XOM, o XML será produzido (por exemplo).

É sobre isso que o autor incita quando diz que o projeto de herança é importante no desenvolvimento. Não queremos, como usuários, perder muito tempo com coisas específicas de uma linguagem. Queremos utilizar diretamente as ferramentas que elas nos proporcionam de acordo com nossas necessidades.

Apesar disso, diz que sua mente ainda insiste em preferir herança aberta. Uma alternativa para apaziguar a discussão seria o modo como você sai do seu padrão para sobrescrever algo que não foi antes desenvolvido. Se você mantém a classe concreta, então o compilador só permite herança normal, mas se você utilizar o mecanismo abstrato, o compilador permitirá que você faça mais alteração do que se já fosse um bloco selado, concreto.

Martin Fowler diz, então, que há muito o que pensar sobre interface, tanto como por instância como por herança. Precisamos de mais ideias e estudos a respeito de providers e consumers, utilizados em pacotes de serviços, para que possamos fazer um melhor uso das interfaces.

A herança aberta é o oposto do projeto de Herança (e é por isso que há tanto embate entre elas). Apesar de serem opostas, a comunidade não costuma discordar quanto aos perigos da herança. Por ser uma relação muito íntima, é dificultosa para os desavisados.

Um trecho que nos ajuda a entender melhor a diferença entre herança aberta e fechada é que o projeto de herança, á herança fechada exige muito