

Contents

1	crosscutting concerns	1
1.1	what is it	1
1.2	options to deal with it	1
2	context-oriented approach	3
2.1	pros	3
2.2	cons	3
2.3	public acceptance	3
3	aspect-oriented vs context-oriented	3
3.1	pros	3
3.2	cons	3
3.3	public acceptance	3
4	feature-oriented vs context-oriented	3
4.1	pros	3
4.2	cons	3
4.3	public acceptance	3
5	develop the soldier app in contextJ	3
5.1	compare with the solution made in class	3
5.2	pros	3
5.3	cons	3

1 crosscutting concerns

1.1 what is it

1.2 options to deal with it

- TemplateMetaprogramming, augmented with PolicyObjects (such as StandardTemplateLibrary allocators)
- MetaObjectProtocols, in particular things like before functions, after functions, etc.
- Decorators (both DecoratorPattern and the language feature in Java), delegation in general.
- TemplateMethodPattern (i.e. use of HookMethods)

- RPC tools
- Many of the CreationalPatterns can be used to allow variation of some cross-cutting concern at runtime.
- MultipleInheritance and MixIns
- ContextObject or ExplicitManagementOfImplicitContext
- A programming paradigm dealing with this stuff is AspectOriented-Programming.

bad example of dealing:

- GrandCentralStation (aggregating lots of independent domain functionality into a BigBallOfMud just so the CrossCuttingConcern(s) can be dealt with in one place).

2 context-oriented aproach

2.1 pros

2.2 cons

2.3 public acceptance

3 aspect-oriented vs context-oriencted

3.1 pros

3.2 cons

3.3 public acceptance

4 feature-oriented vs context-oriented

4.1 pros

4.2 cons

4.3 public acceptance

5 develop the soldier app in contextJ

5.1 compare with the solution made in class

5.2 pros

5.3 cons