

AOP: domande e risposte

- AOP va bene solo per tracing e logging?
 - NO, va bene per molti altri concern che si rivelano crosscutting
 - È comunque non banale individuare i concern crosscutting
 - Vedere esempi: editor per figure [Lezione 8], soluzioni ad aspetti dei design pattern Adapter [Lezione 11] e Factory Method [Lezione 12]
- AOP va bene solo per supportare l'evoluzione di sistemi OO preesistenti?
 - NO, un sistema AOP può essere sviluppato senza partire da codice esistente e l'AOP fornisce un aiuto significativo per realizzare sistemi modulari
- L'AOP è la tecnica migliore per l'evoluzione di sistemi OO?
 - NO, dipende da ciò che si vuol fare. L'ereditarietà, CORBA, etc. potrebbero essere più efficaci in molti casi

Ing. E. Tramontana - Domande AOP - 29-Nov-06 1

AOP: domande e risposte

- AOP si usa per migliorare sistemi OO malfatti (poco modulari)?
 - NO, gli aspetti dovrebbero essere realizzati per sistemi OO benfatti
 - Ovvero, le tecniche OO dovrebbero essere usate efficacemente e l'AOP può ulteriormente migliorare la modularità dei sistemi OO benfatti
- Come è possibile "isolare" un aspetto e quindi un concern crosscutting?
 - Alcuni concern crosscutting sono "classici", quindi se si ha necessità di logging, tracing, etc. è opportuno pensare ad una loro realizzazione ad aspetti
 - Altri concern "classici" che, supportando varie classi di un sistema, sono spesso crosscutting: sincronizzazione, autorizzazione, politiche di verifica dati inseriti, etc.
- E per i concern "non-classici"?
 - Ciascuna classe dovrebbe essere analizzata per determinare quali sue operazioni contribuiscono a realizzare compiti aggiuntivi rispetto alla finalità della classe
 - Le operazioni aggiuntive dovrebbero essere spostate fuori dalla classe ed in alcuni casi costituiscono un aspetto

Ing. E. Tramontana - Domande AOP - 29-Nov-06 2

AOP: domande e risposte

- Come si può determinare se una classe realizza compiti aggiuntivi rispetto alla sua finalità?
 - Se tutte le operazioni della classe sono logicamente relazionate al nome della classe ed alla sua interfaccia non vi sono compiti aggiuntivi realizzati dalla classe
 - Es. per la classe Line in [Lezione 8]
 - Il suo nome e la sua interfaccia non dicono che essa debba aggiornare la classe Display
 - L'interazione con la classe Display è un compito aggiuntivo
 - Viceversa, l'uso della classe Point all'interno della classe Line è pertinente (si pensi alla definizione geometrica di linea-segmento)
 - In altre parole, si dovrebbe valutare ciascuna operazione all'interno della classe sulla base del nome dato alla classe e della sua interfaccia

Ing. E. Tramontana - Domande AOP - 29-Nov-06 3

Modularità: OOP e AOP

- Nei sistemi OO delegare, ovvero chiamare metodi, è un buon modo per poter sviluppare funzionalità diverse in classi separate, ma che interagiscono
 - Delegare aiuta a costruire sistemi modulari
- Nei sistemi con aspetti si può valutare se rimuovere alcune chiamate a metodo da una classe
 - Implementando un appropriato aspetto, che tramite un pointcut cattura l'esecuzione e passa il controllo ad un'altra classe
 - Si parla di refactoring da OOP a AOP tramite rimozione di chiamate a metodi
 - Es. editor per figure

Ing. E. Tramontana - Domande AOP - 29-Nov-06 4