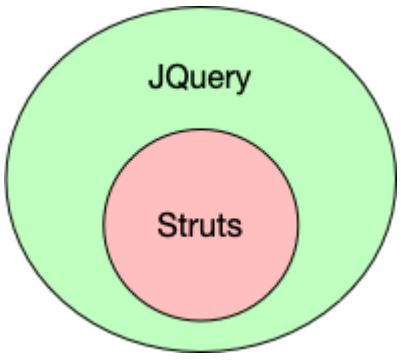
¿Frameworks ciclo de vida? . ¿Cuál es el ciclo de vida de un framework? . Esta es una pregunta interesante porque en muchas entrevistas que suelo realizar mucha gente hace hincapié en que conoce. tal o cual framework y lo considera lo más importante a nivel de conocimiento . Ahora bien los frameworks como herramientas que se encargan de simplificar los desarrollos y aplicar buenas prácticas tienen un ciclo de vida.

Este ciclo de vida puede variar y normalmente si es un Framework de cliente es probable que dure algo menos que un Framework de servidor. ¿Cómo podemos categorizar los diversos conceptos de desarrollo a nivel de ciclo de vida? . Esta es una pregunta muy interesante.

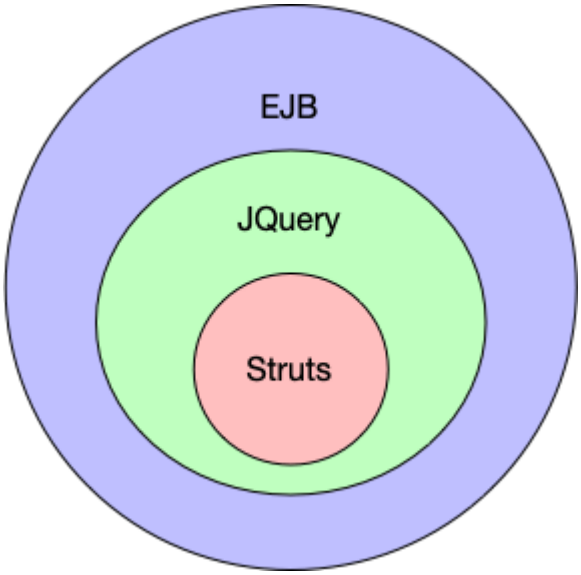
Frameworks ciclo de vida y sus limitaciones

Los frameworks son quizás el concepto más de moda en programación ya que aumentan la productividad. Ahora bien ¿Cuando duran en el tiempo?. La realidad es que no lo sabemos a ciencia cierta y depende de muchas variables. Ahora bien hay algunas cosas que si suelen ser habituales en esta gestión del ciclo de vida. Por mucho que no nos guste Struts ya hace tiempo que paso a la historia y se usa sobre todo en mantenimientos. Sin embargo una librería como jQuery quizás hoy no tenga la fama de otros días debido a React y los componentes pero sigue siendo bastante operativa. Por lo tanto es razonable pensar que una librería vive más que un framework. Siempre habrá excepciones pero es bastante común que así sea.



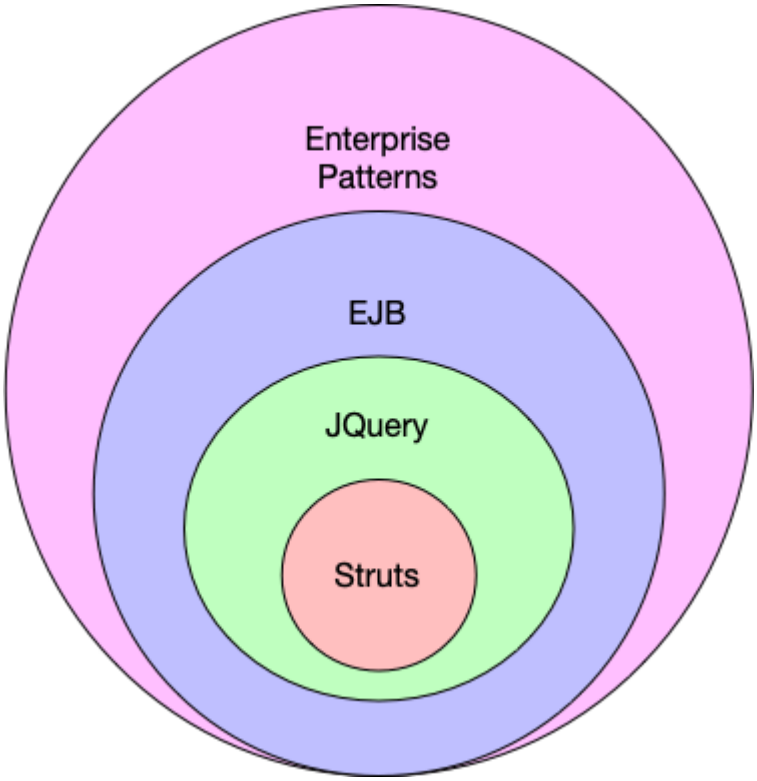
Framework vs Specificación

¿Hay cosas que pueden durar mas que una librería ? . Por supuesto que sí una de ellas es lo que habitualmente se denomina una especificación como por ejemplo es EJB. Nos pueden gustar mas o menos los EJBs pero son una especificación que varios fabricantes implementan y llevamos con ella unos 25 años , más de lo que han durado muchas librerías.



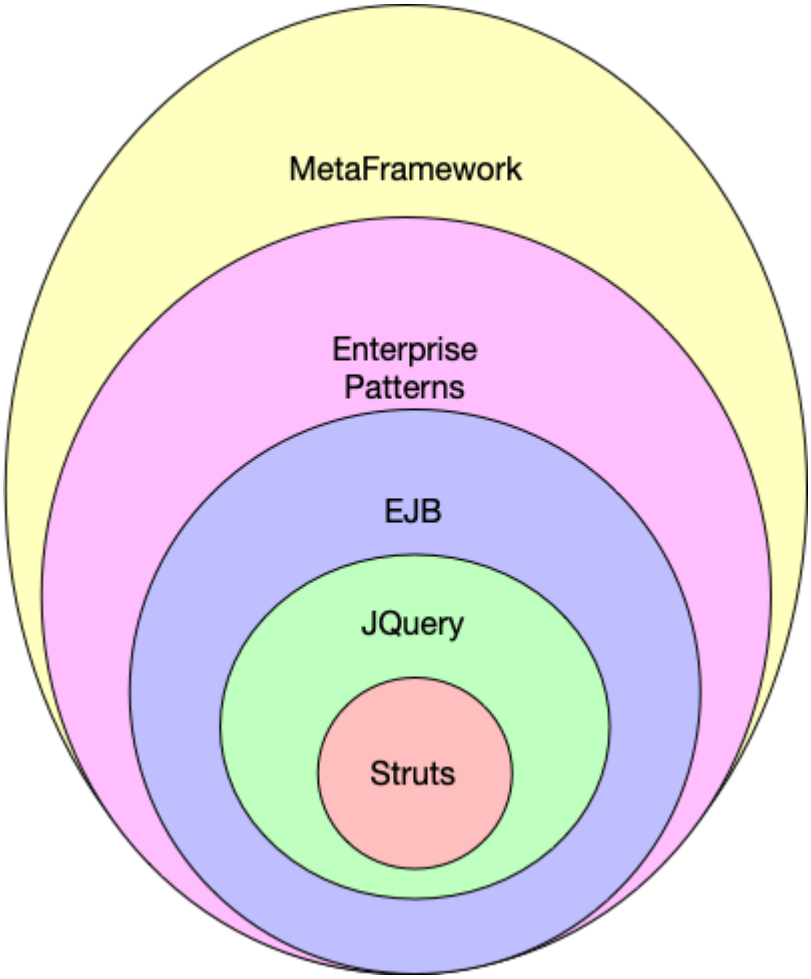
Enterprise Design Pattern

Los patrones de diseño son conceptos que suelen tener un ciclo de vida mayor que las Especificaciones aunque no siempre es así depende mucho del patrón de diseño. Hay algunos que han pasado a mejor vida como Session Facade y otros que se encuentran muy vivos como el MVC.



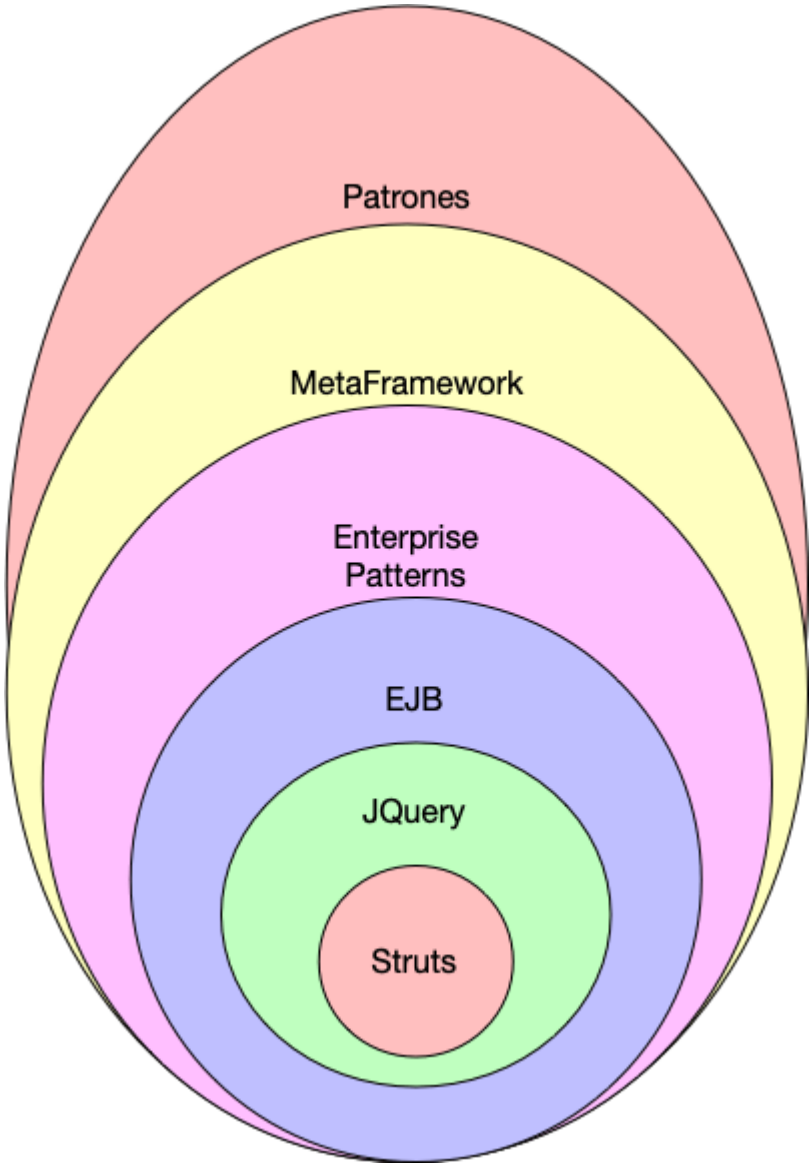
MetaFrameworks

Otro concepto que desde mi punta de vista tiene un ciclo de vida mas largo que un Enterprise Pattern es un MetaFramework (Framework de Frameworks) como puede ser Java EE o Spring Framework. Se tratan de plataformas completas que se encargan de aglutinar un conjunto amplio de frameworks . Net sería algo que encajaría también en este lugar.



Frameworks ciclo de vida y Patrones de diseño

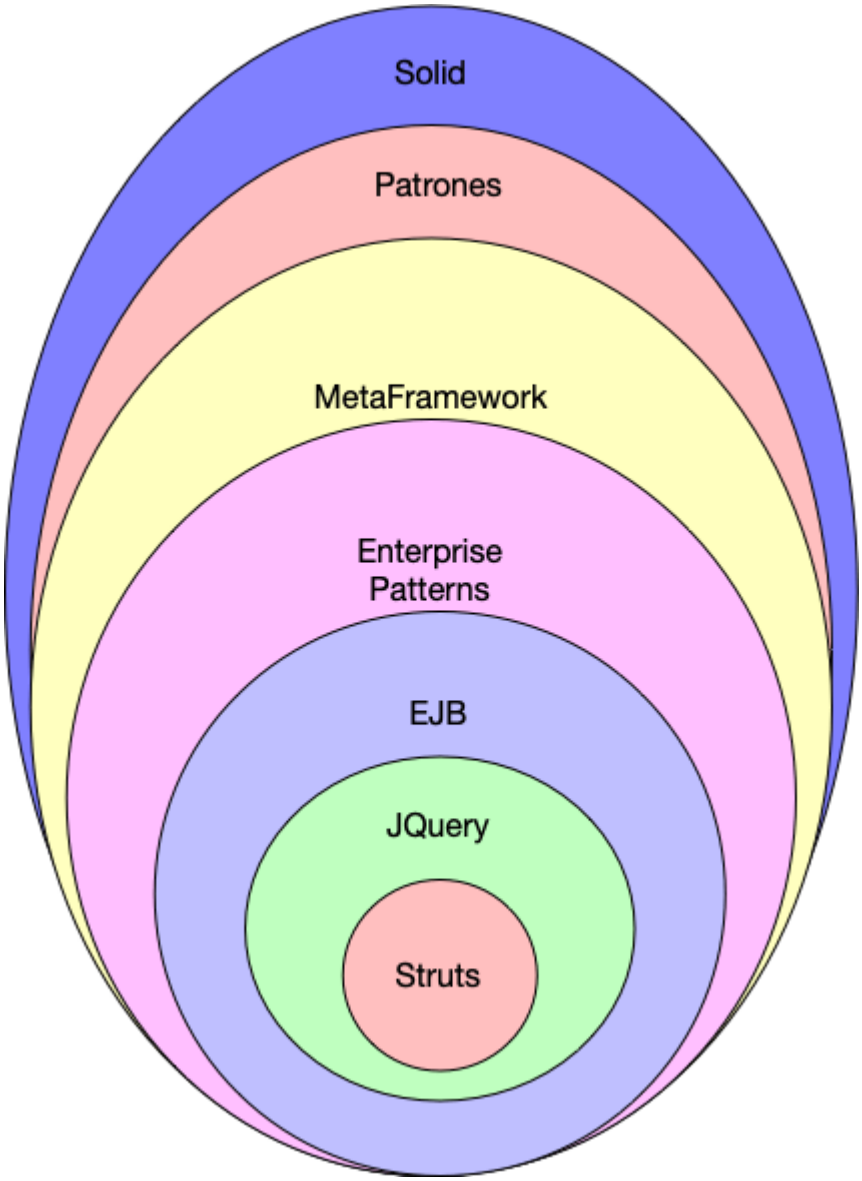
No confundamos los patrones enterprise como DTO , Repository. Con los patrones de diseño clásicos de The Gang of four (la banda de los cuatro) . Singleton , Prototype, Builder, Composite etc. Estos patrones son prácticamente atemporales y su ciclo de vida es mucho mayor.



Principios SOLID

Otro de los conceptos que tiene un mayor ciclo de vida que los propios Patrones de Diseño son los principios sólidos . Son un grupo de principios de ingeniería de software que nos permiten diseñar de mejor forma nuestras aplicaciones. Uno de los mas clásicos es el principio DRY. Dont Repeat YourSelft , hace referencia a que no repitamos código en nuestras aplicaciones este principio es inmutable y tiene desde mi punto de vista una mayor temporalidad que el uso de Patrones de Diseño ,aunque se encuentran cerca. La ventaja es

que los principios sólidos se pueden aplicar a lenguajes que ni siquiera soportan programación orientada a objeto como puede ser un JavaScript ES5. Por lo tanto su ciclo de vida es mayor ya que pueden acceder a un pasado muy lejano como que a veces no pasa con un patrón de diseño que son más bien válidos a partir de que los lenguajes de programación orientada a objeto se imponen.



Principios de Ingenieria de Software

Existen principios de Ingenieria de Software que son todavía más generales que los propios principios sólidos como por ejemplo el Principio de bajo acoplamiento o el de alta cohesión que hacen referencia a principios más universales que nos permiten incluso ir a entornos externos al software. Por ejemplo el principio de bajo acoplamiento se usa habitualmente en el hardware evitando que una maquina tenga excesiva modularización.

Frameworks ciclo de vida y Conclusiones

Tengamos muy en cuenta estos tiempos ya que cada uno de ellos nos puede ayudar a entender que conceptos aguantan mejor el paso del tiempo y son más interesantes de entender que un mero Framework. En muchas ocasiones el Ciclo de Vida de un Framework es limitado.

Otros artículos relacionados

- Frameworks vs Libraries
- Java Collections Framework y su estructura
- ¿Framework vs Frameworkless y cuál elegir?
- El concepto de Framework