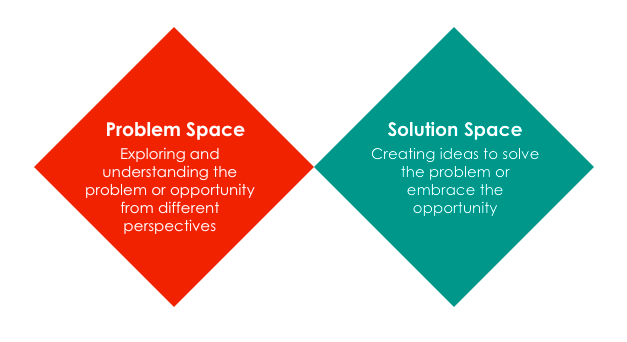
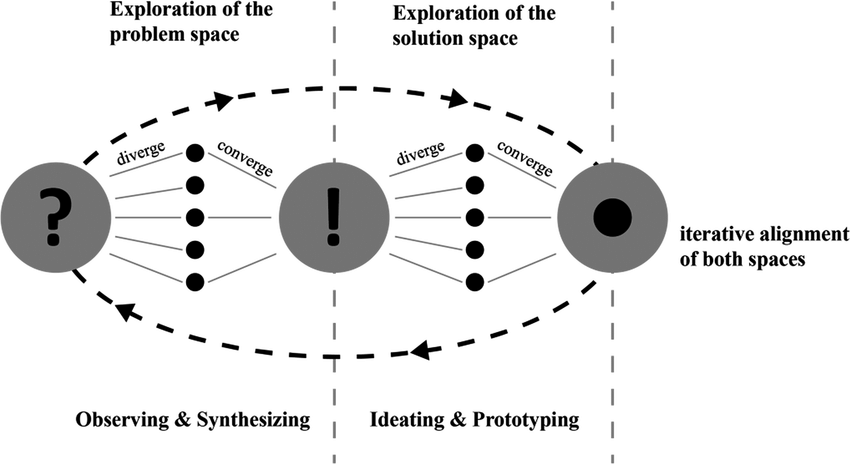
Gap Semántico e Isomorfismo Estructural

Todo programa consta de dos espacios, el espacio de los problemas, y el espacio de las soluciones. Podríamos definir al **espacio de los problemas**, como a todo aquello que presente el motivo por el cual iniciamos el desarrollo de software, podríamos decir que es, la especificación para nuestro programa, o el problema que nuestro programa desea solucionar, e de aquí el nombre. Por el contrario, **el espacio de soluciones**, vendría a ser precisamente, nuestro software o solución a un problema dado.



El desarrollo de software involucra un mapeo de aspectos o características de los objetos y conceptos presentes en el espacio del problema en representaciones abstractas en el espacio de soluciones, de manera tal, que operaciones sobre estas representaciones abstractas correspondan a operaciones en la realidad, a partir de ahí se diseñan algoritmos que ejecutados producirán resultados, resultados estos que podrán ser mapeados físicamente sobre algún suceso en tiempo real en el mundo real, o que serán examinados y mapeados mentalmente por personas sobre resultados en la realidad.

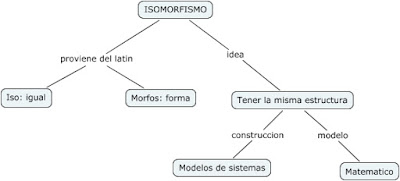


Todo programa es además, en última instancia, una composición de acciones, procesos o procedimientos actuando sobre composiciones o estructuras de datos. Los lenguajes y métodos llamados tradicionales tienden a privilegiar una determinada manera de combinar estos ingredientes básicos. Los lenguajes de uso más general proveen un buen soporte para la representación de acciones o abstracción procedural, pero proveen poco o deficiente soporte para la representación de entidades o conceptos presentes en la realidad, o abstracción de datos, como Grady Booch, señala:  
  
*"Si examinamos los lenguajes naturales, verificamos que todos tienen dos componentes primarios, verbos y sustantivos, debería existir una estructura similar o paralela en los lenguajes simbólicos que provea construcciones para implementar objetos y operaciones.*  
*Todavía la mayoría de los lenguajes son primariamente imperativos, ellos proveen de un rico conjunto de construcción que soportan implementar opiniones, mas en general son significativamente pobres o deficientes en cuanto a la abstracción de objetos de la realidad"*  
  
Mas aún como Tadeo Takahashi nota,  
  
*"--- la topología de los lenguajes funcionales parece indicar que ellos tienen una estructura realmente plana, pero la realidad no es ni plana ni secuencial sino por el contrario, multidimensional y con frecuencia altamente paralela"*  
Teniendo en cuenta lo mencionado anteriormente podemos inferir, que cuanto más cerca conceptualmente estén ambos espacios, más fácil será la tarea de desarrollar software, y obviamente con más facilidad se podrá asegurar la **comprensión, corrección, robustez y extensibilidad** de la aplicación.  
  
Fundamentalmente se trata de un proceso de identificación de abstracciones, si las representaciones de los aspectos más importantes no tuviera una expresión igual o cercana al espacio del problema, entonces sería cada vez más complicado el desarrollo de software.  
  
*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 “ La distancia existente entre la representación del problema y el problema en si, es denominado****Gap Semántico****, el objetivo más claro de los procedimientos, herramientas, métodos y desarrollo del lenguaje, es reducir esta brecha conocida como el gap semántico.*  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Este paralelismo entre los aspectos estables de la caracterización del problema y la estructura de la solución es esencial para la evolución del modelo, este principio, llamado ***"principio de isomorfismo estructural"*** es central en la capacidad de estas abstracciones de encapsular el cambio.

Dos estructuras (sistemas o conjuntos) son isomorfas entre sí cuando a cada elemento de la primera estructura corresponde sólo un elemento de la segunda, y a cada operación (nexo) de una estructura corresponde una única operación (nexo) en la otra, y recíprocamente.

Isomorfismos Estructurales

"isomorfismo estructural; las teorías tienen que ver con la verdad, en la medida en la que hay fenómenos que han de tener algún tipo de isomorfismo con los modelos que caracterizan matemáticamente a dichas teorías"



Es una forma de obtener estructuras de construcción de modelos de sistemas.

Características del isomorfismo

El descubrimiento de un isomorfismo entre dos estructuras significa esencialmente que el estudio de cada una puede reducirse al de la otra, lo que nos da dos puntos de vista diferentes sobre cada cuestión y suele ser esencial en su adecuada comprensión. También significa una analogía como una forma de inferencia lógica basada en la asunción de que dos cosas son la misma en algunos aspectos, aquellos sobre los que está hecha la comparación. En ciencias sociales, un isomorfismo consiste en la aplicación de una ley análoga por no existir una específica o también la comparación de un sistema biológico con un sistema social, cuando se trata de definir la palabra "sistema". Lo es igualmente la imitación o copia de una estructura tribal en un hábitat con estructura urbana.

ISOMORFISMO

Así, las semejanzas son semejanzas de forma más que de contenido: sistemas formalmente idénticos pueden ser aplicados, en efecto, a diferentes dominios. Isomorfo viene de las palabras iso que significa igual y morphê que significa forma. Se define como aquel principio que se aplica igualmente en diferentes ciencias sociales y naturales. La Teoría General de Sistemas busca generalizaciones que refieran a la forma en que están organizados los sistemas. (Isomorfismo) El concepto matemático de isomorfismo pretende captarla idea de tener la misma estructura. Se afirma que sobre la base del desarrollo de modelos formales, con base matemática, dos sistemas, dos realidades, se comportan soportados por el mismo “modelo genérico”, es decir, mismas variables y relaciones. Es como sustituir las variables por las letras del álgebra, permaneciendo las ecuaciones sin variación. El isomorfismo, que es un proceso de homogeneización que puede originarse de dos modos: de la competición y adecuación de organizaciones individuales a cambios en el mercado, lo que se

El término ‘isomorfismo’ significa etimológicamente ‘igual forma’, y con ello se quiere destacar la idea según la cual existen semejanzas y correspondencias formales entre diversos tipos de sistemas, a veces muy aparentemente disímiles entre sí en cuanto al contenido.

ISOMORFISMO

***Esta característica significa construir modelos similares al modelo original, esto con el fin de aumentar o mejorar el desempeño de un sistema.***

Sistema Abstracto

Un ejemplo de ***sistema abstracto es un software***, que es un gran conjunto de líneas de   
código organizadas según reglas, gramáticas y métricas que fueron creadas por el mismo hombre y son conceptos que existen en su mente, pero que en la naturaleza puramente no tienen ningún lugar; son parte y producto del pensamiento pero constituyen un sistema.

Un sistema abstracto requiere operar en conjunto con un sistema concreto para cumplir su función, por ejemplo un software requiere de un sistema físico, una computadora o un dispositivo que sea capaz de ejecutarlo, para cumplir su función, y viceversa, un dispositivo electrónico o un computador no sería de utilidad sin su respectivo sistema abstracto, en este caso el software.

Isomorfismos de diseño de software

Al programar, a menudo puede resolver el mismo problema de diferentes maneras. A veces, las alternativas son isomorfas: se puede ir y venir entre dos alternativas sin pérdida de información.

El libro de **Martin Fowler Refactoring** (Tema de AyED II) contiene varios ejemplos. Por ejemplo, puede aplicar el método de extracción seguido del método en línea y volver a donde comenzó.

Hay muchos otros isomorfismos en la programación. Algunos son morfismos en el mismo lenguaje, como es el caso del ejemplo de C# anterior. Este es también el caso de los isomorfismos en Refactoring, porque una refactorización, por definición, es un cambio aplicado a una base de código en particular, ya sea C #, Java, Ruby o Python.

Otros isomorfismos de programación van entre lenguajes, donde un concepto puede ser modelado de una manera en, digamos, C++, pero de otra manera en Clojure. El presente blog, por ejemplo, contiene varios ejemplos de traducción entre C# y F#, y entre F# y Haskell.

Ser consciente de los isomorfismos de diseño de software puede convertirlo en un mejor programador. Le permitirá seleccionar la mejor alternativa para resolver un problema en particular. Identificar isomorfismos de programación también es importante porque nos permitirá pensar formalmente en la estructura del código al reducir muchas representaciones alternativas a una representación canónica.

Por estas razones, este artículo presenta un catálogo de isomorfismos de diseño de software:

* Isomorfismos unitarios
* Isomorfismos de función
* Isomorfismos de lista de argumentos
* Isomorfismos uncurry
* Isomorfismos de objetos
* Isomorfismo de clase abstracta
* somorfismo herencia-composición
* Isomorfismos Tester-Doer
* Isomorfismos de constructor

En general, he tratado de nombrar cada isomorfismo después de su representación canónica. Por ejemplo, por isomorfismos unitarios, me refiero a los isomorfismos al valor unitario. Sin embargo, no es una estrategia de nomenclatura totalmente coherente.

Existen muchos más isomorfismos de diseño de software, por lo que si revisa este artículo en el futuro, es posible que haya agregado más artículos a este catálogo. De ninguna manera debe considerar este catálogo exhaustivo.

Resumen

***Un isomorfismo es un mapeo para el cual también existe un mapeo inverso. Es una forma de describir la equivalencia.***

En la programación, a menudo tiene la opción de implementar una característica en particular de más de una manera. Estas alternativas pueden ser equivalentes, en cuyo caso son isomorfas. Esa es una de las razones por las que muchas bases de código vienen con una guía de estilo.

Comprender cómo el código es isomorfo a otro código nos permite reducir muchas alternativas a una representación canónica. Esto hace que el análisis sea más fácil, porque podemos reducir nuestro análisis a la forma canónica y generalizar a partir de ahí.

**Gap Semántico**

El intervalo semántico (del inglés semantic gap] ) es la diversidad de significado de dos descripciones de la misma cosa, debido al uso de lenguajes de expresividad distintos.  
 Esa definición no es unitaria, por lo cual el intervalo semántico también puede ser la diferencia entre una cosa real y su modelo de Software..