ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INFORMATICA, SCIENZE E INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Modularizzazione dell'interprete tuProlog su piattaforma Java 9

CANDIDATA

Maria Russo

RELATORE

Prof. Enrico Denti

Contesto

- Java 9 Platform Module System:
  - migliore gestione delle applicazioni a livello di sviluppo, efficienza, manutenzione, aggiornamento
  - permette la creazione di runtime personalizzate
- tuProlog:
  - interprete Prolog sviluppato presso l'Università di Bologna
  - leggero ed open-source per applicazioni ed infrastrutture distribuite
  - core minimale
  - multi-linguaggio, multi-paradigma
  - multi-piattaforma: Java, .NET, Android, iOS, come plugin per Eclipse

Obbiettivo

- Analizzare le dipendenze all'interno di tuProlog (versione base per Java)
- Analizzare le dipendenze delle versioni per le altre piattaforme
  - .NET, Android, Eclipse plugin, iOS
- Definire una proposta di modularizzazione di tuProlog che:
  - catturi al meglio le dipendenze in opportuni moduli
    - evitando cicli di dipendenza
    - sfruttando efficacemente le caratteristiche dei moduli Java
  - rispettando e valorizzando la struttura logica del progetto tuProlog

Moduli Java

- Insieme di package
- [risorse]
- Module Declaration:

```
module moduleName
{
      // Module Directive
}
```

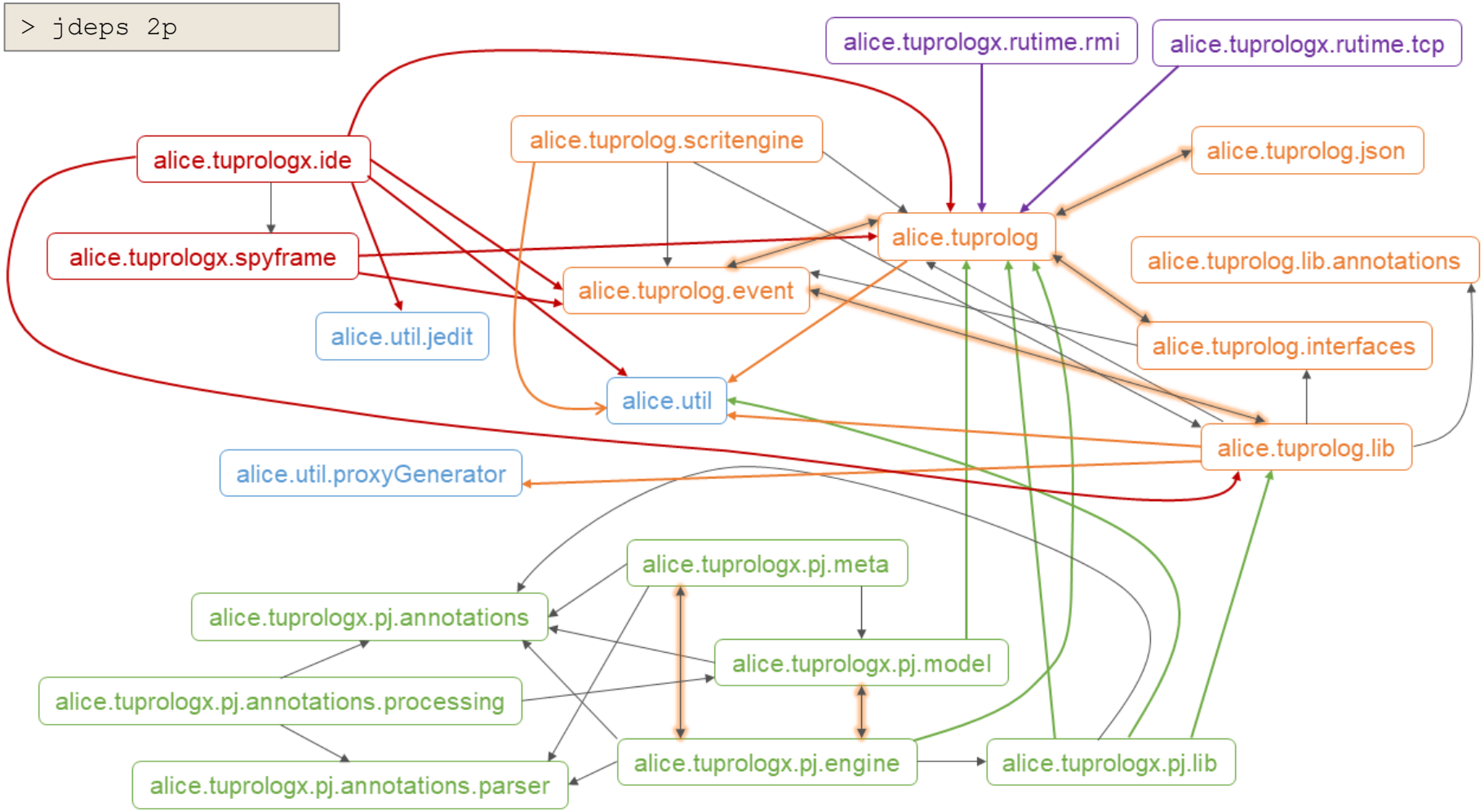
  - Module Directive principali:
    - `exports` ed `exports…to`
    - `requires` e `requires transitive`

```
module moduleName
{
      exports packageName1, ...;
      exports packageName to moduleName1, ...;
}
```

```
module moduleName
{
      requires moduleName1, ...;
      requires transitive moduleName;
}
```

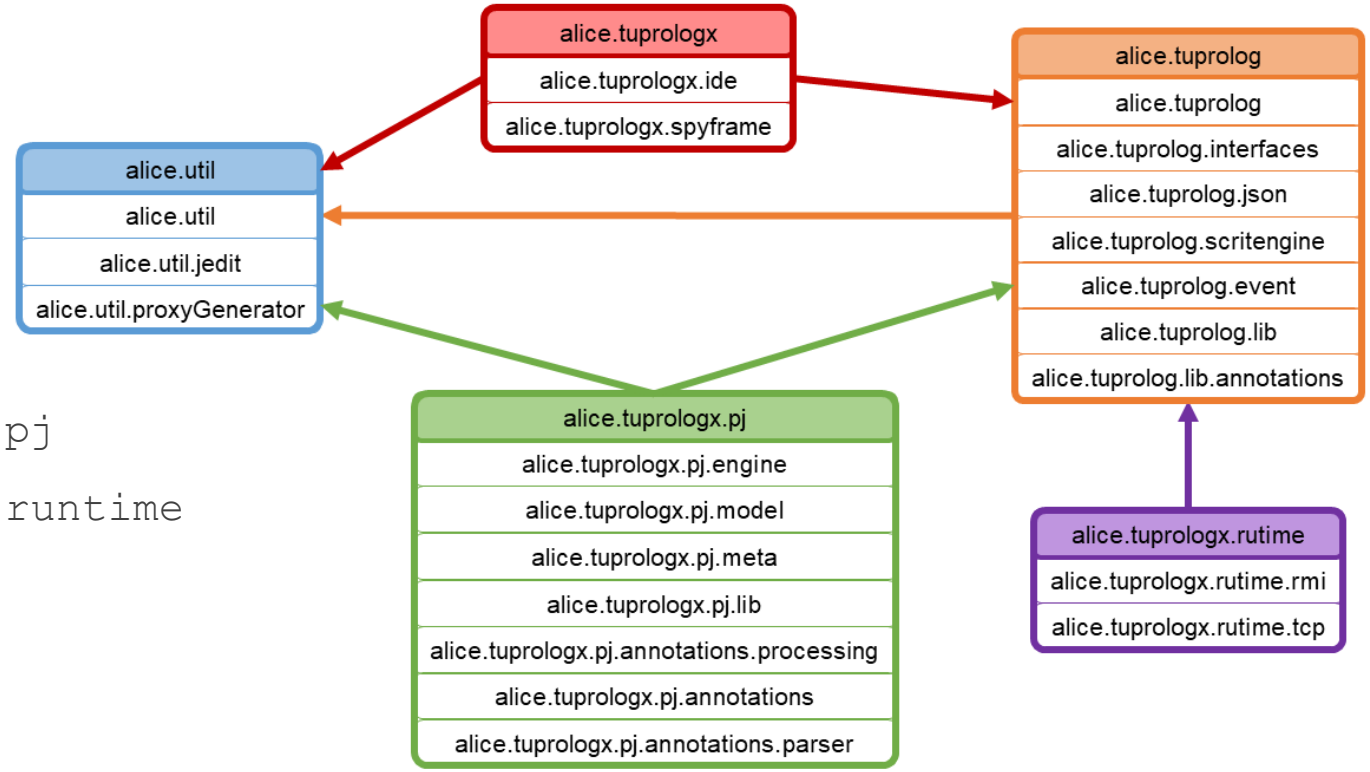
Dipendenze in tuProlog per Java

```
> jdeps 2p
```



Progettazione Moduli

- 5 moduli:
  - `alice.tuprolog`
  - `alice.util`
  - `alice.tuprologx`
  - `alice.tuprologx.pj`
  - `alice.tuprologx.runtime`



- IDE scelto per modularizzare:
  - Eclipse Photon (un progetto per modulo)

Creazione Moduli: `module-info.java`

```
module alice.tuprologx
{
   /* requires of modules used by this module */
   requires alice.tuprolog;
   /* the alice.util module is required by requires transitive directive from alice.tuprolog module */

   /* requires of libraries used by this module */
   requires java.desktop;
   requires autocomplete;
   requires rsyntaxtextarea;
   /* mscorlib is required by requires transitive directive from alice.util module */
}
```

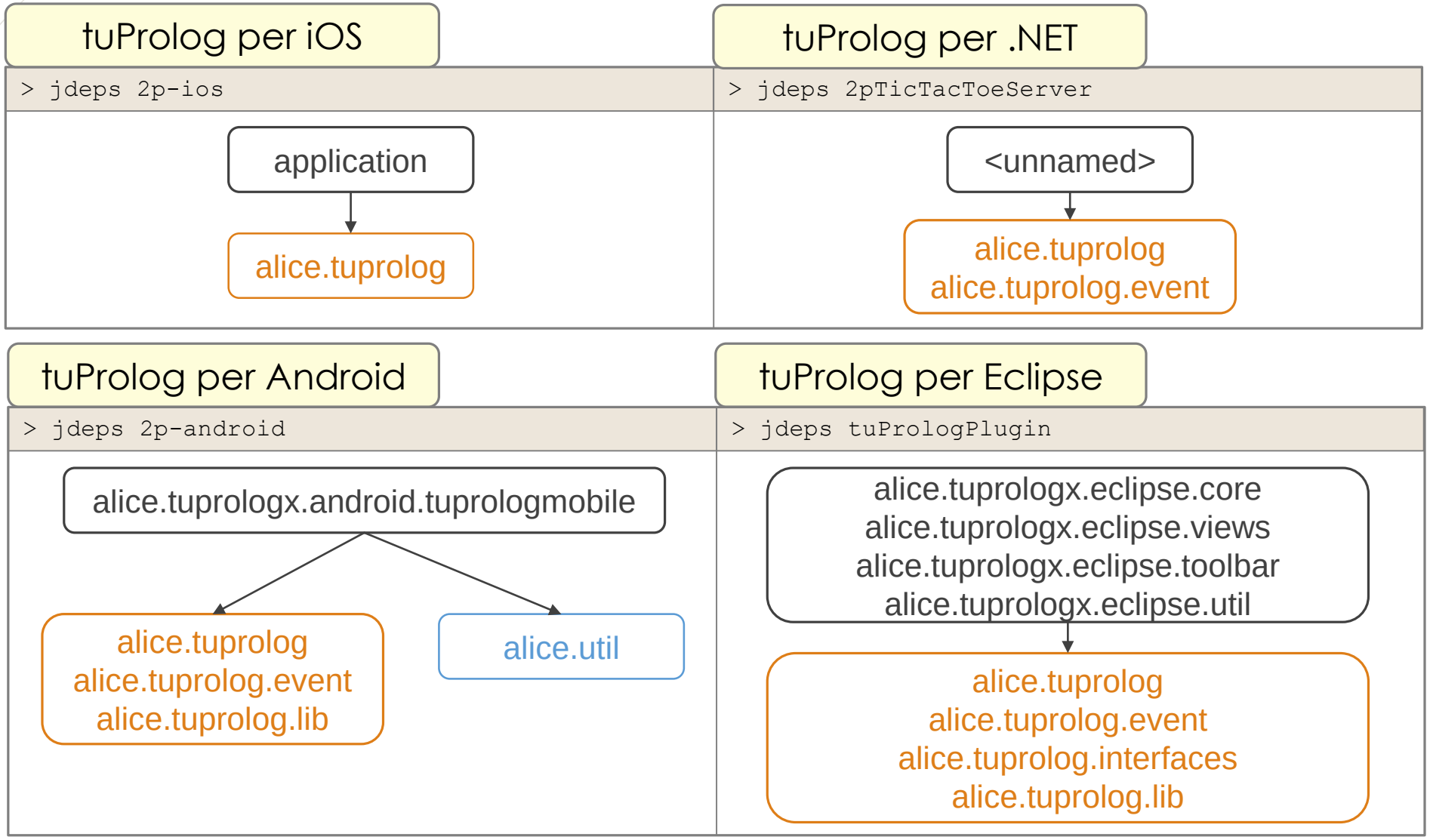
```
module alice.util
{
   /* selective export of module's packages */
   exports alice.util to alice.tuprologx, alice.tuprolog, alice.tuprologx.pj;
   exports alice.util.jedit to alice.tuprologx;
   exports alice.util.proxyGenerator to alice.tuprolog;

   /* requires of libraries used by this module */
   requires java.compiler;
   requires java.datatransfer;
   requires java.desktop;
   requires ikvm.api;
   /* with requires transitive directive who requires this module
    * it can also uses the library indicated in the directive */
   requires transitive mscorlib;
}
```

```
module alice.tuprolog
{
   /* selective export of module's packages */
   exports alice.tuprolog.event to alice.tuprologx;
   exports alice.tuprolog to alice.tuprologx.pj, alice.tuprologx.runtime, alice.tuprologx;
   exports alice.tuprolog.lib to alice.tuprologx.pj, alice.tuprologx;

   /* with requires transitive directive who requires this module
    * it can also uses the module indicated in the directive */
   requires transitive alice.util;

   /* requires of libraries used by this module */
   requires java.scripting;
   requires commons.lang3;
   requires gson;
   /* mscorlib is required by requires transitive directive
    * from alice.util module */
}
```

```
module alice.tuprologx.pj
{
   /* requires of modules used by this module */
   requires alice.tuprolog;
   /* the alice.util module is required by requires transitive directive from alice.tuprolog module */

   /* requires of libraries used in this module */
   requires java.compiler;
   requires java.desktop;
   requires javassist;
}
```

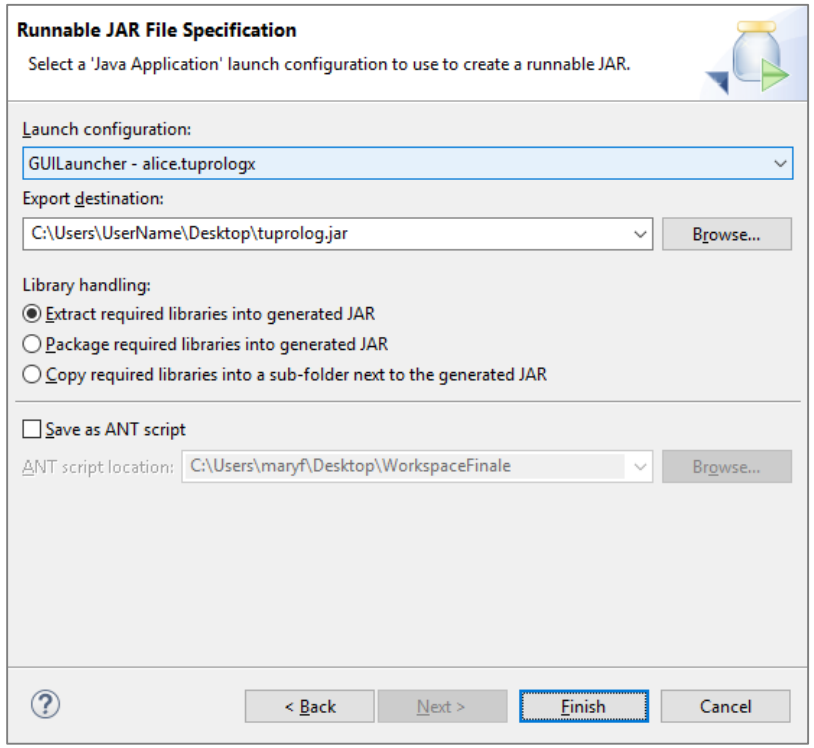
```
module alice.tuprologx.runtime
{
   /* requires of modules used by this module */
   requires alice.tuprolog;

   /* requires of libraries used in this module */
   requires java.rmi;
}
```

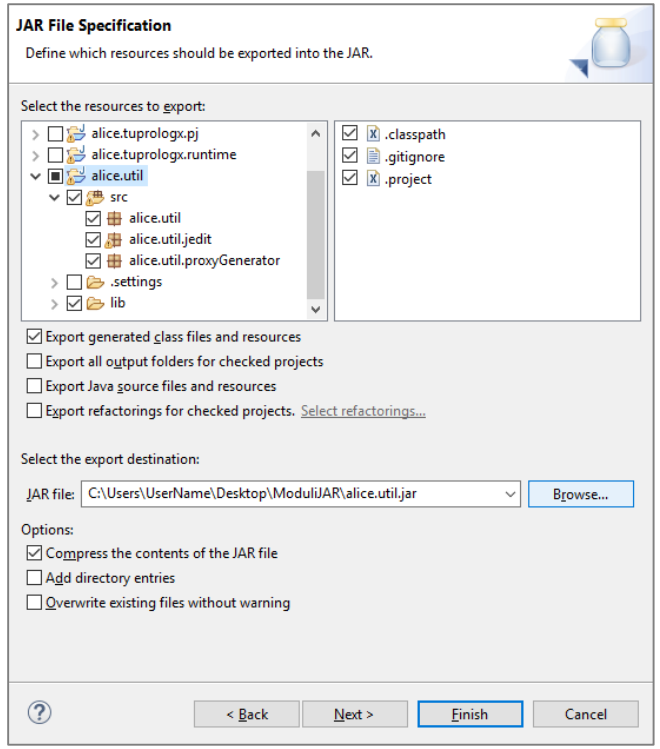
Studio dipendenze tuProlog per altre piattaforme



Risultato finale: tuProlog modulare

- Progetto in 5 moduli:
  - risorse: per limitazioni di Eclipse inserite nel modulo base `alice.tuprolog`
- Realizzazione file JAR eseguibile di tuProlog modulare tramite Eclipse
- Possibile realizzare JAR dei moduli utili alle applicazioni tuProlog per le altre piattaforme





Conclusione

- È stato possibile realizzare 5 moduli per l'applicazione tuProlog in Java
  - Ciò ha permesso di realizzare librerie personalizzate per ciascuna delle applicazioni tuProlog presenti per le altre piattaforme
- Un'applicazione modulare può risultare vantaggiosa per quanto riguarda la fase di sviluppo e manutenzione delle future versioni di tuProlog
  - ad esempio: permettendo di modificare determinate caratteristiche o funzionalità semplicemente intervenendo sui moduli
- Un miglioramento futuro potrebbe avvenire:
  - in seguito ad un aggiornamento di Eclipse che risolva il problema che obbliga la realizzazione di un progetto per ogni modulo
  - all'eventuale scelta di un diverso ambiente di sviluppo, quale IntelliJ, che supporti nativamente i moduli senza costringere a partizionare artificiosamente il progetto in progetti mutuamente dipendenti