Alma Mater Studiorum – Università di Bologna

Processo di sviluppo per applicazioni multi-piattaforma e multi-paradigma: il caso tuProlog

Corso di Laurea Magistrale in Ingegneria Informatica
Tesi in Linguaggi e Modelli Computazionali M

Relatore:
Prof. Enrico Denti

Presentata da:
Andrea Bucaletti

Correlatore:
Dott.Ing. Roberta Calegari

Sessione II – AA 2014/2015

Introduzione

- Processo di sviluppo di applicazioni multi-piattaforma, multi-paradigma e multi-linguaggio
  - Compilazione, testing, packaging, documentazione …
  - Costituito da passi legati tra loro da relazioni di dipendenza
  - Complessità crescente con il numero di linguaggi e piattaforme considerate
    - Integrazione di “sotto-processi” di sviluppo specifici per i linguaggi e le piattaforme considerate
    - Integrazione di componenti implementati in linguaggi diversi
    - Gestione di molteplici implementazioni dell'applicazione

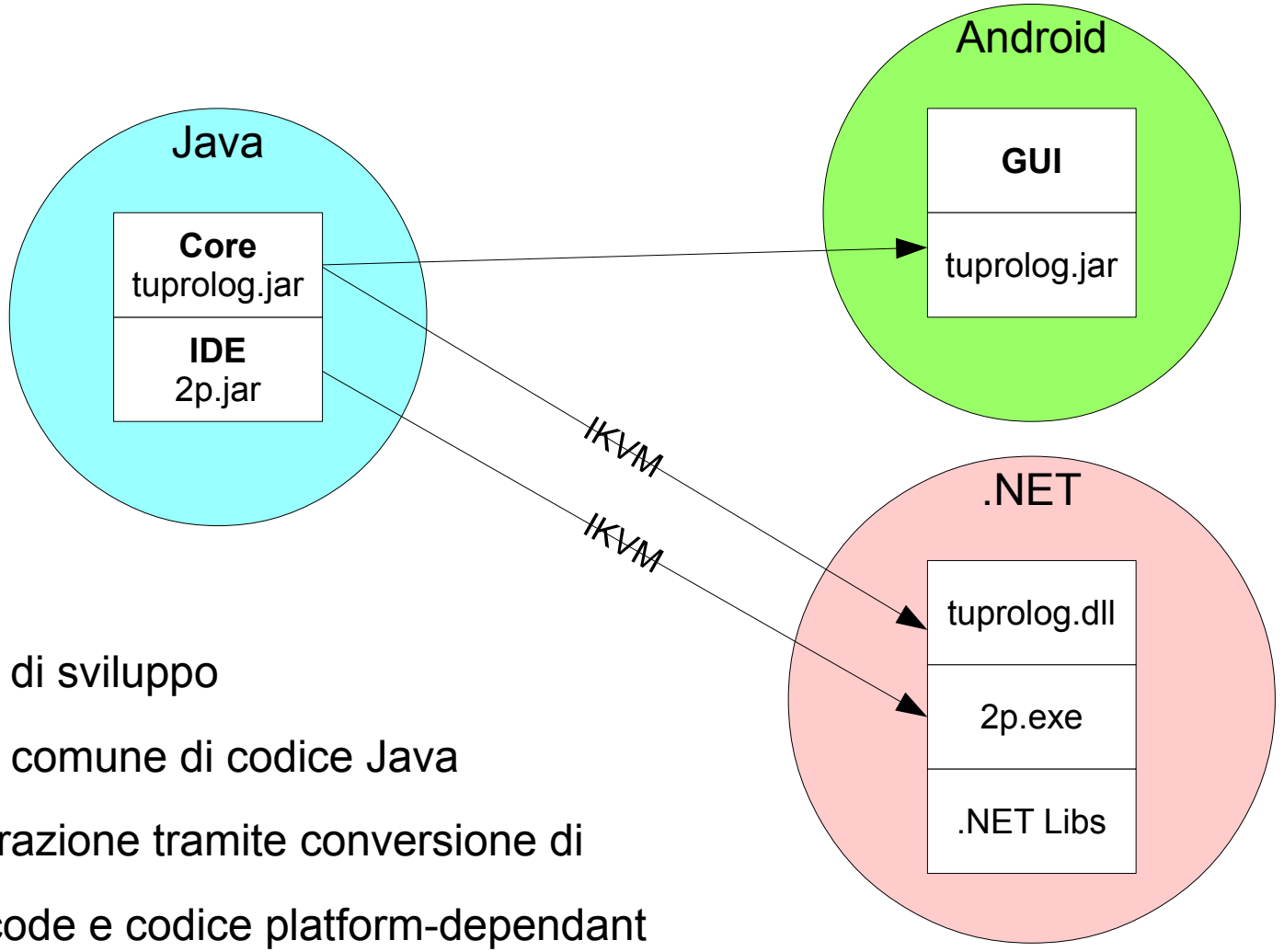
Obiettivi

- Analizzare i principali problemi di progettazione di processi di sviluppo multi-piattaforma e multi-linguaggio
- Applicazione ad un caso concreto: tuProlog
  - Interprete prolog disponbile per Java, .NET, Android e iOS
  - Refactoring del processo di sviluppo di tuProlog in modo da renderlo
    - Estendibile
    - Facilmente riconfigurabile
  - Estendere la capacità di programmazione multi-paradigma di tuProlog
    - Specifica JSR-223 - Scripting for the Java Platform

Dimensioni del problema

- Complessità crescente con il numero di piattaforme e linguaggi utilizzati dal processo
  - Integrazione di molteplici “sotto-processi” relativi alle singole piattaforme e linguaggi
- Gestione delle dipendenze
  - Dipendenze tra i sotto-processi (basi di codice comuni)
  - Dipendenze interne ai sotto-processi
- Configurazione
  - Ogni sotto-processo utilizza un insieme di tool per realizzare le diverse fasi (compilatore, packager, documentazione, test, …)
  - Ogni tool utilizzato necessita di un insieme di parametri di configurazione

Caso di studio: tu Prolog



Processo di sviluppo

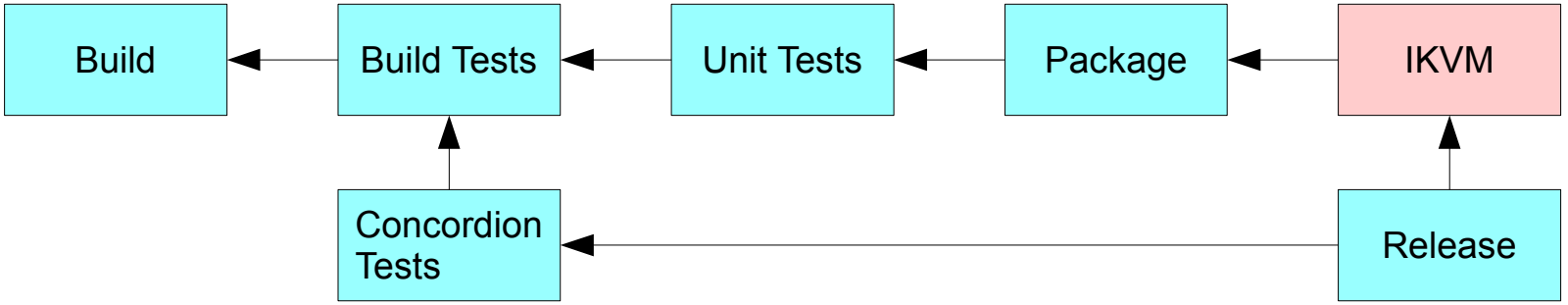
- Base comune di codice Java
- Integrazione tramite conversione di bytecode e codice platform-dependant
- Svolto in buona parte in maniera manuale con l'ausilio di alcuni script Apache ANT

Analisi del problema – Dipendenze inter-piattaforma

- Basi di codice comuni possono creare delle dipendenze tra i processi di sviluppo delle piattaforme considerate
  - Es. Processo di sviluppo Android di tuProlog
    - Utilizzo del processo di sviluppo Java per la generazione del core
    - Integrazione con codice platform-dependat (GUI per Android)
- Le dipendenze inter-piattaforma possono rendere complessa l'integrazione di
  - nuove piattaforme
  - nuove funzionalità relative alle piattaforme esistenti
- Requisito: separazione tra i processi di sviluppo per le piattaforme considerate in modo da renderle indipendenti

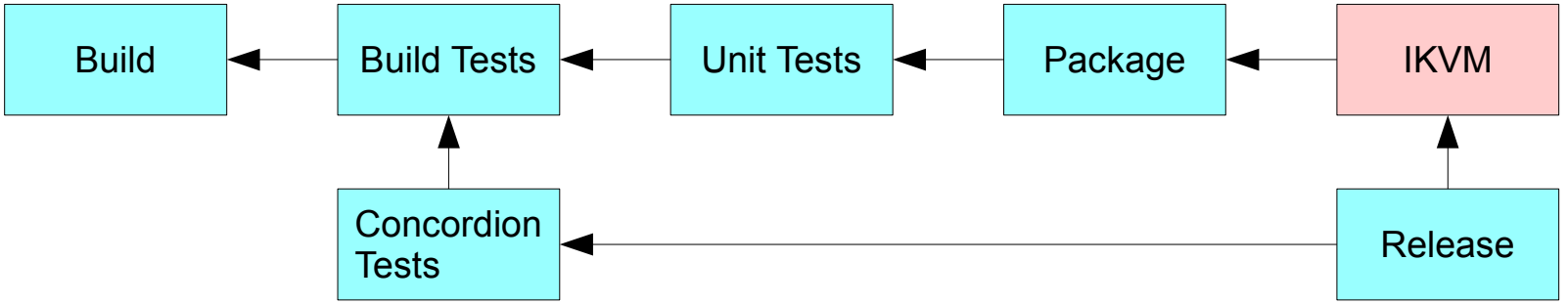
Analisi del problema – Dipendenze interne (1)

- Processi di sviluppo per le singole piattaforme
  - numero anche elevato di passi (target) ordinati secondo dipendenze fisiche e logiche in modo da garantire la consistenza delle operazioni



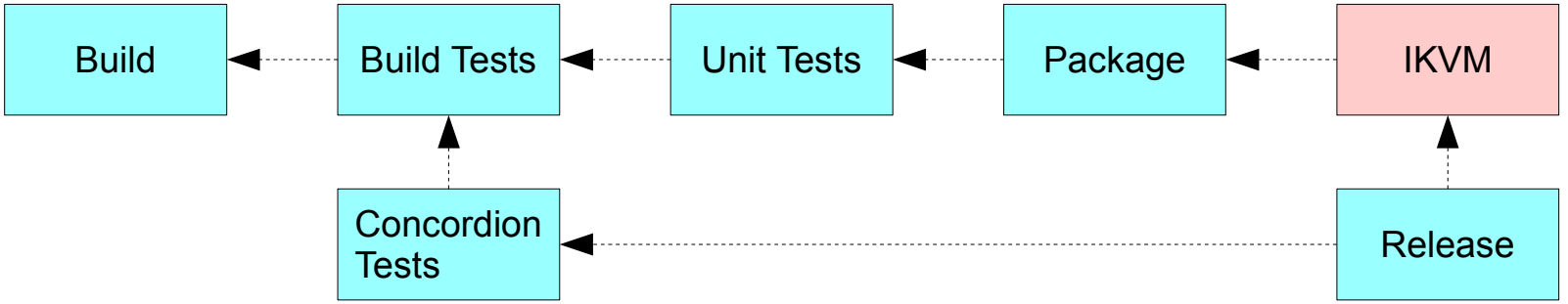
- Processo di sviluppo Java per tuProlog
  - Tutti i target necessitano dell'output di un target da cui dipendono
  - Garanzia di consistenza

Analisi del problema – Dipendenze interne (2)



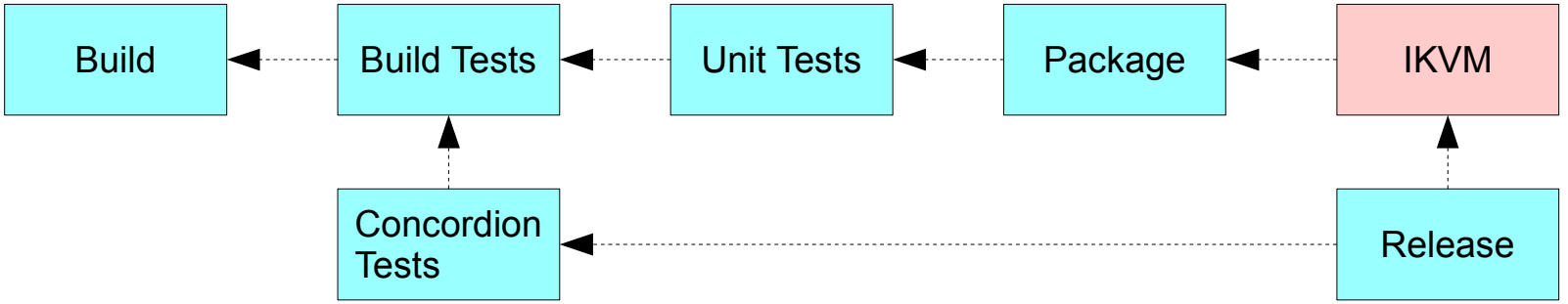
- Efficienza e flessibilità del processo
  - Dipendenze troppo rigide possono portare ad azioni non necessarie
    - Es. Per effettuare il deployment dell'applicazione i target strettamente necessari sono, in ordine, Build → Package → Release
  - Esecuzione ricorsiva di tutte le dipendenze
    - Non è possibile eseguire un target sfruttando l'output di un'esecuzione precedente. Ad esempio:
      - Esecuzione 1: Build → Package
      - Esecuzione 2: Release

Analisi del problema – Dipendenze “soft” (1)



- Eliminazione dell'esecuzione automatica delle dipendenze
  - Esecuzione dei target strettamente necessari
  - Possibilità di esecuzione di un singolo target in maniera indipendente
- Maggiore complessità
  - L'utente deve specificare quali target eseguire e in quale ordine
  - Per un utilizzo corretto è necessaria una conoscenza dettagliata dell'architettura del processo
- Nessuna garanzia di consistenza
  - I target sono indipendenti e quindi qualsiasi sequenza è ammissibile

Analisi del problema – Dipendenze “soft” (2)

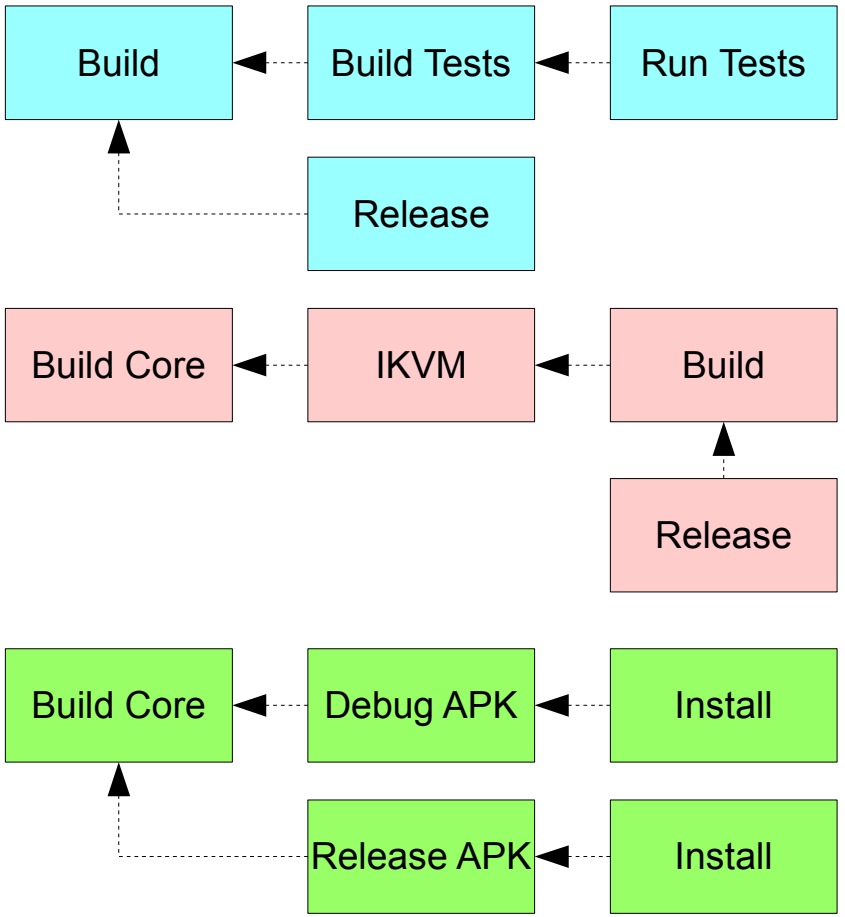


- Per gestire la maggiore complessità è necessario
  - fornire una visione globale dell'architettura del processo
  - permettere la selezione e l'esecuzione di un sotto-insieme dei target
- E' necessario garantire la consistenza delle operazioni
  - I target specificati devono essere eseguiti secondo l'ordine dettato dalle (precedenti) dipendenze
- Un'interfaccia grafica è uno strumento idoneo a gestire complessità e consistenza

Analisi del problema - Configurazione

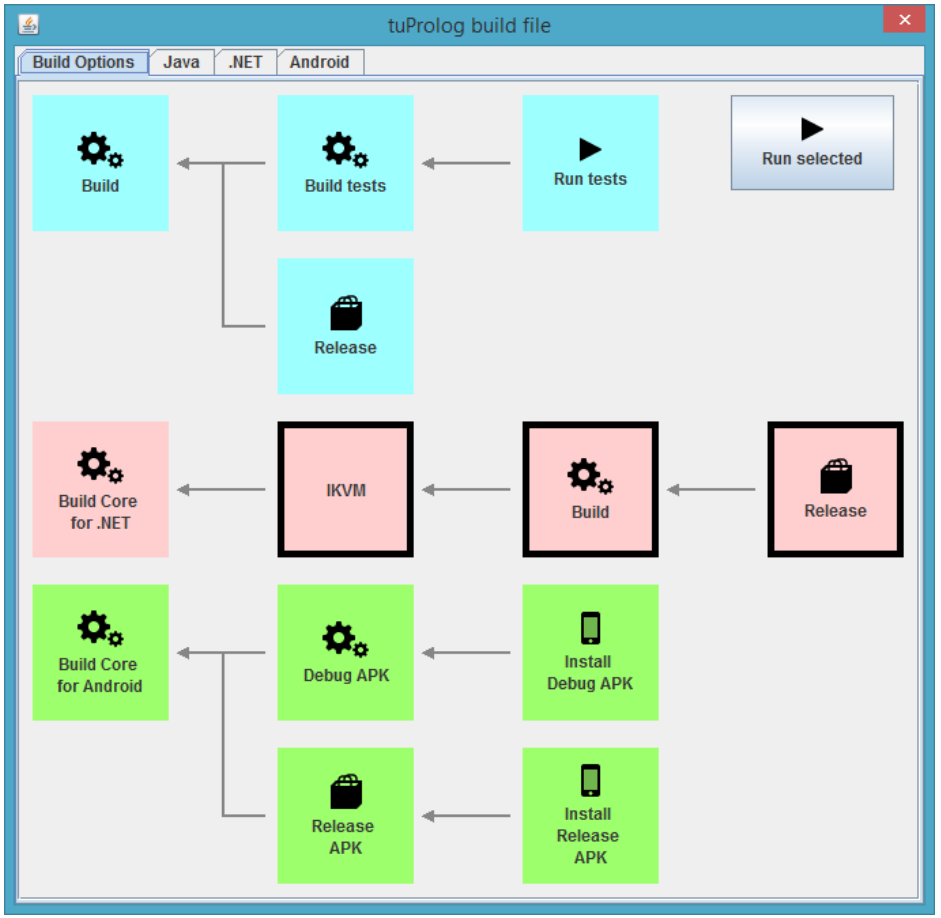
- Integrazione di molteplici “sotto-processi” relativi alle piattaforme ed ai linguaggi considerati
  - Un insieme di tool per ogni “sotto-processo”
  - Es. tuProlog
    - JDK (versioni multiple), IKVM, MSBuild, Android SDK, ...
    - Apache ANT
- Complessità di configurazione
  - Ogni tool utilizzato necessita di un insieme di parametri di esecuzione
- L'interfaccia grafica può essere utilizzata anche per gestire questo aspetto del processo di sviluppo

Nuova architettura



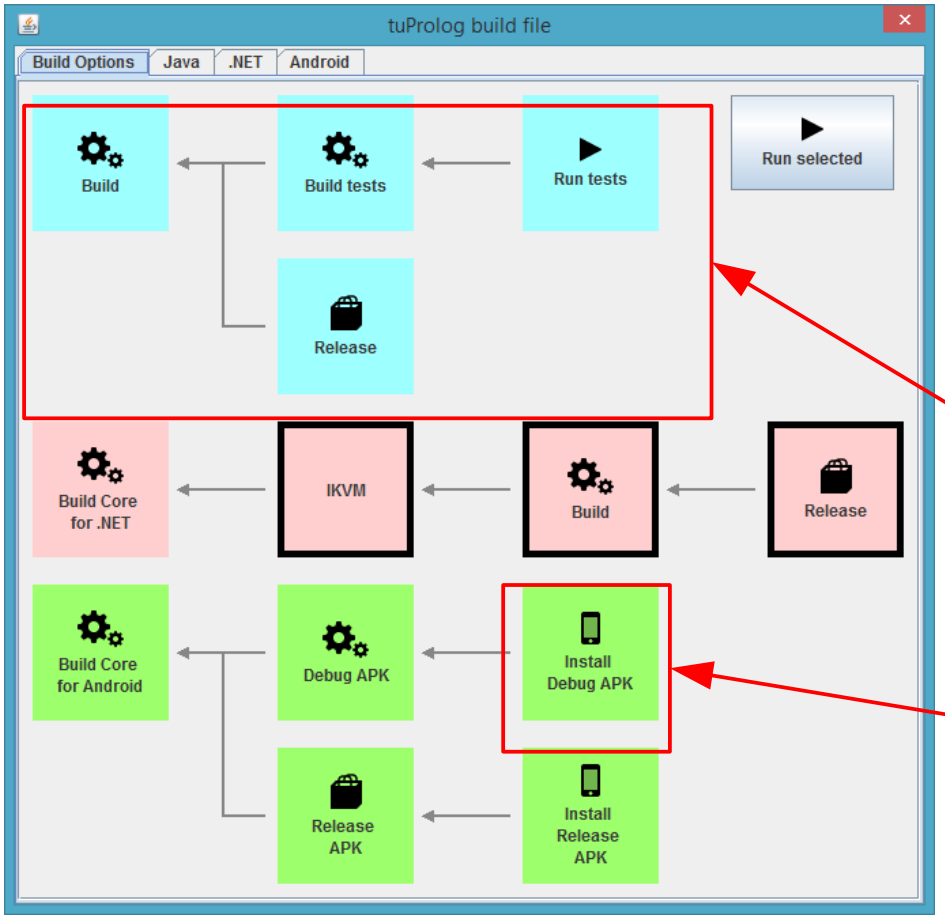
- Piattaforme separate a livello logico
  - Integrazione di nuove piattaforme non invasiva
  - Integrazione di nuove feature o modifica di quelle esistenti locali alla piattaforma considerata
- Dipendenze soft per ottenere maggiore flessibilità ed efficienza
- Complessità e consistenza gestite tramite un'interfaccia grafica

Complessità e consistenza

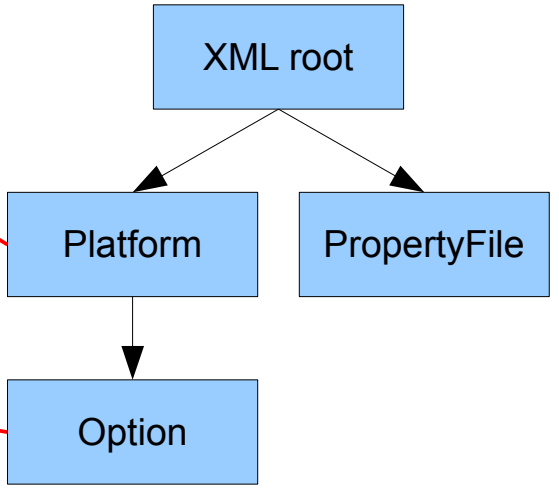


- Complessità
  - Visione globale dell'architettura del processo
  - Meccanismi di selezione dei target singola e multipla
- Consistenza
  - Esecuzione ordinata dei target selezionati secondo le dipendenze

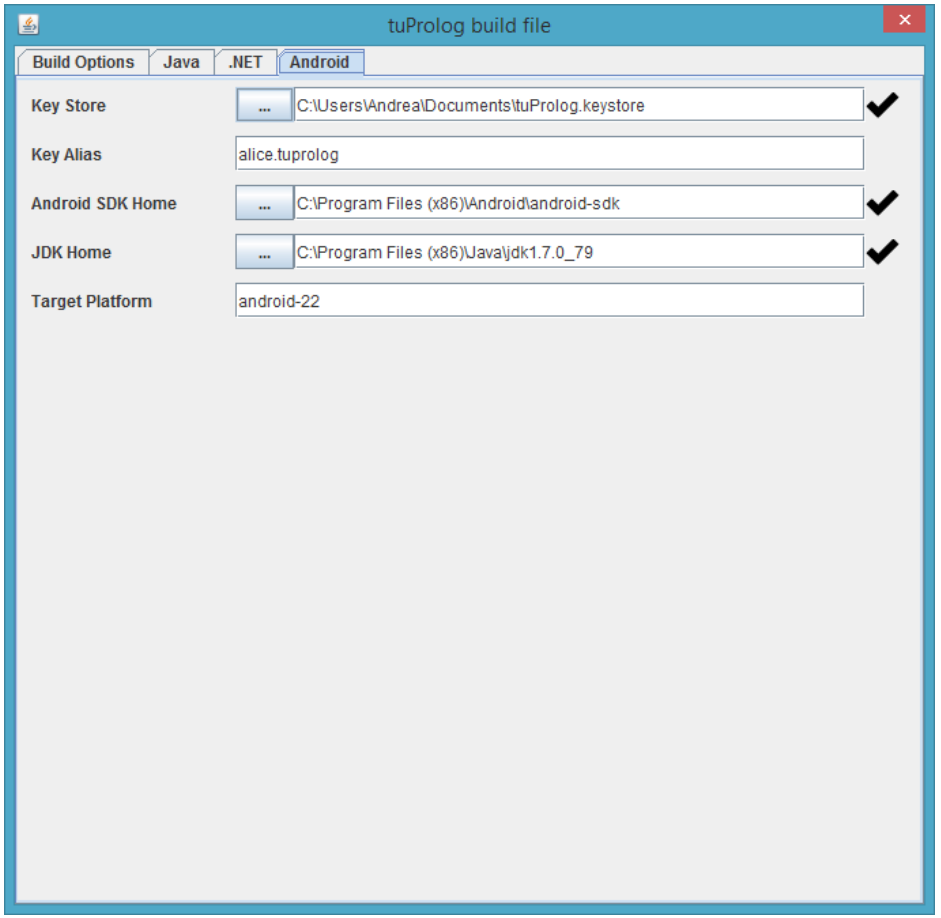
Configurazione ed estendibiltà



- L'interfaccia è facilmente riconfigurabile tramite XML in modo da
  - Integrare nuove piattaforme
  - Riorganizzare le piattaforme esistenti



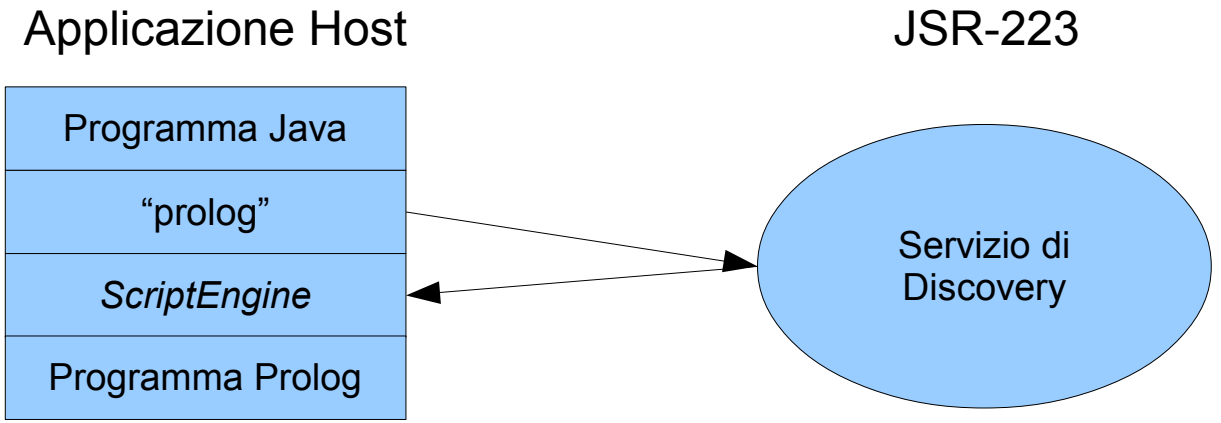
Configurazione dei “sotto-processi”



- Schede per la gestione di tutti i parametri di configurazione
- Basati su file di configurazione in formato Java Properties
- Gestione dei tipi di dato
  - Stringhe, File, Cartelle, ...
  - Custom (via Java)

Specifica JSR-223

- Estensione della capacità di programmazione multi-paradigma di tuProlog



Caratteristiche della specifica

- Interfacce standard per la valutazione di script nell'applicazione host
- Interoperabilità tra l'applicazione host e l'ambiente di esecuzione dello script

Applicazione al caso tuProlog

- Utilizzo di tuProlog come script engine in applicazioni Java in modo standard
- Possibilità di programmazione multi-paradigma utilizzando i linguaggi Java (applicazone host) e Prolog

Conclusioni (1)

- L'architettura del processo di sviluppo aumenta di complessità in funzione dei linguaggi utilizzati e delle piattaforme supportate.
  - La difficoltà di gestione delle dipendenze aumenta con l'integrazione di nuove piattaforme e funzionalità (target)
  - Ogni piattaforma e linguaggio comportano l'integrazione del relativo processo di sviluppo che necessita di essere configurato
- La complessità del processo ha portato alla necessità di fornire
  - Una visione globale dell'architettura del processo
  - Uno strumento per la configurazione

Conclusioni (2)

- L'architettura proposta
  - limita la complessità del sistema di dipendenze che si può creare separando logicamente le piattaforme
  - è estendibile per supportare nuove piattaforme lasciando inalterati i processi esistenti
  - è riconfigurabile localmente alla piattaforma considerata
- L'interfaccia grafica
  - si integra con il processo fornendo all'utente una visione globale dell'architettura e facilitando la configurazione
  - garantisce la consistenza delle operazioni relativamente ai target selezionati
  - è utilizzabile in altre applicazioni il cui processo di sviluppo è implementato tramite Apache ANT, e quindi costituisce un approccio generale al problema

Grazie per l'attenzione!