**Frameworks**

Un framework se puede ver como el resultado de refactorizar una aplicación, no solo para mejorar su calidad interna sino para abstraer lo que es común todas una familia de aplicaciones similares. En ese proceso de refactoring se identifica lo que se mantendrá constante y será, a partir de ese momento, responsabilidad de los desarrolladores del framework y lo que podrá variar de aplicación a aplicación y será responsabilidad de los desarrolladores de aplicaciones que usen el framework.

 Para separar la implementación de la parte constante  (a lo que llamamos **frozenspot)** y de las partes que varían (a las que llamamos **hotspots)**, se introducen en el framework puntos de extensión. Estos tienen la forma de plantillas que ofrecen ganchos para que, mediante herencia o composición, los programadores de aplicaciones definan el comportamiento variable. Algunos patrones de diseño (particularmente el template method y el strategy) son frecuentemente utilizados para introducir esos puntos de extensión.

 La construcción y uso de frameworks es una estrategia para modularizar y reutilizar, que mejora la calidad del software y aumenta la productividad de quienes lo hacen. Un framework, visto como módulo de software, no solo encapsula operaciones y estructuras de datos reusables sino que las combina para resolver, con demostrada eficacia, los requerimientos comunes a una familia de aplicaciones. En pocas palabras, encapsula el diseño y la implementación de la parte más compleja de dichas aplicaciones y permite su reutilización.

Un framework es en sí mismo un módulo de software reutilizable. Muchas aplicaciones se construirán sobre él. El framework será independiente de las aplicaciones. Sin embargo, las aplicaciones dependerán de que la interface (conjunto de puntos de extensión) y comportamiento del framework se mantenga estable. Esto es importante dado que, a lo largo de su vida, el framework será foco de contínuo refactoring y de mejoras para eliminar bugs y hacerlo más efectivo.

A continuación de ofrecen un conjunto de artículos científicos para profundizar mas en el tema frameworks. Acompañan los artículos un conjunto de preguntas guía para enfocar la lectura.

# Guia de lectura

A continuación se ofrece una guía para la lectura de una selección de artículos sobre Frameworks (y temas fuertemente relacionados). La guía está organizada por tema.

## Reúso de software

El capítulo 16 de la 9na edición del libro Ingeniería de Software de Ian Sommervile (Sommerville 2010) trata específicamente el tema reúso. Habla especialmente de Frameworks y de líneas de producción de software (este tema no lo tratamos en la asignatura). El autor publicó en su sitio una presentación de ese capítulo que puede servir como ayuda-memoria. Prestar atención a las transparencias en las que habla de los problemas y los beneficios (6 a 9) y a las que hablan sobre las distinas alternativas de reuso disponibles (12 y 13). De esas, solo nos hemos enfocado en patterns y frameworks.

## Frameworks: que son, como hacerlos, como usarlos, como aprenderlos

El libro Building Application Frameworks (Fayad  1999) es uno de los mas completos sobre frameworks. Está disponible en biblioteca. El capitulo 1 (que podrán descargar desde la sección de ficheros adjuntos) define frameworks y discute ventajas y desventajas. También discute brevemente que implica hacerlos, usarlos, aprenderlos (estos temas se tratan mas en profundidad en el resto del libro).

## Reúso Blackbox vs Whitebox - Componentes vs. Objetos

Si bien el foco del articulo de C. Szyperski  (Szyperski 1999) no es Frameworks, discute la diferencia entre reúso whitebox y reúso blackbox (intrepretación que es valida cuando hablamos de frameworks y hostpots). Adicionalmente, hace un análisis interesante de la diferencia entre Objetos y Componentes. En el marco de la asignaura no hablamos de componentes ni de Programación Orientada a Componentes. Es interesante, de todas maneras, conocer la discusión porque sirve de base a otras estrategias de reuso sistemático como es el caso de composición de servicios (Service Oriented Programming), tema que se ve en la asignatura optativa "Patrones de Arquitecturas de Software de Aplicaciones Enterprise". En el pdf del articulo que podrá encontrar en la sección de ficheros adjuntos se han resaltado con color algunos pasajes cuya lectura es recomendable.

## Frameworks - Inversión de control - Whitebox & Blackbox - Relación con Patterns - Fuerzas y debilidades

El articulo "Object Oriented Application Frameworks" (Fayad 1997) discute los beneficions de la creación y uso de frameworks. Define inversión de control. Clasifica los frameworks de acuerdo a su alcance (infraestructura, integración, y empresariales). Da una completa definición de blackbox y whitebox y discute la relación de esto con patrones como el template method, y el strategy. En el pdf se han resaltado con color algunos pasajes cuya lectura es recomendable. Hace una discusión extensa de las fortalezas y debilidades de los frameworks.

## Desarrollo de frameworks - Hotspots - White vs Black - Patterns -Herramientas

En su artículo "Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks" (Roberts 1996), los autores describen el proceso de desarrollo de frameworks. Hablan de la importancia de conocer bien el dominio antes de comenzar; hablan de blackbox y whitebox, hablan de hotspots, y hablan de herramientas para asistir a los programadores en el uso/instanciación del framework. Discute que patrones de diseño pueden ser de utilidad en cada paso del desarrollo del framework. Está escrito en forma de patrones.

## Hotspots - Template/Hook - Patones

El objetivo del articulo de Pree (Pree 1995) es presentar a los hotspots como la característica principal en la descripción/especificación de un framework. Los hotspots guian todo el proceso de desarrollo del mismo y actuan como documentación de uso. Habla de templates y hooks pero no se restringe a la interpretación que se hace en el template method sino que también habla de templates/hooks en la composición. Discute que impacto tiene esta diferencia en el comportamiento runtime de la aplicación. Discute también la relación entre patrones y hotspots. Introduce el concepto de Tarjeta de Hotpot para describir un hotspot (ya sea como requerimiento para quien desarrolla el framework o como documentación para el usuario). Define un proceso a en el cual, la definición de los hotspots guia el desarrollo del framework. En la asignatura no alcanzamos a hablar de este proceso y de las tarjetas.

## Reúso - Librerías y frameworks - Blackbox y Whitebox - Toolkits

En su articulo "Designing reusable classes" (Johnson 1988), Johnson y Foote discuten como los diseños orientados a objetos fomentan el reúso. Primero habla de abstracción, herencia y polimorfirmo. Luego habla sobre frameworks y  librerías de clases. Da una definición de inversión de control. Explica cuales son las dificultades con el uso de whitebox frameworks. También discute como, con el tiempo, un framework evoluciona desde whitebox a blackbox. En algunas secciones da a entender que cualquier buen conjunto de clases puede verse como un framework (sin hacer referencia a inversión de control). Trabajos que se escribieron mas tarde, introdujeron inversión de control como una característica fundamental para diferenciar framework de libreria o jerarquia de clases. Noten también en la sección 6, que muchas de las recomendaciónes de diseño que dan se relacionan a los malos olores y refactorings que serían publicados mas adelante. Si bien es un articulo viejo y repite temas que ya sabemos (o deberíamos saber) no está demás repasarlos y reflexionar sobre ellos.

# Ficheros adjuntos

* [Ian Sommerville. Software Engineering. Addison-Wesley; 9 edition (March 13, 2010). Presentación del capítulo 16. El libro completo lo encontrará en biblioteca o si no, comuniquese son la cátedra. (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/sommervile04_reuse_ppt.pdf)
* [Szyperski, Clemens. Components vs. objects vs. component objects. Proceedings of OOP. Vol. 1999. 1999. (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/Szyperski99.pdf)
* [Fayad, Mohamed, and Douglas C. Schmidt. Object-oriented application frameworks. Communications of the ACM 40.10 (1997): 32-38 (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/fayad97.pdf)
* [M. E. Fayad, D. C. Schmidt, and R. E. Johnson, Building Application Frameworks: Object-oriented Foundations of Framework Design. New York, NY, USA: John Wiley & Sons, Inc., 1999. Capitulo 1 - Introducción (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/building-applicaction-frameworks-chapter-1.pdf)
* [Don Roberts and Ralph Johnson. Evolving frameworks: A pattern language for developing object-oriented frameworks. Pattern languages of program design 3 (1996): 471-486. (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/roberts96.pdf)
* [W. Pree, Hot-spot-driven framework development, in Summer School on Reusable Architectures in Object-Oriented software Development, 1995, pp. 123–127 (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/pree95.pdf)
* [Johnson, Ralph E., and Brian Foote. Designing reusable classes. Journal of object-oriented programming 1.2 (1988): 22-35 (Ventana nueva)](https://catedras.linti.unlp.edu.ar/pluginfile.php/85073/mod_imscp/content/1/johnson88.pdf)

**Preguntas de autoevaluación**

Las siguientes es una lista de preguntas que debería poder contestar luego de asistir a clase, leer los artículos mencionados, y hacer la práctica.

1. Cuando hablamos de reuso en el marco del desarrollo de software, ¿a qué nos referimos?
2. ¿Cuáles son las dificultades que encontramos cuando intentamos reusar código?
3. Cuando aplicamos patrones de diseño, ¿qué estamos reusando? ¿De qué manera los patrones de diseño atacan las dificultades del reúso?
4. Cuando utilizamos librerías de clases, ¿qué estamos reusando? ¿a qué casos apuntan? ¿De qué manera atacan las librerías de clases las dificultades de reúso?
5. Cuando utilizamos frameworks orientados a objetos, ¿qué estamos reusando? ¿a qué casos apuntan? ¿De qué manera atacan los frameworks las dificultades de reúso?
6. Por lo general, ¿es posible y lógico reusar, de forma aislada, solo algunas clases de un framework? ¿es posible y lógico utilizar solo algunas clases de una librería de clases? Justifique.
7. De acuerdo a el tipo de problema que atacan, ¿como se clasifican los frameworks?
8. ¿A que nos referimos con los términos hot spot y frozen spot?
9. ¿A qué nos referimos con "instanciar un framework"? ¿Qué sería "instanciar un hostpot"?
10. ¿En qué se diferencia la visión que tienen de un framework: a) los desarrolladores del framework, b) quienes usan/instancian el framework?
11. ¿En qué se diferencia un framework blackbox de uno whitebox? Un framework, ¿cae exclusivamente en una de esas categorías?
12. Desde la perspectiva de quien "hace" el framework, ¿qué implica optar por whitebox, o blackbox? Piense por ejemplo en las facilidades y las dificultades que encuentra en cada caso.
13. Desde la perspectiva de quien "usa" el framework, ¿qué implica que el mismo sea whitebox, o blackbox? Piense por ejemplo en las facilidades y las dificultades que encuentra en cada caso.
14. La cantidad de "puntos de variabilidad" de un framework (es decir, sus hotspots) depende de cuan complejo es el dominio que ataca (es decir, de cuantas cosas podrían cambiar de aplicacion a aplicación). Por lo general, un framework tiene más hotspots que los que el usuario promedio requiere (piense que quien hizo el framework tuvo en cuenta las expectativas de muchos de ellos). ¿Siempre es necesario instanciar (configurar, subclasificar) todos los hotspots o hay algo que el diseñador del framework puede hacer para reducir el trabajo de instanciación del mismo en los casos más simples?
15. ¿Que patrones de diseño, de los que hemos visto en clase, se encuentran comúnmente en los hotspots de un framework? ó ¿Qué patrones de diseño, de los vistos en clase, son generadores de hotspots?
16. Los conceptos de template y hook (plantilla y ganchos) los encontramos en el template method, en el marco de una relación entre una clase y sus subclases. También se lo puede encontrar en una relación de composición. Ambas situaciones son comunes en los hotspots de un framework. ¿De que manera afecta una y la otra alternativa al comportamiento runtime de las aplicaciones que instancia el framework/hostpot?
17. (esto no es un pregunta, es una confirmación) Comprender que un hotspots es en principio un concepto abstracto que describe un acuerdo entre el desarrollador del framework y sus usuarios (esto es claro cuando uno piensa el cosas como los hotspot cards). Puede variar en granularidad. En el código pueden terminar siendo: a) sólo un método; b) clases a subclasificar, implementando un conjunto de métodos en cada una; c) objetos a configurar/componer
18. Le proponen hacer hacer una aplicación en un dominio que es nuevo para usted. Le aclaran que a esa le seguirán muchas parecidas. Usted piensa automáticamente en reúso y en cuanto tiempo y dinero ganará a futuro si puede reusar. Enfoca toda su fuerza en, como primer paso, construir un framework. ¿Tiene sentido? Si cree que si, ¿cómo lo haría? Si cree que no, cuáles cree que son los pasos para eventualmente poder reusar.
19. ¿Cómo cree usted que se relacionan los temas: patrones de diseño, refactoring (y refactoring to patterns), frameworks, y unit testing?