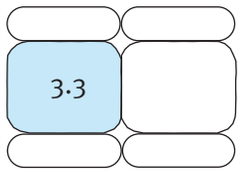
MODELLI PER L'ANALISI DI SISTEMI CRITICI

Francesco Flammini
Nicola Mazzocca
Valeria Vittorini

L'impiego di elaboratori in contesti diversi e sempre più delicati richiede di prestare particolare attenzione alle problematiche di sicurezza e affidabilità. In questo articolo, dopo una breve introduzione ai concetti base della *dependability* dei sistemi di elaborazione, verrà presentata una panoramica sulle tecniche modellistiche più diffuse per l'analisi di sistemi critici. Basandosi su vantaggi e limitazioni dei diversi approcci descritti, sarà introdotto il tema della modellazione multi-formalismo, riferendosi ad alcune delle prospettive di ricerca più interessanti in tale ambito.



1. INTRODUZIONE

L'evoluzione dei sistemi di elaborazione negli ultimi decenni ha portato alla nascita di calcolatori sempre più funzionali e performanti, ma non necessariamente più affidabili. In altre parole, all'aumento di complessità è corrisposta una riduzione di affidabilità. L'unica eccezione è costituita dal settore dei sistemi cosiddetti "*dependable*", a cui appartengono, per esempio, i sistemi di controllo in tempo reale impiegati in applicazioni critiche per la sicurezza di persone e beni materiali. La ragione di ciò è da ricercarsi nelle tecniche di sviluppo e verifica di tali sistemi, imposte da rigorosi standard internazionali. Tali tecniche fanno pesantemente uso di modelli al fine di dimostrare formalmente il rispetto dei requisiti qualitativi e quantitativi imposti dalla specifica. Nonostante ciò, anche tali sistemi non sono stati esenti da clamorosi fallimenti, il cui rischio aumenta rapidamente al crescere della complessità [14].
In questo articolo, dopo una breve introduzione ai concetti base della *dependability* dei sistemi di elaborazione, verrà presentata una panoramica sulle tecniche modellistiche più diffuse per l'analisi di sistemi *dependable*. Basandosi su vantaggi e limitazioni dei diversi approcci presentati, sarà introdotto il tema della modellazione multi-formalismo e dello sviluppo di modelli basato su componenti (che consente di governare sotto diversi aspetti la crescente complessità dei sistemi *dependable*), riferendosi anche ad alcune delle prospettive di ricerca più interessanti in tale ambito.

2. CONCETTI BASE E TECNICHE PER LO SVILUPPO DI SISTEMI *DEPENDABLE*

Una particolare classe di sistemi di elaborazione è quella dei sistemi cosiddetti "critici", ovvero tali che un fallimento nell'assolvere una o più delle funzioni ad essi richieste può comportare conseguenze di notevole gravità in termini di perdite economiche e/o di danni a persone o ambiente (Figura 1). Tra questi rientra-

no, in particolare, i sistemi cosiddetti in tempo reale (*real-time*) usati in applicazioni di controllo industriale, i quali devono reagire in tempi prestabiliti a stimoli esterni (riquadro 1).

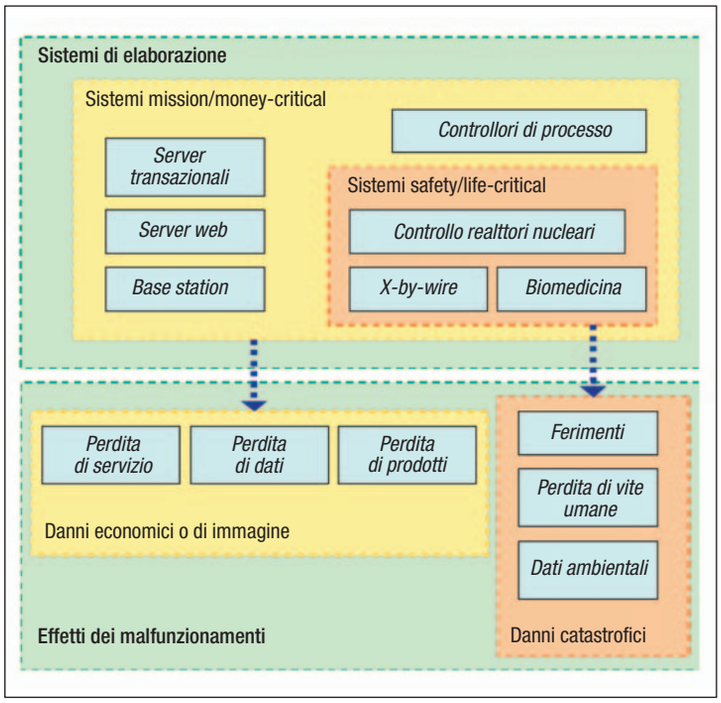


FIGURA 1

Classificazione di alcune tipologie di sistemi critici

Allo scopo di definire una tassonomia di riferimento non ambigua per lo studio di tali sistemi, si è introdotto in tempi relativamente recenti il concetto integrato di *dependability*. Secondo la definizione fornita da Avizienis et al. [1], un sistema può definirsi "*dependable*" se è possibile riporre fiducia nel suo funzionamento in modo giustificato. Chiaramente, questa definizione risulta piuttosto generica, dal momento che è possibile raggiungere diversi gradi di *dependability*, in funzione della particolare applicazione in cui il sistema andrà ad operare.

Un'esposizione sistematica della *dependability* può essere effettuata ricorrendo a tre concetti fondamentali: le minacce, gli attributi e gli strumenti (si faccia riferimento alla Figura 2) [1].

Gli attributi base di un sistema *dependable* sono: affidabilità, disponibilità, manutenibilità, innocuità (*safety*), integrità e riservatezza (*privacy*). A partire da questi, è possibile definire tutta una serie di attributi derivati: resilienza, plasticità, fidatezza (*trustworthiness*), sopravvivenza (*survivability*) ecc.. Tra questi è significativo citare la sicurezza (*security*), composta dagli attributi di disponibilità (per utenti autorizzati), integrità e riservatezza, e la performa-

RIQUADRO 1 - Sistemi *dependable* e sistemi critici

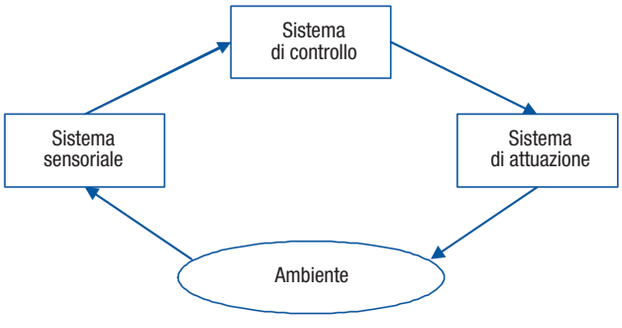
Una particolare classe di sistemi per i quali gli attributi di *dependability* assumono una rilevanza fondamentale è costituita dai sistemi cosiddetti "critici". Esistono diverse tipologie di sistemi critici (si veda la Figura 1):

- Sistemi *mission-critical* o *money-critical*, ovvero tali che un fallimento può comportare perdite notevoli in termini finanziari (*web server*, *base stations* ecc.).
- Sistemi *safety-critical* o *life-critical*, ovvero tali che un fallimento può comportare danni in termini di ferimenti o perdita di vite umane (macchine per dialisi, sistemi di guida automatica *X-by-wire* per autoveicoli, treni o aerei, controllo delle centrali nucleari ecc.).

Tipicamente un sistema *life-critical* è anche *money-critical*, ma ciò non sempre è vero, come nel caso dei sistemi biomedici. È invece vero che un sistema l*ife-critical*, dovendo reagire a stimoli esterni entro tempi rigidamente prestabiliti (Figura **A**), è anche di tipo *hard real-time*. Solo in linea teorica sarebbero classificabili come *life-critical* anche alcuni tipi di sistemi non di tipo *real-time* come quelli usati per il calcolo scientifico o strutturale: infatti, anche se i tempi di calcolo in tal caso non risultano rilevanti, risultati errati potrebbero ugualmente compromettere l'incolumità degli individui.

Per rendere rigorosa una tale classificazione si ricorre al concetto di "costo previsto del fallimento" (CP_M, a volte definito anche "rischio"), definito come segue:

$$CP_F = p_F \cdot C_F$$



A - *Schema teorico di un sistema di controllo*

Dove:
p_F è la probabilità di occorrenza del fallimento (esempio, numero di volte in un anno);
C_F è una stima delle conseguenze del fallimento (esempio, espresso in numero di vittime o danni monetari).

bilità (*performability*), ovvero la capacità di fornire prestazioni adeguate anche in presenza di uno o più malfunzionamenti.
Si noti che i due termini *safety* e *security*, entrambi traducibili in italiano come "sicurezza", esprimono concetti diversi. Il primo è relativo alla sicurezza in termini di incolumità di persone e ambiente, il secondo si riferisce alla sicurezza intesa in senso informatico[1].
Minacce alla *dependability* di un sistema sono guasti, errori e fallimenti. Un errore rappresenta l'alterazione dello stato del sistema causato da un guasto, mentre si ha un fallimento quando l'errore si propaga all'interfaccia del sistema (o del sotto-sistema), causando un'alterazione percettibile di uno o più servizi forniti. La distinzione tra errore e fallimento deriva dalla considerazione che in un sistema complesso l'errore può essere confinato ad uno o più componenti senza raggiungere l'interfaccia di servizio del sistema. Le minacce possono essere classificate in diversi modi, in funzione di origine (naturali, intenzionali), consistenza (riproducibili, elusive), persistenza (transitorie, intermittenti, permanenti), severità (minori, catastrofiche) ecc.. La figura 3 mostra un possibile diagramma di contesto per un sistema di controllo computerizzato, in cui sono evidenziate le entità esterne di diversa natura con cui il sistema interagisce e che possono essere sorgenti di guasti (errori di progettazione, sbalzi di tensione, surriscaldamento, errato utilizzo, manomissione ecc.).
Gli strumenti disponibili per assicurare la *dependability* consistono nel prevedere, preve-

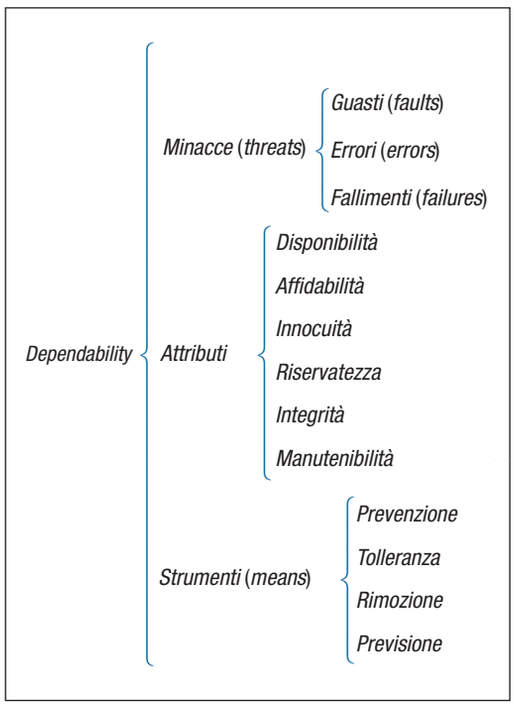


FIGURA 2
L'albero della dependability

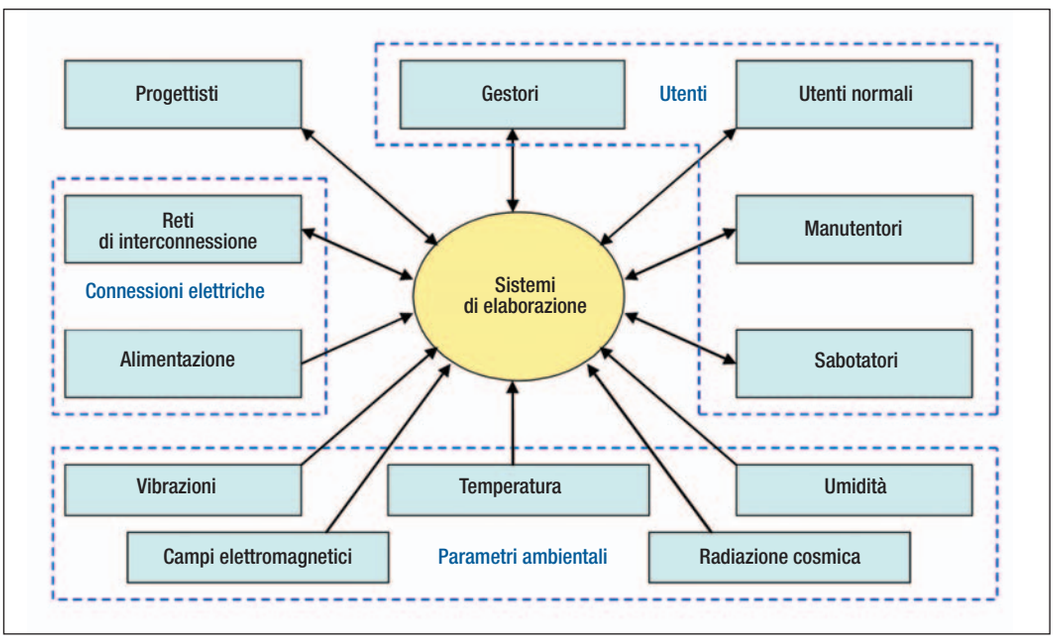


FIGURA 3
Diagramma di contesto di un sistema di controllo riferito alle sorgenti di guasti

[1] Nel contesto delle infrastrutture critiche, la parola *security* può anche essere intesa come protezione nei confronti di minacce esterne al sistema, di origine naturale o intenzionale.

RIQUADRO 2 - Tecniche più diffuse per lo sviluppo e l'analisi dei sistemi dependable

Tra le tecniche più diffuse e di successo in ambito industriale (*best practices*), è opportuno elencare le seguenti (trattasi di approcci complementari applicabili a diversi livelli di astrazione e in fasi distinte del ciclo di vita del sistema):

- Analisi degli azzardi (*Hazard Analysis*, HA) e dei modi di fallimento (*Failure Modes and Effects Analysis*, FMEA), che hanno lo scopo di individuare criticità del sistema ed opportuni meccanismi di protezione.
- Ispezioni tecniche della specifica (*Formal Technical Inspection*, FTI) per garantire non ambiguità, consistenza, completezza e coerenza dei requisiti.
- Iniezione dei guasti a livello hardware (*fault injection*), eseguita sul sistema reale o un suo modello simulato, allo scopo di valutare l'efficacia dei meccanismi di tolleranza ai guasti.
- Verifica del software statica (analisi, ispezione, *walk-trough*) o dinamica (*testing*), finalizzata ad individuare errori sistematici presenti nel codice prodotto.
- Tolleranza ai guasti basata su ridondanza spaziale (replicazione attiva o passiva), tra cui codici a correzione d'errore (*Error Correcting Codes*, ECC), architetture N-modulari (*N-Modular Redundancy*, NMR) con votazione a maggioranza sulle uscite di sezioni indipendenti (isolate e diversamente sviluppate), *watchdog timers*, *middleware* e protocolli robusti basati su cifratura, firma digitale, marcature temporali e numeri di sequenza.
- Ringiovanimento (*rejuvination*) e tolleranza ai guasti a livello software (*software fault tolerance*), per esempio, basata su ridondanza temporale realizzata con blocchi di recupero (*recovery blocks*).
- Programmazione difensiva, che consiste in una disciplina di codifica che contempla e tratta opportunamente condizioni d'errore e anomalie impreviste.
- Analisi dei ritorni dal campo ed elaborazione delle contromisure (*Failure Reporting Analysis & Corrective Action System*, FRACAS).

Nel caso dei sistemi critici, l'adozione di molte delle suddette tecniche è imposta da rigorosi standard internazionali (per esempio, il DO-178B impiegato in campo aeronautico) in funzione del livello di integrità della sicurezza (*Safety Integrity Level*, SIL) richiesto al sistema.

nire, tollerare e rimuovere le minacce ed i relativi effetti. Alcune delle *best practices* industriali utilizzate nelle diverse fasi del ciclo di sviluppo dei sistemi *dependable* sono riportate nel riquadro 2 (per approfondimenti si può far riferimento a [23]).

La tabella 1 riporta le suddette tecniche classificandole in funzione del ciclo di vita, del livello di astrazione (hardware o software) e delle tipologie di minacce che riescono a fronteggiare.

In particolare, in questo contesto ci concentreremo sulle tecniche di analisi basate su modelli formali. Tali modelli, rispetto agli approcci simulativi, consentono di valutare sia in termini qualitativi (proprietà del sistema) che quantitativi (indici numerici) gli attributi della *dependability* del sistema oggetto di analisi, allo scopo di:

- supportare la scelta dei parametri progettuali prevedendone l'effetto fin dalle prime fasi del ciclo di sviluppo;
- dimostrare il soddisfacimento dei requisiti degli standard RAMS[2] internazionali.

La verifica formale di proprietà è possibile ricorrendo a tecniche di *model-checking* e *theorem-proving* [18], mentre la valutazione di parametri quantitativi è in genere ottenuta attraverso l'analisi di modelli stocastici. Alcuni dei formalismi basati su grafi consentono

TABELLA 1
Metodi o strumenti utilizzati nell'ambito dei sistemi dependable

Metodo o strumento	Fase	HW/SW	Classi di minacce
Hazard analysis, FMEA	Specifica	Entrambi	Tutte
Analisi statica, ispezione codice e testing	Verifica	SW	Sistematiche
Fault Injection	Verifica	HW	Sistematiche
Meccanismi di Fault-Tolerance	Operativa	Entrambi	Casuali (tutte in caso di diversity)
Programmazione difensiva	Codifica	SW	HW casuali, SW sistematiche
Metodi formali	Specifica, verifica	Entrambi	Sistematiche
Modelli stocastici predittivi	Specifica, verifica	HW	Casuali
FRACAS, Measurement Based Analysis	Operativa	Entrambi	Tutte
Software Rejuvination	Operativa	SW	Casuali
Formal Technical Inspection	Specifica	Entrambi	Sistematiche

[2] RAMS: *Reliability, Availability, Maintainability, Safety* (Affidabilità, Disponibilità, Manutenibilità, Sicurezza).

entrambe le tipologie di analisi, come sarà meglio descritto nel paragrafo successivo.

3. FORMALISMI PER L'ANALISI DEI SISTEMI DEPENDABLE

I formalismi per lo studio dell'affidabilità dei sistemi *dependable* possono dividersi in due grandi categorie: quelli che seguono un approccio di modellazione top-down (o deduttivo) e quelli di tipo *bottom-up* (induttivi), a seconda che l'analisi parta dal fallimento a livello di sistema o da un guasto dei costituenti elementari. Il formalismo storicamente più diffuso per le analisi top-down è quello degli Alberi dei Guasti (*Fault Trees*, FT), mentre per le analisi *bottom-up* il formalismo di maggior successo è quello dei Diagrammi a Blocchi Affidabilistici (*Reliability Block Diagrams*, RBD). Entrambi questi formalismi si prestano ad analisi di tipo combinatoriale, che ne limita la potenza espressiva. Per esempio, RBD e FT non consentono la modellazione di tipi di guasto di modo comune, cioè guasti simultanei ad unità identiche causati dallo stesso motivo e nello stesso modo, e quindi tali da far cadere l'ipotesi di indipendenza tra le probabilità di guasto dei singoli costituenti.
Per ovviare a tali limitazioni, possono essere impiegati formalismi basati sull'analisi dello spazio di stato, quali catene di Markov a tempo continuo (*Continuous Time Markov Chains*, CTMC) e reti di Petri temporizzate o stocastiche (*Timed/Stochastic Petri Nets*, TPN/SPN). Tali formalismi consentono tra le altre cose una modellazione dettagliata degli aspetti di manutenibilità (diagnostica e recupero dagli errori).
Esistono diversi altri formalismi impiegabili per l'analisi dei sistemi *dependable*, tra cui gli automi temporizzati (*Timed Automata*, TA), i diagrammi di decisione binari (*Binary Decision Diagrams*, BDD) e le algebre dei processi (*Process Algebras*, PA), che possono risultare utili per l'analisi di particolari proprietà o attributi real-time. Inoltre, formalismi nati per scopi diversi sono risultati utili per l'analisi di certi aspetti dei sistemi *dependable*. Tra questi è opportuno citare le reti di code (*Queueing Networks*, QN), utili per modellare aspetti di *performability* in congiunzione con altri formalismi, e le reti bayesiane (*Bayesian Networks*, BN), che consentono di estendere la potenza espressiva degli alberi dei guasti senza incorrere nel problema dell'esplosione dello spazio di stato [3].
Infine, sono stati definiti dei formalismi derivati dai formalismi "base" sopra menzionati, aggiungendo varie estensioni e ricorrendo a metodi di soluzione spesso basati sulla traduzione dei modelli in formalismi con potenza espressiva più elevata. È il caso ad esempio degli alberi dei guasti, di cui sono state definite diverse estensioni (dinamici, parametrici, riparabili ecc.) [10].
Modelli espressi in taluni formalismi (per esempio, SPN) si prestano sia ad analisi di tipo quantitativo (esempio, tasso di fallimenti) che di verifica di proprietà (esempio, raggiungibilità di uno stato non sicuro). Altri (esempio, BN) consentono analisi qualitative sia di tipo "*what if*?", finalizzate a valutare quali sono le conseguenze provocate dal verificarsi di una determinata condizione, che di tipo "*backward*" (a ritroso) o di "*most probable explanation*" (spiegazione più plausibile), che hanno lo scopo di valutare le cause (esempio, guasti) a partire dagli effetti (esempio, fallimenti), e pertanto risultano particolarmente adatti ad analisi di sensitività parametrica o anche, in fase operativa, di tipo diagnostico.
Anche linguaggi per la specifica di sistemi, quali UML (*Unified Modeling Language*) e SysML (*Systems Modeling Language*), sono impiegabili per analisi formali di *dependability* [4].
In conclusione a questo rapido excursus, che non ha le pretese di essere esaustivo, è opportuno evidenziare come non esista un formalismo che sia in assoluto migliore degli altri. La scelta di adottare un linguaggio piuttosto che un altro va formulata in funzione di tutta una serie di fattori, tra cui la facilità d'uso, la verificabilità del modello, la potenza espressiva e, non ultima, l'efficienza degli algoritmi risolutivi (che potrebbe impedire, di fatto, la soluzione dei modelli di una certa complessità, tutt'altro che infrequenti nelle applicazioni reali). Pertanto, la scelta dei modelli va assolutamente calibrata in funzione degli "aspetti" da modellare (tipologia di componente, livello di astrazione ecc.) e degli obiettivi dell'analisi, ovvero di quali attributi è necessario valutare in corrispondenza di quali minacce [20]. Tipica-

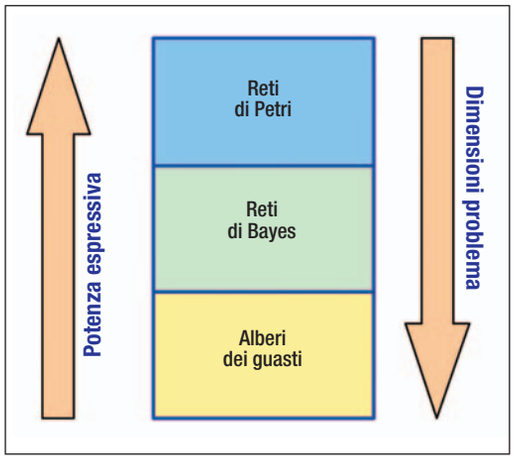


FIGURA 4
Scelta dei formalismi in base a potenza espressiva e dimensioni del problema

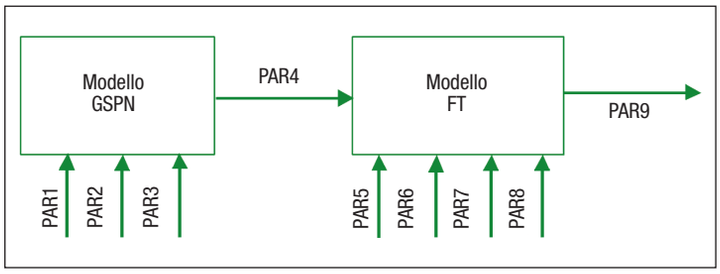


FIGURA 5
Un semplice esempio di composizione

mente, la scelta è dettata da un *trade-off* tra potenza espressiva ed efficienza: il rapporto reciproco tra questi due fattori è rappresentato dalla figura 4, che suggerisce, per esempio, di impiegare alberi dei guasti per sistemi di grandi dimensioni in cui non si abbiano esigenze particolari riguardo gli aspetti da modellare (ovvero si sia disposti a tollerare i limiti modellistici del formalismo).

4. NUOVI APPROCCI ALLA MODELLAZIONE

Si è visto nel precedente paragrafo come la modellazione della *dependability* richieda un compromesso tra diversi fattori, tra cui i più importanti sono la facilità d'uso degli strumenti, la potenza espressiva dei formalismi e l'efficienza dei risolutori. Detto in altre parole, sarebbe impensabile modellare un sistema complesso utilizzando formalismi di elevata potenza espressiva, come quelli basati sulle reti di Petri, perché anche se si riuscisse a gestire la difficoltà modellistica, si otterrebbe un modello impossibile da risolvere su qualsiasi elaboratore commerciale. D'altra parte, è noto come la valutazione degli attributi di *dependability* di un sistema richieda un approccio olistico. Si intuisce subito come l'unica soluzione soddisfacente al problema debba essere ricercata in tecniche modulari o composizionali in cui modelli eterogenei sono integrati in viste coese dell'intero sistema.
Una possibile strada per affrontare il problema consiste nell'adottare tecniche di modellazione **multi-formalismo**[3], il cui obiettivo è di supportare la rappresentazione di sistemi complessi combinando diversi linguaggi e tecnici di analisi. Un modello multi-formalismo è un modello costituito da parti, ciascuna delle quali modella un particolare componente o aspetto del sistema che si intende modellare, utilizzando il formalismo più adatto allo scopo. I diversi sotto-modelli contribuiscono alla creazione del modello complessivo che viene pertanto risolto coinvolgendo diverse tecniche e strumenti di analisi.
Prima di esaminare alcuni concetti fondamentali ed i diversi approcci attualmente utilizzati nell'ambito dello sviluppo di modelli multi-formalismo, può essere utile descrivere un semplice esempio. Supponiamo di volere analizzare un modello costituito da due sotto-modelli, rispettivamente realizzati mediante una Rete di Petri Stocastica Generalizzata (GSPN) e mediante un Albero dei Guasti (FT). Supponiamo che risolvendo il modello complessivo si voglia ottenere una misura della disponibilità del sistema modellato, ottenibile calcolando la probabilità di guasto del sistema data dal modello FT. Supponiamo infine che uno dei parametri necessari all'analisi dell'albero dei guasti (per esempio, la probabilità di occorrenza di un evento) sia fornita dall'analisi del modello GSPN. Si tratta dunque di un semplice caso di composizione sequenziale descritta nella figura 5, dove i due sotto-modelli sono graficamente rappresentati da rettangoli, mentre i dettagli relativi alla loro struttura sono nascosti. In figura sono mostrate solamente le informazioni quantitative che devono essere fornite per popolare il modello e poterlo utilizzare (da PAR1 a PAR3 per il sotto-modello GSPN e

[3] La modellazione multi-formalismo è un aspetto della modellazione multi-paradigma, che comprende anche i concetti di astrazione, meta-modellazione e trasformazione di modelli [19].

da PAR5 a PAR8 per il sotto-modello FT), le informazioni che devono essere prodotte dall'analisi della rete GSPN (PAR4) ed il valore atteso dalla risoluzione del modello complessivo (PAR9). Questo modello può essere risolto utilizzando diversi strumenti di analisi, per esempio Sharpe [21].

Il modello di figura 5 è un modello multi-formalismo, costruito componendo esplicitamente sotto-modelli espressi mediante linguaggi di modellazione differenti. La composizione è realizzata in generale mediante operatori più o meno sofisticati ed è un punto cruciale nella definizione di metodologie di modellazione avanzate. Gli operatori di composizione consentono la condivisione di *stati*, *azioni* o *eventi* tra modelli diversi. Nel caso particolare in cui essi si limitano allo scambio di risultati tra modelli risolti in modo isolato (come nel caso dell'esempio) si parla di "connettori".

Affinché una composizione sia corretta è necessario in generale che:

- il modello composto verifichi determinate proprietà e che garantisca la conservazione di proprietà già verificate dai sotto-modelli (per esempio, proprietà strutturali come l'assenza di situazioni di *deadlock*);
- l'operazione sia semanticamente valida, e cioè che abbia senso l'interazione tra diversi formalismi come definita attraverso la composizione (per esempio, deve aver senso utilizzare la frequenza di scatto di una transizione di una rete di Petri per valutare la probabilità di occorrenza di un evento in un albero dei guasti);
- siano definite opportunamente la sequenza con cui i diversi sotto-modelli devono entrare in gioco nella risoluzione del modello complessivo (per esempio, il modello GSPN deve essere risolto prima del modello FT) e le azioni da effettuare (per esempio, valutare il *throughput* di una transizione del modello GSPN, calcolarne l'inverso, usare questo dato come valore di un parametro del modello FT).

Una volta che i precedenti aspetti sono stati affrontati e risolti (non esiste un metodo generale, queste problematiche vanno affrontate a partire dagli specifici formalismi coinvolti), la composizione è uno strumento molto potente perché consente di introdurre nell'ambito della modellazione di sistemi complessi tecniche simili a quelle utilizzate nel-

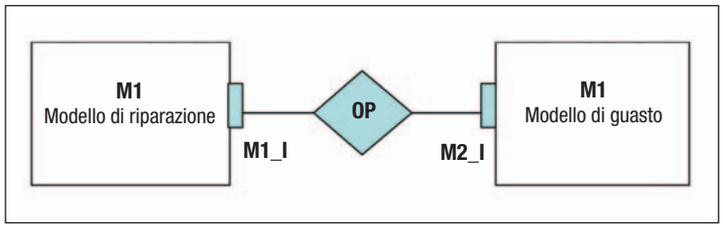


FIGURA 6
Un modello composto

l'ingegneria del software nello sviluppo per componenti.

Nella figura 6 è mostrata una diversa rappresentazione del modello descritto nella figura 5: i due sotto-modelli sono visti come scatole nere (componenti) M1 ed M2, che esportano un'interfaccia (M1_I e M2_I) e forniscono un comportamento noto (che cosa modellano); il connettore/operatore è esplicitato (il rombo etichettato OP); gli archi connettono l'operatore agli "operandi" cui si applicano. L'interfaccia esplicita le informazioni coinvolte ed eventuali vincoli che devono essere rispettati. L'operatore incapsula al suo interno le azioni da compiere e le informazioni circa l'ordine con cui tali azioni devono essere svolte. Nel caso dell'esempio, con riferimento ad un sistema (o ad una sua parte), M1 modella un'azione di riparazione che influenza la probabilità di guasto di uno dei componenti del sistema ed M2 modella le combinazioni di eventi di guasto che possono condurre all'indisponibilità del sistema.

A questo punto è possibile pensare di sostituire, per esempio, il sotto-modello M1 con un altro sotto-modello M espresso mediante un differente formalismo (o un modello espresso mediante lo stesso formalismo ma ad un diverso livello di astrazione), a condizione che quest'ultimo:

a. fornisca un comportamento equivalente (M modella lo stesso aspetto del sistema modellato da M1);

b. rispetti l'interfaccia specificata.

La definizione degli operatori di composizione, delle interfacce ed i problemi relativi alla loro tipizzazione sono un aspetto estremamente delicato e sono un campo aperto di ricerca [17]. L'interesse della comunità scientifica per questo approccio alla modellazione è dettata dai vantaggi che esso comporta: promuove il riuso e la definizione di librerie di

modelli, e quindi lo sviluppo di strumenti che in parte alleviano la difficoltà legata allo sviluppo di modelli complessi; introduce un certo grado di flessibilità e dinamicità nella definizione del modello (si potrebbe definire quale sotto-modello utilizzare per realizzare una parte del modello complessivo sulla base, ad esempio, di valori forniti dalla risoluzione di altre parti); supporta efficacemente metodologie di tipo *divide-et-impera* e la definizione di modelli gerarchici.

A titolo di esempio, la figura 7 mostra un possibile modello multi-formalismo di un complesso sistema di controllo ferroviario. Il sistema preso per esempio deriva dallo standard europeo per le nuove linee ferroviarie interoperabili ad alta velocità. Si tratta di un sistema geograficamente distribuito, in cui si distinguono la parte installata nelle cabine di guida dei treni (*on-board*), quella di terra distribuita lungo linea (*lineside*) e quella di terra concentrata nei centri di controllo (*trackside*). Il modello globale è realizzato attraverso una composizione multi-livello di modelli eterogenei relativi a componenti (o sotto-sistemi) distinti. I modelli al livello di astrazione inferiore rappresentano aspetti hardware, che si modellano tipicamente ricorrendo ad alberi dei guasti (e relative estensioni) e/o reti bayesiane. I modelli al livello di astrazione intermedio rappresentano le diverse modalità operative (marcia a vista, supervisione completa ecc.), che sono legate alla disponibilità dei sottosistemi hardware e a loro volta condizionano le funzionalità del sistema a più alto livello; questi aspetti possono essere modellati agevolmente attraverso automi a stati finiti (o formalismi analoghi). Infine, i livelli più alti rappresentano le procedure operative (es. inizio e fine missione), che richiedono formalismi adatti a modellare aspetti comportamentali, quali per esempio reti di Petri e relative estensioni.

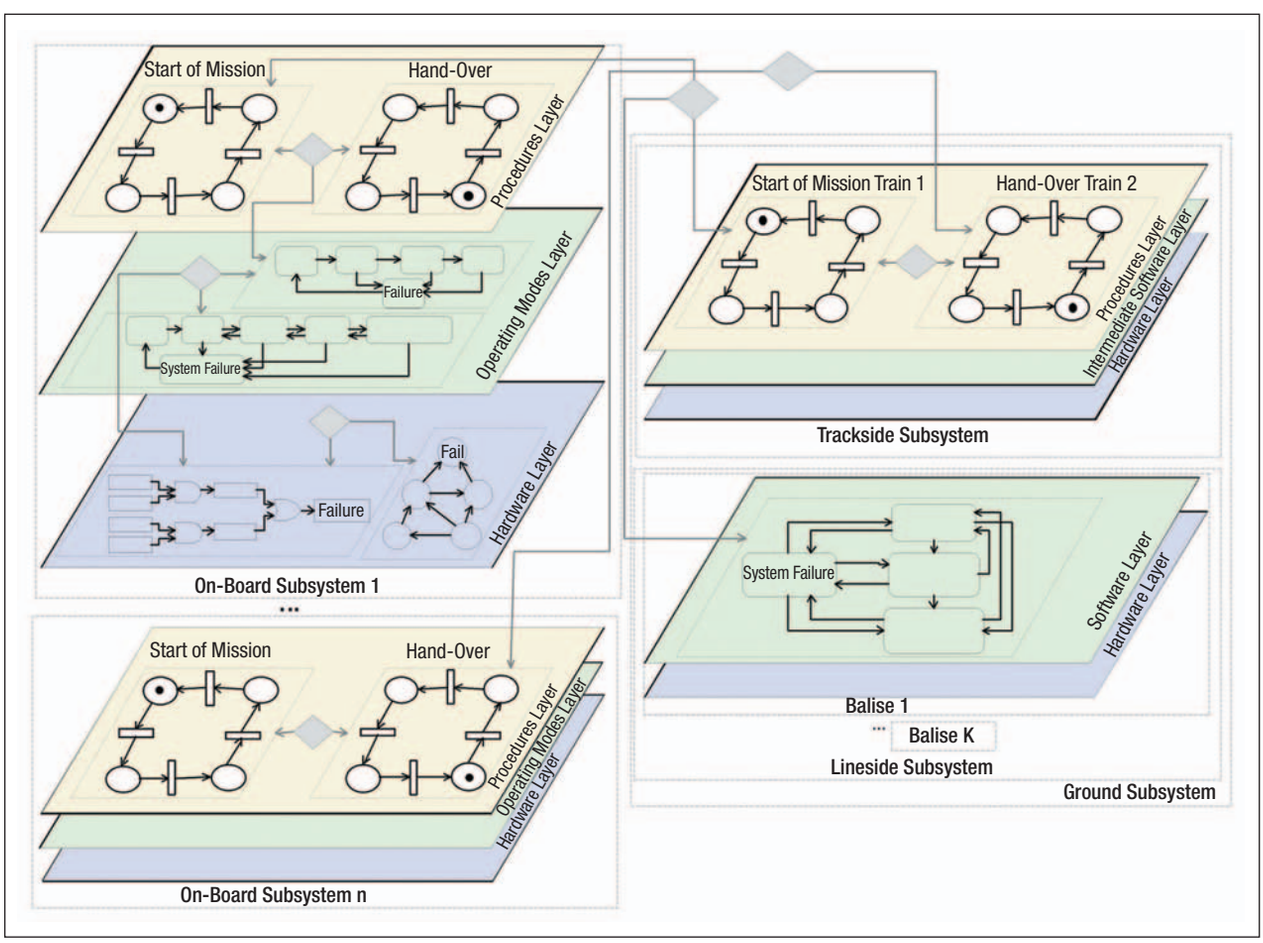


FIGURA 7

Schema per la modellazione multi-formalismo di un complesso sistema di controllo ferroviario

I modelli sono interconnessi attraverso operatori (rappresentati nella Figura 7 dai rombi a sfondo grigio):
- *intra /inter-livello*, a seconda che l'interazione avvenga tra modelli dello stesso livello (*intra*) o di diversi livelli (*inter*);
- *intra /inter-componente*, a seconda che l'interazione avvenga tra modelli dello stesso componente (*intra*) o di componenti diversi (*inter*).

Un siffatto modello consentirebbe una valutazione olistica di attributi di *dependability* (per esempio, disponibilità o sicurezza), relativi a modi di guasto definiti a livello di sistema complessivo, che altrimenti andrebbero valutati attraverso l'analisi di modelli isolati, con la conseguenza di dover effettuare ipotesi poco realistiche o eccessivamente conservative (laddove possibile). Ipotesi poco realistiche possono compromettere la validità dei risultati, perché si trascurano i dettagli di alcune dinamiche di interazione; dal lato opposto, l'essere eccessivamente conservativi costringe ad un dimensionamento non ottimale dei parametri di progetto, con il rischio di comprometterne la fattibilità economica. Inoltre, la valutazione olistica di attributi di *dependability* facilita la produzione delle evidenze formali necessarie ad ottenere eventuali certificazioni.

La definizione di siffatti modelli gerarchici consente di affrontare l'analisi un sistema complesso secondo opportune metodologie. Per esempio è possibile modellare un sistema definendo un livello che descriva il comportamento dei diversi sotto-sistemi, un livello che descriva l'interazione tra i sotto-sistemi mediante una rete e un livello che modelli la rete di comunicazione. Ancora, è possibile sviluppare dei modelli che consentano di considerare l'impatto della manutenzione sull'affidabilità e sicurezza del sistema, sviluppando separatamente e poi componendo un modello di guasto ed uno di riparazione [12].

Modelli multi-formalismo sviluppati secondo un "approccio a componenti" consentono inoltre di gestire la complessità in diversi modi. Per esempio, tra gli approcci finalizzati alla riduzione delle dimensioni dei modelli è possibile individuare quelli:
- modulari, iterativi e gerarchici, ovvero di tipo "*divide et impera*" [7];
- di sostituzione di sottomodelli con modelli equivalenti, ma di dimensioni ridotte (si veda per esempio [15]);
- parametrici, che consentono una compattazione attraverso lo sfruttamento di simmetrie (tecnica nota come "*folding*", letteralmente "ripiegamento") [5].

A più basso livello, si possono impiegare tecniche che ricorrono al parallelismo distribuendo il carico di lavoro necessario alla risoluzione dei sotto-modelli su diverse macchine.

Evidentemente, l'analisi di modelli così complessi implica la disponibilità di opportune tecniche di soluzione e di strumenti automatici che supportino anche lo sviluppo del modello. Se è pensabile di sviluppare e risolvere separatamente utilizzando i *tool* disponibili modelli molto semplici, quale quello di figura 5, realizzando "a mano" l'operatore di composizione, ovvero prelevando il risultato ottenuto dall'analisi del primo modello e immettendolo opportunamente nel secondo), non è possibile procedere in questo modo per modelli quale quello mostrato nella Figura 7, e nei casi in cui gli operatori realizzino funzioni complesse, ad esempio nel caso in cui forniscano informazioni circa l'evoluzione di simulazioni in corso.

Esistono diversi strumenti di supporto allo sviluppo ed alla risoluzione di modelli multi-formalismo abbastanza assestati (uno stato dell'arte è riportato in [22]), tra cui è opportuno citare *SHARPE* [21], *SMART* [6], *DEDS* [2], *AToM*^{3} [16] e *Möbius* [8], che si differenziano tra loro per diversi aspetti. Per esempio, un'importante distinzione è tra i *framework* che ricorrono ad un unico formalismo di più alto livello per la rappresentazione del modello globale e quelli che invece mantengono separati i sottomodelli, che vengono valutati ognuno attraverso un proprio specifico risolutore. Tra questi ultimi è opportuno citare l'approccio innovativo proposto dal *framework OsMoSys* (*Object-baSed Multiformalism mOdeling of SYStems*) [24], che consiste nell'orchestrazione di risolutori esterni attraverso tecniche di *work-flow management* [9].

5. CONCLUSIONI

I modelli di *dependability* risultano di estrema importanza per supportare lo sviluppo e l'analisi dei sistemi *dependable*, in particolare di

quelli dedicati impiegati in applicazioni critiche (controllo di reattori nucleari, *fly/brake/steer-by-wire*, e-medicine, *e-banking* ecc.). Il loro impiego è contemplato da tempo nelle *best-practice* industriali per la valutazione di determinati attributi di affidabilità e sicurezza per sottosistemi limitati. Tre sono i fattori di cui tener conto allo scopo di indirizzare le tematiche di ricerca future in questo ambito:

- la crescente complessità dei sistemi di elaborazione, dovuta all'elevato numero di requisiti funzionali, che impattano su dimensioni del codice, distribuzione ed eterogeneità;
- la crescente criticità dei requisiti non funzionali, che diventano sempre più stringenti dal momento che un fallimento può avere impatti disastrosi in termini economici o di salute;
- la necessità di adottare approcci olistici per assicurare gli obiettivi di progetto, supportando le decisioni in fase di sviluppo e dando evidenza formale dei risultati ottenuti in fase di certificazione (*assessment*).

La gestione attraverso tecniche di modellazione multi-formalismo delle problematiche sopra menzionate richiede il raggiungimento di obiettivi ambiziosi, legati sia ad evoluzioni metodologiche che ad aspetti tecnologici. Tra questi citiamo la riduzione della complessità risolutiva, la ricerca di nuovi operatori per l'interoperabilità tra modelli eterogenei, e la simulazione distribuita. A fronte di uno sforzo sicuramente notevole richiesto in tali ambiti, i nuovi paradigmi di modellazione si prestano ad impieghi che travalicano i diversi settori dell'ingegneria, essendo applicabili a contesti multidisciplinari, quale quello delle infrastrutture critiche [13].

Bibliografia

[1] Avizienis A., Laprie J.-C., Randell B., Landwehr C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. In: *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, 2004, p. 11-33.

[2] Bause F., Buchholz P., Kemper P.: *A toolbox for functional and quantitative analysis of DEDS*. In: Proc. 10-th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, LNCS 1469, Springer-Verlag, 1998, p. 356-359.

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

[4] Bondavalli A., Dal Cin M., Latella D., Majzik I., Pataricza A., Savoia G.: Dependability analysis in the early phases of UML-based system design. In: *Computer Systems Science and Engineering*, Vol. 16, n. 5, 2001, p. 265-275.

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

[6] Ciardo G., Miner A.: SMART: *Simulation and Markovian Analyser for Reliability and Timing*. In Proc. 2-nd International Computer Performance and Dependability Symposium (IPDS'96), IEEE Computer Society Press, 1996, p. 60.

[7] Daly D., Sanders W.H.: *A connection formalism for the solution of large and stiff models*. In: Proc. 34-th Annual Simulation Symposium, 2001, p. 258-265.

[8] Deavours D.D., Clark G., Courtney T., Daly D., Derisavi S., Doyle J.M, Sanders W.H., Webster P.K.: The Möbius Modeling Framework and Its Implementation. *IEEE Trans. Software Engineering*, Vol. 28, n. 10, 2002, p. 956-969.

[9] Di Lorenzo G., Flammini F., Iacono M., Marrone S., Moscato F., Vittorini V.: *The software architecture of the OsMoSys multisolution framework*. In: Proc. 2-nd International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS'07, Nantes, France, October 23-25, 2007, p. 1-10.

[10] Flammini F., Iacono M., Marrone S., Mazzocca N.: *Using Repairable Fault Trees for the evaluation of design choices for critical repairable systems*. In: Proceedings of the 9-th IEEE Symposium on High Assurance Systems Engineering, HASE'05, Heidelberg, Germany, October 12-14, 2005, p. 163-172.

[11] Flammini F., Marrone S., Mazzocca N., Vittorini V.: *Modelling System Reliability Aspects of ERTMS/ETCS by Fault Trees and Bayesian Networks*. In: Safety and Reliability for Managing Risk: Proceedings of the 15-th European Safety and Reliability Conference, ESREL'06, Estoril, Portugal, September 18-22, 2006, p. 2675-2683.

[12] Flammini F., Marrone S., Mazzocca N., Vittorini V.: *A new modeling approach to the safety evaluation of N-modular redundant computer systems in presence of imperfect maintenance*. In: Reliability Engineering & System Safety, Vol. 94, Issue 9, September 2009, p. 1422-1432.

[13] Flammini F., Mazzocca N., Pragliola C., Vittorini V.: *A Study on Multiformalism Modelling of Critical Infrastructures*. In: Proc. 3-rd International Workshop on Critical Information Infrastructures Security, CRITIS-08, LNCS 5508, 2009, p. 336-343.

[14] Holzmann G.J: Conquering Complexity. In: *IEEE Computer*, Vol. 40, n. 12, 2007, p. 111-113.

[15] Jungnitz H., Desrochers A.A.: *Flow equivalent nets for the performance analysis of Generalized Stochastic Petri Nets*. In: Proceedings of the IEEE International Conference Robotics and Automation, 1991, p. 122-127.

[16] Lara J. d., Vangheluwe H.: AToM3: A *Tool for Multi-formalism and Meta-modelling*. In: Proc. 5-th Intl. Conf. on Fundamental Approaches To Software Engineering, LNCS vol. 2306. Springer-Verlag, London, 2002, p. 174-188.

[17] Laurent Doyen L., Henzinger T.A., Jobstmann B., Petrov T.: *Interface theories with component reuse*. In: Proc. 8-th Annual Conference on Embedded Software (EMSOFT), ACM Press, 2008, p. 79-88.

[18] Merz S.: Model checking: a tutorial overview. In: Modeling and Verification of Parallel Processes. *LNCS*, Vol. 2067, 2001, p. 3-38.

[19] Mosterman P.J., Vangheluwe H.: Computer Automated Multi-Paradigm Modeling: An Introduction. *SIMULATION*, Vol. 80, n. 9, 2004, p. 433-450.

[20] Nicol D.M., Sanders W.H., Trivedi K.S.: Model-based evaluation: from dependability to security. In: *Dependable and Secure Computing, IEEE Transactions* on, Vol.1, Iss.1, 2004, p. 48- 65.

[21] Sahner R.A., Trivedi, K.S., Puliafito, A.: *Performance and Reliability Analysis of Computer Systems. An Example-based Approach Using the SHARPE Software Package*, Kluwer Academic Publishers, 1996.

[22] Sanders W.H.: *Integrated Frameworks for Multi-Level and Multi-Formalism Modeling*. In: Proc. of the 8-th Intl. Workshop on Petri Nets and Performance Models, 1999, p. 2.

[23] Shooman M.L.: *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design*. John Wiley & Sons Inc., 2002.

[24] Vittorini V., Iacono M., Mazzocca N., Franceschinis G.: The OsMoSys approach to multi-formalism modeling of systems. *Software and Systems Modeling*, Vol. 3, Issue 1, 2004, p. 68–81.

FRANCESCO FLAMMINI ha ottenuto la laurea con lode (2003) ed il dottorato di ricerca in Ingegneria Informatica (2006) presso l'Università di Napoli "Federico II". Dal 2003 lavora in Ansaldo STS come progettista e ricercatore, occupandosi di verifica dei sistemi di controllo e protezione delle infrastrutture. Ha tenuto come professore a contratto corsi di informatica ed ingegneria del software e vari seminari sui sistemi sicuri ed affidabili. È autore di oltre 30 articoli scientifici pubblicati su riviste, libri e atti di congressi internazionali. È membro di IEEE CS, EWICS TC7 ed ERCIM FMICS.
E-mail: frflammi@unina.it

NICOLA MAZZOCCA ha ottenuto la laurea con lode (1987) ed il dottorato di ricerca in Ingegneria Elettronica (1992) presso l'Università di Napoli "Federico II". È professore ordinario di Sistemi di Elaborazione presso l'Università di Napoli "Federico II". Ha tenuto oltre 35 corsi universitari ed è autore di oltre 200 pubblicazioni, riguardanti prevalentemente prestazioni ed affidabilità dei calcolatori. È stato Consigliere del Ministro per l'Innovazione e le Riforme nella Pubblica Amministrazione ed è attualmente Assessore all'Università e alla Ricerca Scientifica per la Regione Campania.
E-mail: nicola.mazzocca@unina.it

VALERIA VITTORINI è professore associato presso l'Università di Napoli "Federico II", dove è docente di Fondamenti di Informatica e di Programmazione. È autrice di numerose pubblicazioni scientifiche sull'utilizzo di tecniche formali di modellazione di sistemi reali in diversi ambiti applicativi. Afferisce al Dipartimento di Informatica e Sistemistica, presso cui svolge la propria attività di ricerca, prevalentemente incentrata sullo studio di sistemi distribuiti e sullo sviluppo di metodologie e strumenti per l'analisi di sistemi critici.
E-mail: valeria.vittorini@unina.it