Entity Perspectives

Asked 16 years, 2 months ago Modified 16 years, 2 months ago

Viewed 175 times

4

Context: So, I am attempting to build a ridiculously complex domain model. Talking with the users, there are multiple ways the system will be used (For example: one way is from the perspective of the employee; another way is from the perspective of the employer). In actuality, they are multiple applications reusing the core domain.

Question: Is it wrong to create the domain from multiple perspectives? For instance, to build the domain as the business is run would mean creating all kinds of relationship classes that carry extra information about the entity and so on... However, when running from a particular perspective, the picture is much clearer because most of the relationship classes can be rolled into their parent entity.

Ideas? Rebuttals? Am I completely off-base?

dns domain-driven-design entity

Share

Improve this question

Follow

edited Oct 14, 2008 at 13:03

Kev

120k ● 53 ● 305 ● 391

asked Oct 14, 2008 at 12:53

Craig Wilson

12.6k ● 3 ● 44 ● 45

2 Answers

Sorted by: Highest score (default)

4

No, this is absolutely appropriate. Evan's calls these 'perspectives' Bounded Contexts in his book.

Instead of having a really complex domain model, try building a simple one that addresses the domain in a specific way really well. Bounded Contexts can be linked together to form a web of smaller, simpler, direct domain models.

Share Improve this answer

Follow

answered Oct 14, 2008 at 13:08

Ben Scheirman

40.9k ● 21 ● 103 ● 139

I have Eric's book and apparently forgot about that chapter. Thanks for the direction. – Craig Wilson Oct 14, 2008 at 17:39

2

I don't think you should try and predefine the 'core domain' up front. Let it emerge over the development process. Additionally, anything that is not common to the 2 (or more) perspectives should not be in the 'core'.

For instance, build a portion of the system from the employeer perspective. This might prompt you to create

entities like 'Project', 'Task', and 'Customer'. Then build a portion from the employeer perspective. This might prompt you to build new entities, and to reuse 'Project' and 'Task'. That's when I'd move 'Project' and 'Task' to the 'core library' shared by the rest of the system.

Sometimes you'll find common entities, but related in different ways. In that case the relationship should be injected by the context instead of being built-in with the entities themselves.

Share Improve this answer

Follow

answered Oct 14, 2008 at 13:07

FOR

4,358 2 27 36