per la sicurezza	1400-1600
Codice critico per la sicurezza	700-800
Codice interamente sviluppato formalmente	200-400

- difficoltà nell'applicazione a problemi di grosse dimensioni;
- alta specializzazione richiesta nel loro uso.

Quindi i metodi formali, applicati correttamente e rigorosamente, impongono processi di sviluppo software tali da rallentarne la produttività. Questo svantaggio viene però ampiamente ricompensato se si allarga quest'analisi economica all'intero progetto. A supporto di tale affermazione citiamo due progetti industriali atti a dimostrare l'effettivo beneficio economico apportato nel lungo termine dai metodi formali.
In ambito ferroviario possiamo citare il recente progetto METEOR[26] (1998) relativo allo sviluppo della linea della metropolitana parigina n° 14 interamente automatica da parte di MATRA Transport International e Siemens. Tale progetto è il risultato di una lunga serie di applicazioni al segnalamento ferroviario in Francia dei metodi formali: capostipite di questa serie è stato il progetto SACEM relativo al periodo 1982-1989 di MATRA e GEC Alstom Transport. L'applicazione al progetto METEOR dei metodi formali è consistita nell'utilizzo di una metodologia di specifica e verifica del sistema in linguaggio B[27] . L'uso di tale metodologia su tale progetto (di dimensioni significative come evidenziato in Tabella 16), ha apportato i seguenti vantaggi:

- alcuni errori sono stati trovati durante le fasi di verifica formale;
- nessun errore è stato trovato durante le fasi di:
  - test funzionale su piattaforma ospite (general-purpose computer);
  - test funzionale su piattaforma target;
  - test in campo;
  - esercizio del sistema.

Tabella 16: Dimensioni del progetto METEOR

Software Product	*Lines of Ada*	*Instructions of Ada*	*Ada Packages*
Wayside	37.000	22.000	305
On-Board	30.000	20.000	160

[25] Misurata in termini di linee di codice prodotte su anni-uomo

[26] METro Est-Ouest Rapide

[27] B è una sigla che comprende una metodologia di sviluppo e di verifica formale (*B-Method*) e strumenti a supporto di tale metodologia (*B-Toolkit*). La metodologia è basata, per quanto riguarda il modello di specifica, su un metodo assiomatico basato su un formalismo logico-matematico: infatti B usa l'*Abstract Machine Notation* (AMN) sia per la modellazione delle specifiche che delle implementazioni; per tali modelli vengono specificate proprietà invarianti per lo stato e pre/post-condizionali per le operazioni. Il metodo di verifica si basa su theorem-proving.

Line	19.000	15.000	165
Total	*86.000*	*57.000*	*630*

Mentre il progetto METEOR è annoverabile nella categoria dei sistemi safety-critical, un altro esempio di applicazione industriale dei metodi formali è costituita da un sistema qualità critical: il progetto AUTO-TOPAS[28] della AT&T (1990) . Per questo progetto, dalle caratteristiche di difficoltà intrinseca (progetto innovativo) e dagli stringenti requisiti di disponibilità, è stato seguito un processo fortemente orientato all'uso di metodi formali sia per la specifica che per quanto riguarda la verifica. Esso è consistito in:

- specifica e verifica formale di alto livello (logica del protocollo);
- specifica e verifica formale incrementale dell'implementazione;
- specifica di basso livello (linguaggio C-like);
- verifica formale;
- generazione automatica del codice C.

In Tabella 17 sono descritte alcune caratteristiche del progetto come esse erano state stimate in fase di pianificazione e dopo la consegna del prodotto.

Tabella 17: Caratteristiche del progetto AUTO-TOPAS

Valore Metrica	Stimata	Effettiva
Sforzo (mesi/uomo)	12	1
Time-to-market (mesi)	9	5
Volume Sw (KLoC)	10	1.1

I metodi formali hanno dunque ridotto al 10% il volume di software sviluppato, ridotto i costi del 90% e ridotto i tempi di sviluppo di circa il 40%. Inoltre l'assenza di errori in fase di test di sistema e l'individuazione di alcuni errori critici nel progetto, hanno dimostrato l'applicabilità ai protocolli di comunicazione e all'hardware dei metodi formali diventando tecnologie standard AT&T.

In generale, considerati anche i tempi necessari all'apprendimento di tali metodologie ed all'integrazione all'interno dei processi aziendali preesistenti, i metodi formali dunque rappresentano costi aggiuntivi; tali costi si rivelano essere un investimento altamente proficuo in quanto permettono di migliorare la qualità del prodotto eliminando i costi dovuti a ritardi e/o alla bassa qualità. Inoltre migliorano decisamente i processi aziendali essendo i vantaggi appena enunciati, ripetibili per ogni progetto.

[28] Autonomous (AUTO) message protocol for an interface to AT&T product Trunk Operations Provisioning Administration System (TOPAS)