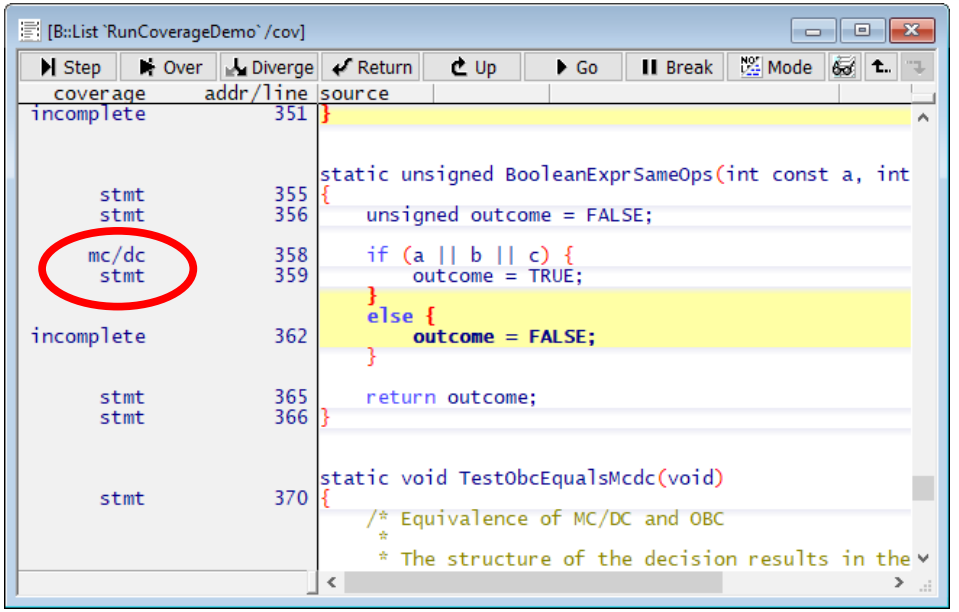
Vantaggi e insidie dell'analisi di copertura del codice basata sul trace per ASIL D

Una soluzione che fa la differenza

Maurizio Menegotto
Lauterbach Italy

Christoph Sax
Lauterbach Germany



www.lauterbach.com

LAUTERBACH
DEVELOPMENT TOOLS

Introduzione

ISO 26262 richiede un rigoroso processo di verifica e validazione multilivello, studiato per ridurre in un sistema automobilistico il rischio di errori dovuti al software.

→ Uno strumento fondamentale per il successo di questo approccio è l'analisi di copertura del codice. Per ottenere informazioni sul comportamento dinamico di un'applicazione, le tecniche esistenti richiedono un'instrumentazione del codice sorgente.

→ Un'alternativa praticabile all'instrumentazione del codice si fonda sulle capacità insite nell'hardware dei moderni chip multicore.

→ Questa presentazione intende proporre un metodo per la misura di metriche complesse di copertura strutturale – come MC/DC (Modified Condition/Decision Coverage) – basato su un monitoraggio hardware non intrusivo. Di questa tecnica non invasiva si esamina l'adeguatezza rispetto alle linee guida indicate da ASIL D. Vengono inoltre evidenziati vantaggi e svantaggi in confronto alle attuali soluzioni che dipendono dall'instrumentazione.

Una soluzione che fa la differenza

Vantaggi e insidie dell'analisi di copertura del codice basata sul trace per ASIL D

Christoph Sax

2018 / 02 / 22

www.lauterbach.com

LAUTERBACH
DEVELOPMENT TOOLS

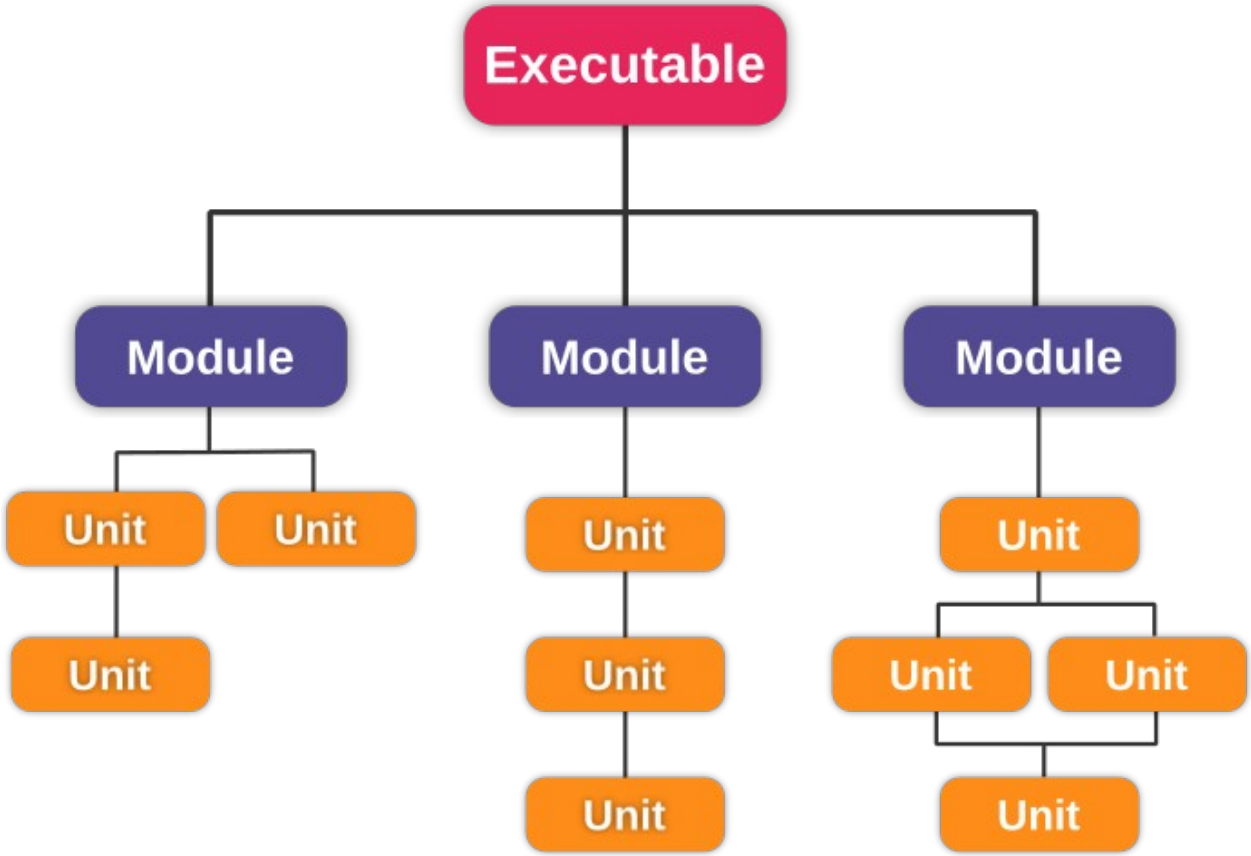
Agenda

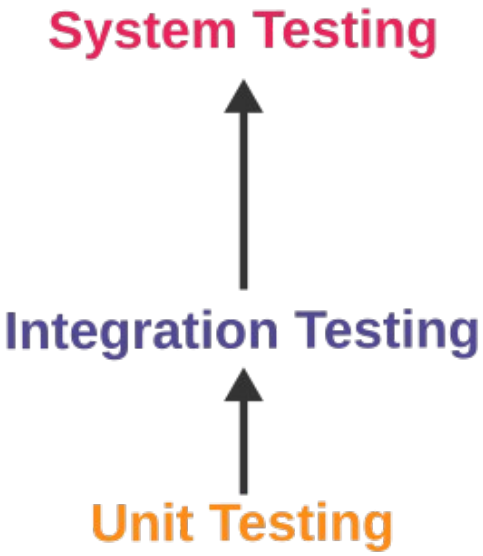
- Introduzione
- Copertura del codice basata sul trace per ASIL-D
- Un nuovo approccio che migliora lo stato dell'arte
- Prospettive
- Conclusioni

Agenda

- Introduzione
- Copertura del codice basata sul trace per ASIL-D
- Un nuovo approccio che migliora lo stato dell'arte
- Prospettive
- Conclusioni

Verifiche Software in ISO 26262





Metriche di copertura strutturale in ASIL D

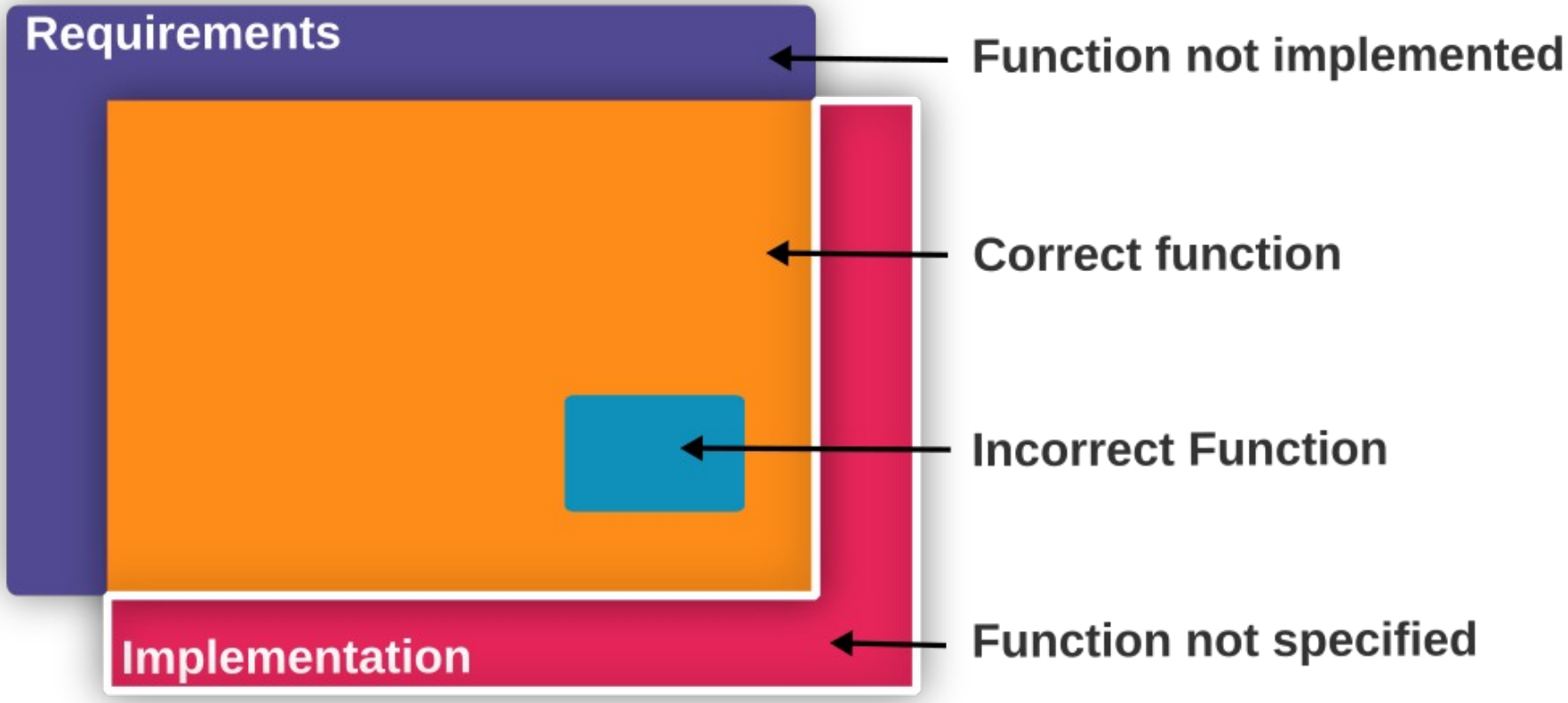
- Test di Unità

Automotive Safety Integrity Level	Highly Recommended
ASIL A	Statement Coverage
ASIL B	Statement Coverage + Branch Coverage
ASIL C	Branch Coverage
ASIL D	MC/DC

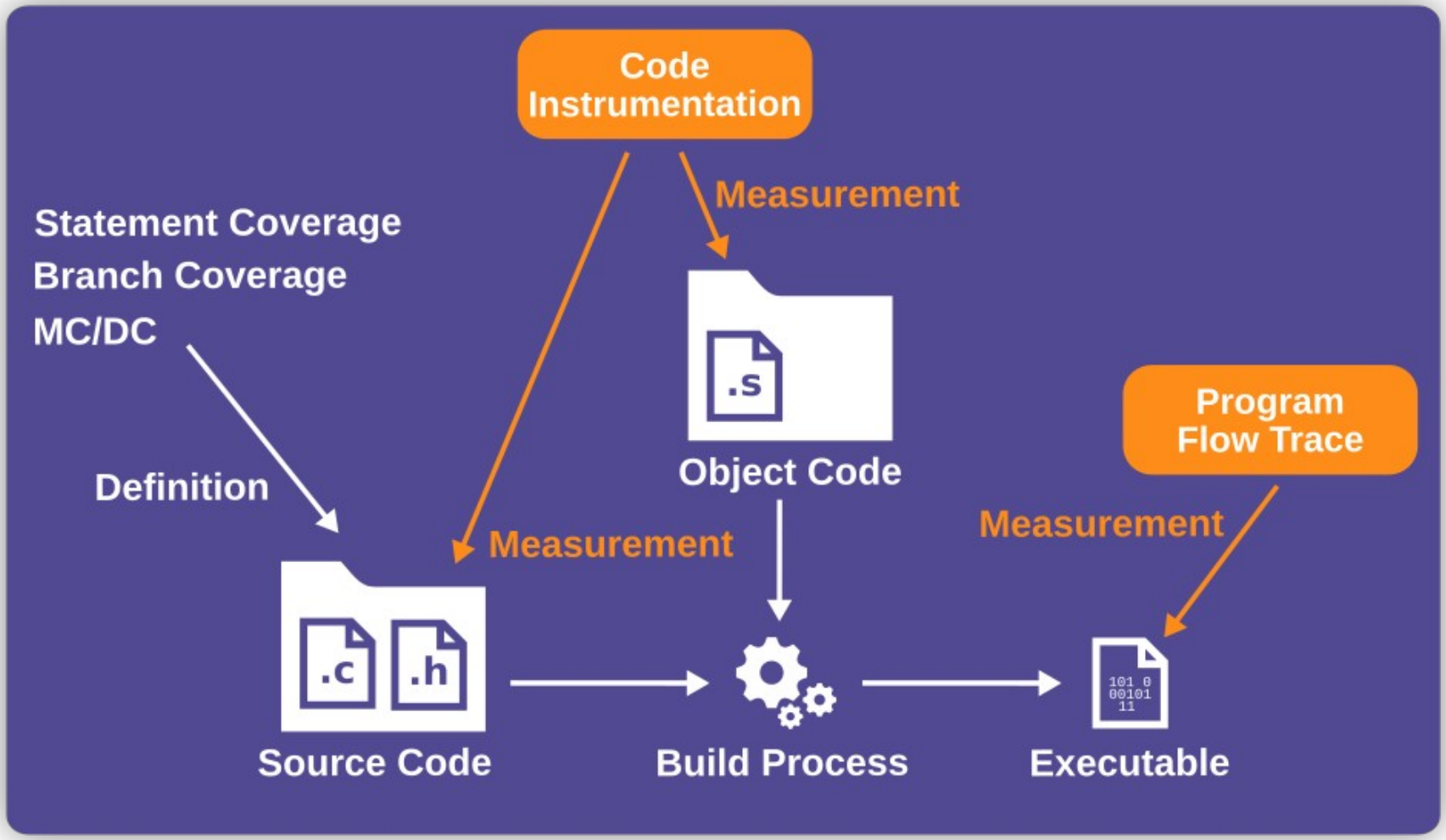
- Test di Integrazione

Automotive Safety Integrity Level	Highly Recommended
ASIL A	-
ASIL B	-
ASIL C	Function Coverage + Call Coverage
ASIL D	Function Coverage + Call Coverage

Obiettivi dell'analisi di copertura strutturale



Misurare la copertura di codice



Tracciamento del flusso di programma

- Monitoraggio dinamico del codice oggetto eseguibile
  - Sequenze lineari
  - Esecuzione condizionale
  - Eventi eccezionali
- Tracciamento del contesto del sistema operativo

- Interfaccia dedicata
- Memoria interna
- I/O funzionale
  - USB
  - PCI-e

Vantaggi

- Nessun impatto sul comportamento runtime
- Nessun sovraccarico sulla memoria
- Possibilità di test sul codice finale di produzione
- Protezione contro effetti collaterali indesiderati
- Può essere unito al Profiling dell'esecuzione
- Analisi della logica di basso livello non presente a livello di codice sorgente
  - Controlli di errore
  - Uso di librerie runtime

Agenda

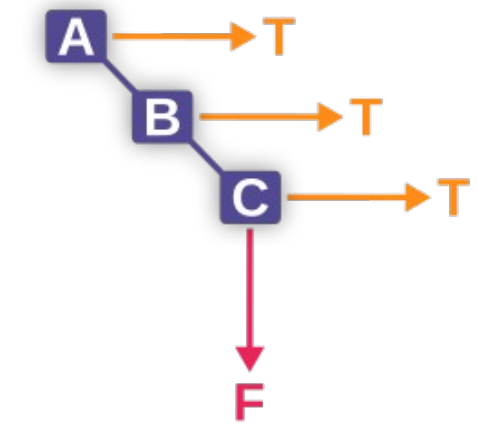
- **Introduzione**
- Copertura del codice basata sul trace per **ASIL D**
- **Un nuovo approccio che migliora lo stato dell'arte**
- **Prospettive**
- **Conclusione**

Modified Decision/Condition Coverage (MC/DC)

- Definizione
  - Ogni punto di entrata e di uscita è stato invocato almeno una volta
  - Ogni decisione ha avuto tutti i possibili risultati
  - Ogni condizione di una decisione ha avuto tutti i possibili risultati almeno una volta
  - Si è mostrato che ogni condizione di una decisione influenza in modo indipendente il risultato di quella decisione

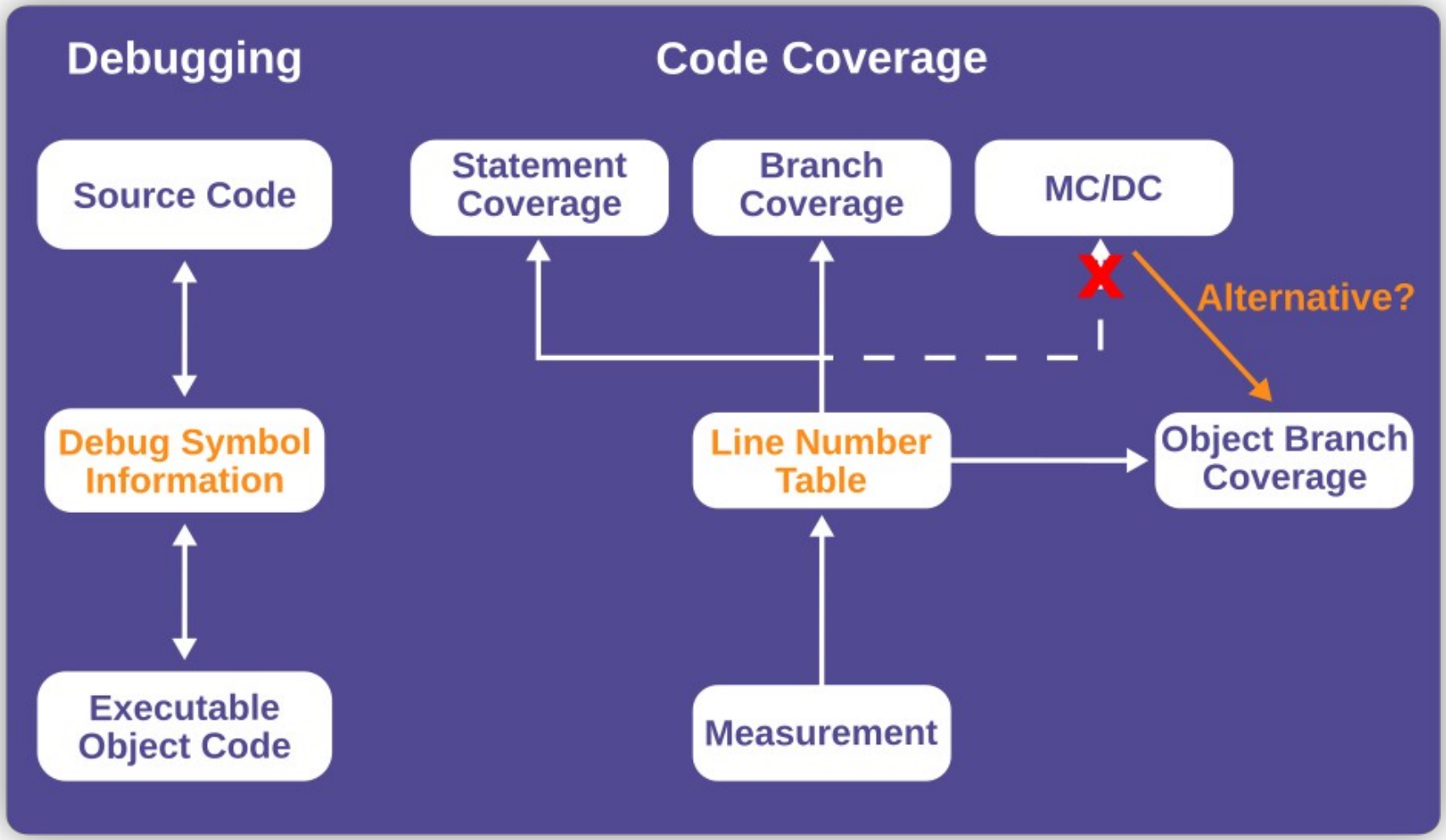
- Esempio: `A or B or C`

#	A	B	C	Outcome	Pairs		
1	T	x	x	T	4	-	-
2	F	T	x	T	-	4	-
3	F	F	T	T	-	-	4
4	F	F	F	F	1	2	3



Control Flow Graph

Schema di principio



Validità dell'analisi di copertura a livello di codice oggetto

- Posizione dell'industria avionica
  - L'analisi di copertura a livello di codice oggetto può essere usata per raggiungere la conformità agli obiettivi di copertura strutturale
  - Andrebbe dimostrata l'equivalenza con l'analisi di copertura di codice a livello di codice sorgente
  - Dovrebbe essere richiesto lo stesso numero di casi di test minimi

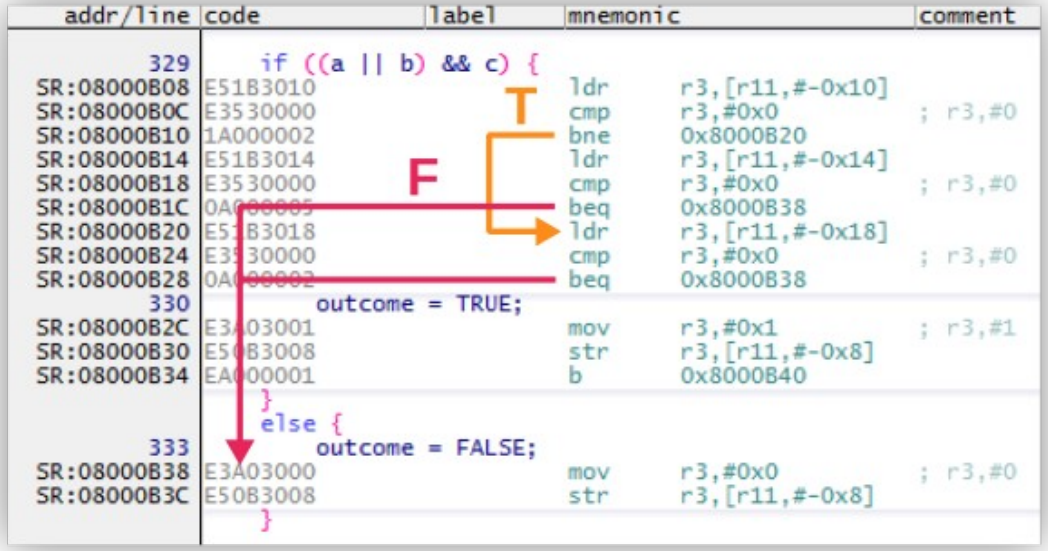
► Alla base di tutto: capacità di rilevare gli errori

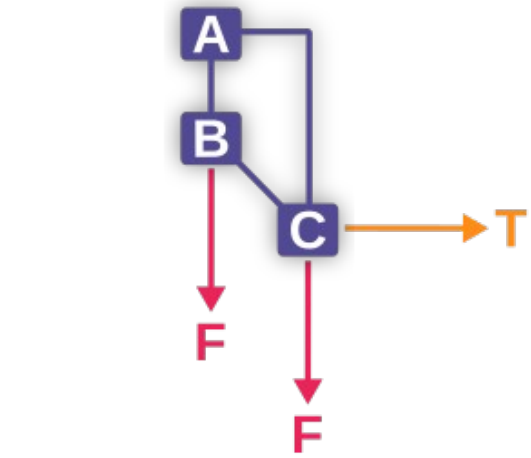
- Rilevazione errori in MC/DC
  - Risultati di una decisione
  - Esecuzione di coppie indipendenti (independence pairs)

Object Branch Coverage (OBC)

- Definizione
  - Ogni istruzione del codice oggetto è stata eseguita almeno una volta
  - Ogni salto condizionato è stato eseguito in entrambe le direzioni (predicato vero/falso) almeno una volta

- Esempio: `(A or B) and C`





Edge Coverage
in Control Flow Graph

Valutazione del Object Branch Coverage

- Ridotta capacità di rilevare gli errori
  - È meno impegnativo dimostrare OBC
  - È difficile confermare le coppie indipendenti
  - Equivalenza con MC/DC per le decisioni che hanno fino a due condizioni
  - La differenza fra OBC e MC/DC aumenta al crescere della complessità delle decisioni
- Le ottimizzazioni del compilatore peggiorano le cose
- Contromisure per le mancanze
  - Trovare coppie indipendenti
  - Eseguire una per una le coppie indipendenti
  - Confermare l'esecuzione delle coppie indipendenti

► OBC non è un buon sostituto di ASIL D

Agenda

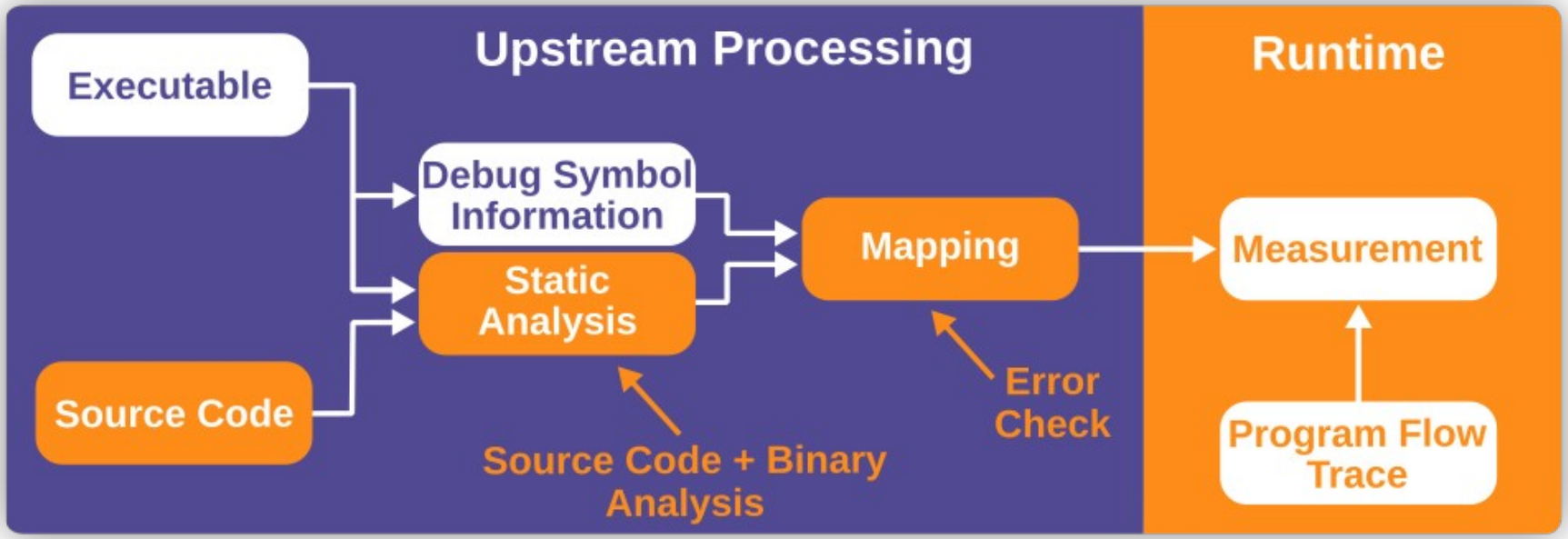
- Introduzione
- Copertura del codice basata sul trace per ASIL-D
- **Un nuovo approccio che migliora lo stato dell'arte**
- Prospettive
- Conclusioni

Lavori correlati

- Analisi di copertura strutturale su codice non instrumentato (2012 AdaCore)
  - Estensione per compilatore GNAT
    - L'informazione di debug memorizza gli obiettivi della copertura
    - Generazione di codice oggetto per le misure di copertura
  - Toolchain completa in cui il compilatore è il componente principale
- Estensione al formato DWARF con dati per controllo di flusso (2014 HighTec)
  - Su misura per un caso d'uso specifico
    - Copertura dei salti per l'istruzione switch
    - Ottimizzazione delle tabelle di jump
  - Il compilatore fornisce ad altri tool dati aggiuntivi

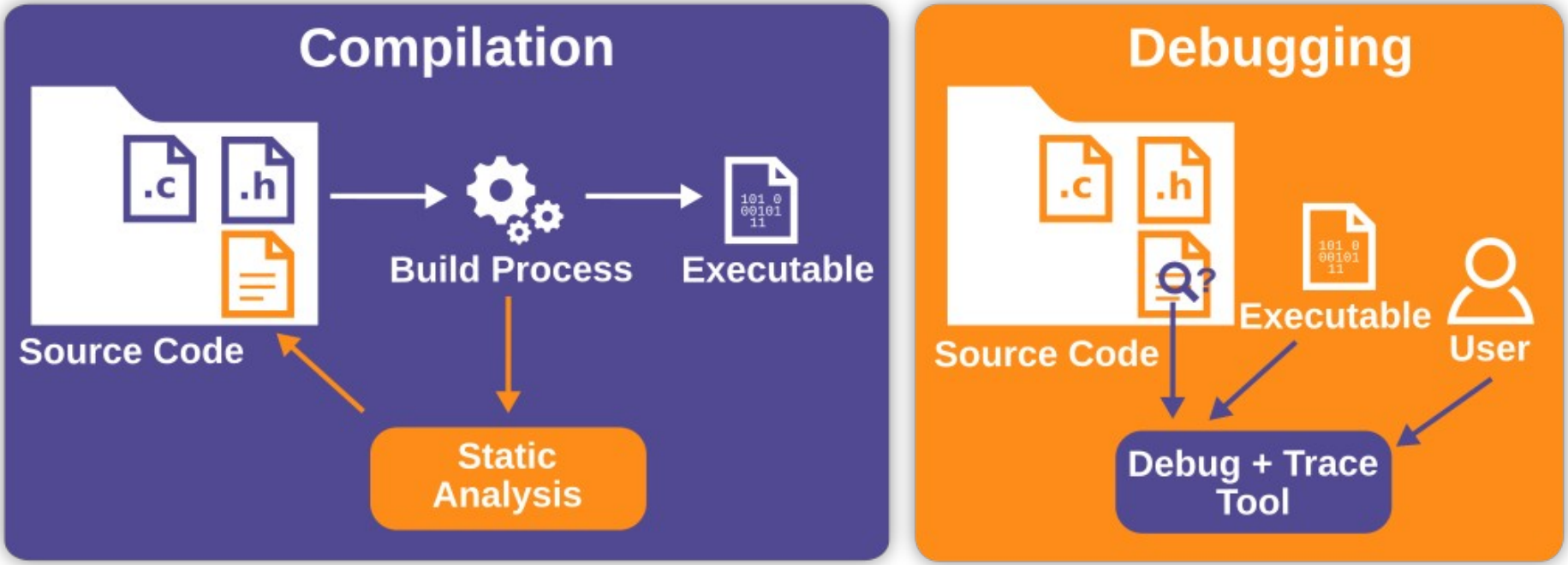
► Idea principale: il codice oggetto riflette la struttura del codice sorgente

Aggiungere i collegamenti mancanti



- Elaborazione a monte
  - Analisi del controllo di flusso mediante analisi statica
  - Traduzione efficiente del codice sorgente in struttura binaria
- Elaborazione runtime
  - Monitoraggio a livello di codice oggetto
  - Non intrusivo

Lo stadio di pre-elaborazione



- Analisi statica eseguita da un tool a linea di comando
- Interfaccia progettata per una semplice integrazione nel processo di build

- I dati sul controllo di flusso possono essere posti sotto controllo di versione
- Caricamento dell'eseguibile e dei dati sul controllo di flusso nella fase di setup

Analisi statica

- Estensioni
  - Tabella del Numero delle Linee
    In quali linee sono presenti decisioni?

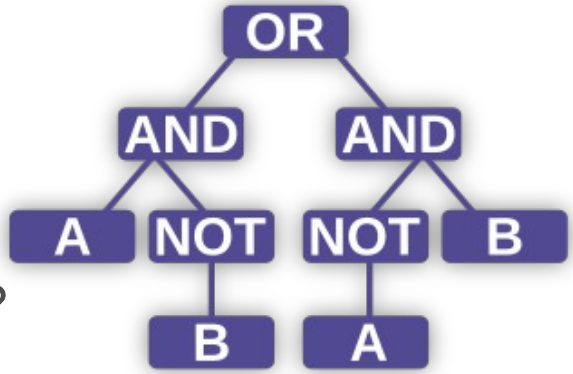
    Qual è l'ordine atteso delle decisioni a livello di codice oggetto?
  - Eseguibile
    A quali indirizzi sono presenti istruzioni/salti condizionati?
  - Codice sorgente
    Come si riflette la struttura di una decisione
    a livello di codice oggetto?

    Qual è la complessità di una decisione?

    Quali termini sono mascherati nella valutazione?

    Qual è il risultato di una condizione?

    In che punto è noto il risultato di una decisione?



Risultati

- Un metodo portabile per la misura del “Masking MC/DC”, costruito sui punti di forza della copertura di codice basata sul trace
  - Nessuna interferenza sulle procedure di misura già esistenti
  - Misura efficiente delle coppie indipendenti
  - Sono richieste solo prestazioni di base del trace
- Supporto per nuovo e vecchio hardware
- Non vincolato a uno specifico compilatore, ma dipendente dal codice oggetto generato
- Demo per ARM e PowerPC come verifica di quanto previsto

Limitazioni all'applicabilità

- Il tracciamento delle condizioni richiede un'esecuzione condizionale nel codice oggetto
  - Limitazione tecnica del trace del flusso di programma
  - Un'istruzione/salto condizionale per condizione
- Le espressioni booleane sono limitate alla forma “short-circuited”
  - Segnalare l'intenzione al compilatore
  - Favorisce l'uso di salti condizionali
- Mancanza di supporto da parte del compilatore
  - Il livello di ottimizzazione è il principale criterio di controllo nella generazione di codice oggetto
  - Impatto globale su file/progetto

Agenda

- Introduzione
- Copertura del codice basata sul trace per ASIL-D
- Un nuovo approccio che migliora lo stato dell'arte
- Prospettive
- Conclusioni

Valutazioni

- Una gran parte del codice sorgente può essere ottimizzato
  - Esempio: decompressore JPEG in C
    - 16 KLOC ~ 1000 decisioni
    - Più del 90% ha <3 condizioni

► È possibile generare codice con alte prestazioni/basso consumo di memoria

- Opportunità per i tool di supporto
  - Generazione di codice ottimizzato con basse penalizzazioni in prestazioni e dimensioni
  - Controllo di flusso tracciabile a livello di codice oggetto
  - Integrazione della fase di analisi statica
- Vantaggi
  - Portabilità su ambienti con vincoli sulle risorse
  - Throughput accresciuto per “continuous integration”
  - Barattare l'efficienza del codice con la riduzione del costo del test

Applicazioni future

- Processo a due fasi obbligatorio per ISO 26262
  - Test con instrumentazione → ha valore per la copertura del codice
  - Test senza instrumentazione → ha valore per i risultati del test
- La soluzione basata sul trace non richiede instrumentazione di codice
- Uguali opzioni di compilazione per tutti i test
  - Test con speciali opzioni di build → ha valore per la copertura del codice
  - Test con le opzioni di default del build → ha valore per i risultati del test
- Uso di un compilatore qualificato
  - Qualificare entrambe le configurazioni di build
  - Test con speciali opzioni di build → ha valore per la copertura del codice e per i risultati del test
- Le prestazioni del codice oggetto generato sono piuttosto buone
  - Eseguire il test → ha valore per la copertura del codice e per i risultati del test

Agenda

- Introduzione
- Copertura del codice basata sul trace per ASIL-D
- Un nuovo approccio che migliora lo stato dell'arte
- Prospettive
- Conclusione

Conclusioni

- Usare Object Branch Coverage per ASIL D è difficile e si possono ottenere risultati non corretti
- Questa presentazione ha cercato di esporre un approccio alla copertura di codice basata sul trace, studiato per eliminare le differenze rispetto al codice instrumentato

Thank you!

QUESTIONS?

Christoph Sax
christoph.sax@lauterbach.com

LAUTERBACH
DEVELOPMENT TOOLS

Riferimenti

- Braunes, Jens. “Branch Coverage für optimierten Code”. Design & Elektronik. March 2017.
- Comar, Cyrille, et al. "Formalization and comparison of MCDC and object branch coverage criteria." ERTS (Embedded Real Time Software and Systems Conference). 2012.
- FAA. "DOT/FAA/AR-07/20 – Object-Oriented Technology Verification Phase 3 Report – Structural Coverage at the Source-Code and Object-Code". 2007.
- FAA. “Position Paper CAST-17 – Structural Coverage of Object Code”. 2003.
- ISO 26262-6:2011: “Road vehicles – functional safety”. 2011.