Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Segarra Martínez, Miguel J.; de Antonio Bermejo, Angel; Clavijo Blazquez, Jose A.; Sanz Bravo, Ricardo
Influencia de la Arquitectura en los Sistemas de Control Inteligente
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 4, núm. 10, verano, 2000, pp. 76-81
Asociación Española para la Inteligencia Artificial
Valencia, España

Influencia de la Arquitectura en los Sistemas de Control Inteligente

Miguel J. Segarra Martínez[†], Angel de Antonio Bermejo[*], Jose A. Clavijo Blazquez[*], Ricardo Sanz Bravo[†]

[†]Universidad Politécnica de Madrid

[*]SCILabs Ingenieros

{msegarra, aantonio, jac, sanz}@disam.upm.es

Resumen

Los sistemas de control inteligente son básicamente sistemas software. Esto hace que tengan un ciclo de vida propio compuesto por una serie de fases o etapas: especificación, desarrollo, validación operación, evolución y obsolescencia. La manera en que cada una de estas fases se desarrollará a lo largo de la vida del sistema depende exclusivamente de las decisiones que se tomaron cuando éste todavía no existía. Estas decisiones críticas, adoptadas en las primeras etapas del diseño, son las que dan carácter a muchas de las propiedades importantes de las que goza el sistema de control: rendimiento, eficiencia, atributos de calidad, capacidad de modificación, capacidad de evolución, facilidad de mantenimiento, integración con su entorno, reusabilidad, funcionalidad, dependibilidad, etc. En este artículo se trata de mostrar de qué forma la arquitectura global elegida afecta al sistema de control durante el tiempo que éste permanece en servicio.

Palabras Clave: Ingeniería del software, arquitectura software, estilos arquitectónicos, línea de producto, estructura, diseño, control inteligente, control de procesos industriales, sistemas distribuidos de control.

1-INTRODUCCIÓN

Los sistemas inteligentes de control poseen características que obligan a que su desarrollo se base en la experiencia previa del equipo encargado del proyecto. La más importante de esas características es habitualmente la carencia de una especificación clara de los requisitos o requerimientos que el sistema a diseñar debe poseer. Esto ocurre bien por la falta de una comprensión clara del problema por parte de los diseñadores o bien por una definición pobre del mismo por parte

del cliente. Naturalmente, no todas las personas involucradas en el desarrollo disponen de la misma experiencia. Exitos y fracasos anteriores influyen en las decisiones futuras. A su vez, no parece aceptable que una disciplina que se tilda de "ingeniería" utilice un método tan rudimentario de proceder como es la mera utilización de la experiencia individual adquirida sin disponer de mecanismos de formalización de la misma. Como veremos a continuación, el sistema de control se ve afectado por factores que deben ser tratados dentro de un marco de trabajo adecuado si se desean conseguir los resultados deseados.

1.1-LOS ACTORES

En la construcción de un sistema de control aparecen innumerables actores cuyos efectos se encuentran acoplados y que son de difícil manejo. A los problemas puramente técnicos, cabe añadir la complejidad de la organización global que se enfrenta al desarrollo. Se pueden distinguir varios actores que afectan a la evolución del sistema entre los que destacan los siguientes:

1. *La organización desarrolladora:* Su principal interés es realizar los trabajos dentro del presupuesto y planificación calculados. Para ello debe establecer los procesos internos necesarios que optimicen de una manera continua todos los recursos a su alcance. Necesita básicamente poder financiar sus proyectos a largo plazo que permiten a su vez generar sistemas en los que se reutiliza código, componentes o incluso sistemas completos.

2. *La organización cliente:* Habitualmente los trabajos encargados por la organización cliente responden a líneas directrices o a estrategias (reducción de costes, aumento de la producción o calidad, planes a más largo plazo, etc.) que la organización se plantea para competir dentro de su nicho de mercado. Necesitan un sistema de sencillo mantenimiento, modificable, con un ciclo de vida largo[1], etc. y que además responda perfectamente a sus especificaciones de diseño.

3. *Los usuarios finales:* Los usuarios finales necesitan un sistema fácilmente configurable, cómodo de utilizar y que, en definitiva, simplifique sus tareas diarias. Por supuesto, también desean un sistema robusto, fiable, seguro, íntegro, etc.

4. *El sistema físico:* Los sistemas en la industria presentan peculiaridades que no aparecen en otros tipos de instalaciones. Puede ser necesaria una instalación de los sistemas en "caliente", sin parar la planta o puede ser necesario diseñar sistemas redundantes que entren en funcionamiento al fallar el principal. Todo esto además se ve complicado por la complejidad de trabajar en un entorno heterogéneo con múltiples plataformas y sistemas operativos diferentes.

Existen otros actores que intervienen en la construcción del sistema, puede existir una organización intermediaria, encargada de lanzar el producto al mercado, que desee un sistema de bajo coste, finalizado a tiempo y con la calidad necesaria. También son de gran importancia las cualidades personales de los desarrolladores del sistema, etc. Tampoco se tratarán en este ámbito situaciones especiales como aquellas en que la organización desarrolladora y la cliente son la misma (casos en los que la venta del resultado final de los trabajos se realiza a miembros de la propia organización).

Es fácil comprobar que las diferentes partes interesadas en el sistema de control a menudo tienen intereses contrapuestos. Por ejemplo, parece difícil conseguir un sistema tolerante a fallos con redundancia activa o estática (interesante para la organización cliente) a un coste moderado (una pesadilla para la organización desarrolladora).

A la gran dificultad que conlleva tomar las decisiones acertadas dentro de este marco de intereses dispares y a la vez variantes en el tiempo, se une la complicación de la elección de las herramientas adecuadas para el correcto avance y gestión de los trabajos.

Existe un número inmenso de alternativas con las que atacar un problema determinado. Partiendo de modelos de organización desarrolladora (CMM, SPICE, PSP, IDEAL, SEMA), mejora de sus procesos y gestión de riesgos, modelos de gestión del cambio tecnológico en la empresa cliente y/o desarrolladora (IDEAL), pasando por las prácticas de ingeniería (análisis de requisitos, especificación del sistema, diseño del software, verificación, validación, etc.), lenguajes de modelado de sistemas (UML), sistemas de estimación del coste de proyectos software (COCOMO, SLIM) y terminando finalmente con las técnicas y herramientas de más bajo nivel para la construcción del sistema se puede comprender la cantidad de esfuerzo que la simple elección de las herramientas puede suponer a la hora de encarar un proyecto de control.

[1] No es extraño encontrar sistemas industriales de control todavía en servicio y con una antigüedad de 15 años o más.

Por dar un ejemplo, suponga el lector un proyecto que le resulte familiar y piense que alternativa elegiría para estimar su coste:

1. El modelo COCOMO propuesto por Boehm [3] basado en dos ecuaciones de difícil estimación al comienzo del proyecto pero de un funcionamiento claro.

2. El modelo SLIM [6] en el que el funcionamiento interno del modelo no es tan evidente como en COCOMO.

En realidad, para llegar a tomar una decisión es necesario realizar un análisis sólo al alcance de los expertos. Esta dificultad hace que los diseñadores se enfrenten siempre de la misma forma a problemas diferentes, lo que antes o después acaba terminando en proyectos cancelados tras una gran inversión global de todas las partes.

La concepción del sistema de control desde el punto de vista de su arquitectura (líneas de producto, desarrollo basado en COTS, desarrollo basado en arquitectura, procesos software) permite construir sistemas en los que se llega a un compromiso entre los intereses de todos los actores existentes. La razón de esta afirmación no es otra que la arquitectura del sistema basa su germinación en los requisitos de todas las partes involucradas. Por tanto, la arquitectura viene influenciada por las partes en conflicto y a su vez la arquitectura influencia a las partes.

Evidentemente, la generación de una arquitectura es un proceso costoso en tiempo y dinero. Afortunadamente, existen mecanismos que permiten la reutilización y/o particularización de instancias genéricas de arquitecturas.

2.-ARQUITECTURA DEL SISTEMA

Se podría pensar por lo dicho hasta ahora que la arquitectura del sistema debería considerar los elementos que afectan a los intereses de los actores. Si bien ese conocimiento no puede ser obviado dentro del contexto del sistema, porque afecta a su desarrollo, no es completamente cierto que sea parte de la arquitectura del sistema. El sistema de control tiene naturaleza propia y, dada una métrica específica, responde a una arquitectura que debe ser descubierta[2].

Una vez descubierta la arquitectura del sistema es esencial desglosar esa arquitectura de manera que encaje dentro de las necesidades y capacidades de todos los miembros involucrados.

Asumiendo las puntualizaciones anteriores, una definición comúnmente aceptada de "Arquitectura Software" es la siguiente:

"La arquitectura software de un programa o sistema de cómputo es la estructura o estructuras del sistema, que comprende a sus componentes software, las propiedades de esos componentes visibles de forma externa, y las relaciones entre ellos" [1]

Desde nuestro punto de vista, esta definición debe tomarse en un sentido amplio. La "*estructura o estructuras* del sistema" son muy variadas y afectan no sólo al software sino a los procesos que ligan a todos los miembros involucrados en el desarrollo (organización desarrolladora, cliente, intermediarios, etc.).

No todas las estructuras tienen el mismo peso. La más importante es aquella que representa al sistema de control aislado de los actores que intervienen en su construcción (que muchos entienden como arquitectura del sistema) y que sintetiza la esencia del sistema final. Las demás estructuras se originan a partir de ésta y su único objetivo es realizar una transformación de la estructura del sistema a la estructura real de las organizaciones de los miembros del desarrollo.

3.-EL ARQUITECTO

Sin duda alguna la figura del arquitecto es de una importancia singular cuando el desarrollo parte del desgrane de la arquitectura del sistema.

Sus tareas son de gran complejidad y, aparentemente en gran medida, no pueden ser aprendidas en un centro de educación. Necesitan un profundo conocimiento técnico de su dominio de conocimiento, lo que contribuye a consolidar la capacidad de liderazgo que debe poseer, además de una gran habilidad social que permita negociar con todas las partes. Para realizar esta tarea debe conocer a la mayor rapidez cuáles son las necesidades de todas las partes con el fin de poder negociar contrapartidas desde un punto de vista ventajoso.

Adicionalmente, debe conocer todas las herramientas necesarias en cada momento de la ejecución de los trabajos y la mejor forma de seleccionar entre ellas. Este conocimiento no se transmite fácilmente y, desde nuestra experiencia, se adquiere con el tiempo. Un buen arquitecto, a falta de métodos formales, debe desarrollar sus propios métodos de selección de herramientas.

[2] Una vez dada una métrica, la cuestión sobre la existencia de una arquitectura óptima y de si esa arquitectura es descubierta o es inventada es un gran tema de discusión sobre el que existen variadas opiniones.

Desgraciadamente, existe una tendencia clara a hacer de un arquitecto un simple gestor ó un negociador que aparece de forma puntual en algunos estadíos del proyecto. De esta forma el arquitecto pierde toda la perspectiva del proyecto y es incapaz de resolver los verdaderos problemas que indudablemente terminan apareciendo. Además de esto, se gasta su experiencia de una forma inútil y a la vez su cualificación profesional se deprecia como consecuencia de la pérdida de la nueva experiencia que el proyecto podría haberle proporcionado.

Puede parecer una definición simple pero es clara, *"un arquitecto no es un gestor, es un técnico con don de gentes"*.

4.-CONSTRUCCION DE SISTEMAS DE CONTROL INTELIGENTE

Una vez conocidos algunos de los actores que intervienen en la construcción de sistemas software de control cabría preguntarse cuáles son las tareas que se deben realizar para llevar a cabo un desarrollo con garantías de éxito.

Si comparamos diferentes sistemas de control inteligente veremos que son bastante heterogéneos. En general, no están constituidos por componentes de características similares. A pesar de ello sí es posible encontrar similitudes que permitan establecer pautas de comportamiento.

Se entiende que inicialmente deben aparecer en el sistema comportamientos que lo clasifiquen dentro de lo que se conoce como *control inteligente.* Estamos pensando en proporcionar en alguna medida funcionalidad al sistema mediante técnicas derivadas de la Inteligencia Artificial: lógica borrosa, algoritmos genéticos, redes neuronales, sistemas expertos, razonamiento basado en modelos, etc. Sin embargo, decidir si las técnicas de control inteligente son de aplicación a un cierto sistema requiere cierto estudio del mismo como se indica a continuación.

En un primer acercamiento a la arquitectura del sistema deben clasificarse qué problemas son los que se quieren resolver. La colaboración con el personal de planta es vital en este punto. Son ellos los que verdaderamente conocen el proceso y debe exigirse en este punto una colaboración total de las personas que más conocen los problemas a resolver. Frecuentemente estas personas tienen gran valor para la compañía y tienden a estar sobrecargadas de trabajo, haciéndose su participación en la especificación de los problemas muy difícil. Sin embargo, una mala identificación de las necesidades del sistema dará lugar, a lo largo del tiempo, a una pérdida económica mucho mayor que el valor de las horas que un experto o expertos dediquen a la identificación del problema de control.

Una vez identificados los problemas a resolver, es recomendable evaluar la conveniencia del uso de técnicas de Inteligencia Artificial para la resolución de los mismos. Hemos visto situaciones en las que, por ejemplo, problemas de control en sistemas que se comportaban de una manera lineal se resolvían mediante complejos reguladores *fuzzy*, dándose además la circunstancia de encontrarse la parte cliente verdaderamente entusiasmada con la solución adoptada. El motivo no era otro más que la imagen tecnológicamente avanzada que la introducción de un controlador de este tipo proporciona al cliente.

Debe comprenderse que desde el punto de vista de la ingeniería esta solución, sin dejar de ser válida, no es la más elegante, eficaz o siquiera económica, aunque sí es posible que fuera rentable desde el punto de vista de la imagen de empresa.

Ante la evidencia de situaciones de control en las que las técnicas clásicas no fueran eficaces para cumplir los objetivos propuestos es necesario realizar una *selección de tecnologías* [9] adecuadas a cada problema. Esta técnica clasifica de una forma tabular los siguientes parámetros: *dominio, tarea* y *tecnología*. De esta manera, y de una forma extraordinariamente sencilla, es posible decidir la tecnología y pasos a seguir para resolver un problema una vez que éste ha sido identificado.

En casos simples mediante la selección realizada anteriormente es posible llegar a una solución satisfactoria. Desafortunadamente, esto solamente ocurre en contadas ocasiones.

En situaciones más complejas se requiere la colaboración de diferentes técnicas de control, aumentando en gran medida la complejidad del sistema. En algunas ocasiones y ante problemas pequeños, el sistema puede ser construido desde cero. Cuando el sistema crece de tamaño no es posible considerar esta aproximación. El sistema debe ser entonces compuesto a partir de componentes preconstruidos que se combinan para obtener el efecto deseado.

Actualmente se pueden componer sistemas basados en *COTS*[3] con relativa sencillez. Además, presentan la ventaja de que el coste del desarrollo o compra del componente se reparte entre los diferentes sistemas implantados. La contrapartida del uso de este tipo de componentes es la falta de control sobre su calidad y las posibles faltas en su funcionamiento. Este problema se conoce como el síndrome del *"no hecho aquí"*. El coste de desarrollar componentes

[3] Commercial-off-the-shelf components

propios se ve ampliamente compensado mediante su reutilización continua. Además, en el caso de componentes de terceros, se evita el trabajo extra de evaluar si el componente realmente proporciona la funcionalidad requerida y la seguridad con que esas funciones se ejecutan.

Como recomendación básica en el desarrollo de sistemas críticos basados en componentes es imprescindible preguntarse *"¿cumple el componente con la funcionalidad necesaria?"* y *"¿qué ocurre si un componente provoca una falta?"*.

Una vez seleccionados los componentes que resuelven problemas particulares es necesario elevar un orden de magnitud el nivel de abstracción con que se observa el problema. El objetivo es tratar de observar las relaciones y/o acoplamientos que pudieran existir entre diferentes partes del sistema. Estas relaciones son de muy difícil extracción y deben ser sistematizadas en forma de *patrones* [10]. De esta forma el conocimiento adquirido no se perderá y será conservado de una forma fácilmente transmisible a otros arquitectos. Uno de los fines de la obtención de la arquitectura del sistema es la transmisión del conocimiento que se posee de éste entre todas las partes involucradas en el mismo. Las tablas de selección de tecnologías y los patrones no son más que medios de transporte de ese conocimiento. La obtención de los *patrones* que componen el sistema permite realizar variaciones sobre las políticas de funcionamiento de los componentes individuales de forma que se consiga su colaboración y no su funcionamiento aislado.

Finalmente, se deben comprender las relaciones existentes entre los distintos subsistemas existentes. Realizadas estas tareas, se conocen las propiedades externas de los componentes del sistema, se conoce la estructura de los mismos y las relaciones entre ellos. Es decir, se ha encontrado la arquitectura intrínseca del sistema y que más fielmente representa su naturaleza.

La construcción del sistema ni siquiera ha comenzado puesto que sólo se conoce su naturaleza, pero no cómo se refleja en la organización que pretende desarrollarlo. De aquí nace la afirmación de que la arquitectura del sistema influencia a las partes y a su vez las partes influencian a la arquitectura.

La *focalización progresiva de dominios*[2] puede arrojar alguna luz sobre la forma en la que debe ser transformado el dominio de la arquitectura del sistema, como sistema de control, sobre el dominio de la organización encargada de llevarlo acabo.

Aunque la focalización progresiva de dominios propone una metodología "top-down" para el desarrollo de sistemas autónomos, no desestima la manera de enfrentarse a los problemas propuesta anteriormente. La idea fundamental es que el conocimiento adquirido mediante el método "bottom-up" propuesto debe dar lugar a comportamientos inteligentes independientes de la tarea a desarrollar. Además se reconoce también como fundamental la idea de que la separación entre aplicación y dominio no se encuentra determinada de una manera nítida, existe una frontera borrosa.

La técnica se centra en la focalización del problema en dominios de anchura decreciente hasta que las aplicaciones finales son construidas. En cualquier punto de este proceso es posible generar componentes del sistema final. Al disminuir junto con el dominio la complejidad de la funcionalidad específica objeto de estudio resulta más sencillo realizar una transformación entre la estructura del sistema de control y la de la organización que la desarrolla.

Este tipo de desarrollos sólo es posible si se realiza un esfuerzo investigador a largo plazo, donde una colección ampliable de componentes que representan comportamientos inteligentes se integran para representar diferentes patrones de diseño típicos de los sistemas de control. A su vez el medio de integración de los patrones es un middleware de comunicaciones con características propias de la industria a la que va destinado.

Si el middleware de integración proporciona la funcionalidad adecuada se puede realizar el diseño y construcción del sistema como si fuera monolítico. Esto permite crear prototipos rápidamente con los que explorar las características reales del sistema sobre el que se actúa y realizar una primera aproximación a los dominios del sistema y las diferentes estructuras de su arquitectura. En etapas posteriores se puede realizar una distribución del sistema sobre sus plataformas finales de una manera sencilla, aprovechando la funcionalidad proporcionada por el marco de integración. La capacidad de diseñar un sistema como monolítico y después ser distribuido usando la capacidad del middleware se conoce con el nombre de *Diseño Distribuido Tardío*[4].

5.-CONCLUSIONES

El objetivo final como ingenieros no debe ser otro que realizar el control del sistema de la mejor forma posible. Los procesos aquí expuestos no explican como realizar dicho control pero sí explican como no hacer software.

Los sistemas de control inteligente son intensivos en cuanto al uso de software se refiere, y éste es muy costoso de realizar. La aproximación de objetivos entre las diferentes estructuras relacionadas en la

construcción de un sistema (las estructuras de su arquitectura) puede dar lugar a un proceso de desarrollo más eficiente en el uso de los recursos para conseguir un sistema determinado.

Existen diferentes técnicas que permiten rebajar los costes de tiempo, esfuerzo y dinero en el desarrollo de un nuevo sistema. La arquitectura del sistema influencia a las partes involucradas creando nuevos esquemas organizativos y nuevos procesos para atacar el problema representado por el sistema con un riesgo mínimo. A su vez las partes deben buscar la transformación de la estructura del sistema de control que mejor se adapte a sus características.

Dentro de este entorno en el que existen múltiples interacciones se propone el siguiente marco de trabajo como un "método" útil de desarrollo:

- Identificar el problema. Debe exigirse una completa colaboración del personal de planta en este punto.
- Decidir si son de aplicación las técnicas de control inteligente derivadas de la Inteligencia Artificial.
- Realizar una selección de las tecnologías adecuadas a cada problema a resolver.
- Si existen varias técnicas y/o varios problemas identificar las interacciones entre ellas. En este caso puede ser necesario considerar un desarrollo basado en componentes dentro de un marco de integración.
- Identificar los patrones de control que aparecen en el sistema como consecuencia de la interacción de diferentes componentes. Estos patrones darán lugar a diferentes subsistemas de control.
- Identificando los acoplamientos de los diferentes subsistemas se obtiene la estructura más básica del sistema (para algunos su arquitectura).
- Realizar una transformación de la estructura obtenida del sistema de control a la las diferentes estructuras que representan la organización que pretende desarrollarlo y hallar las zonas comunes. Esas estructuras representan el problema de control visto desde las diferentes partes y bajo diferentes puntos de vista: técnico, de disponibilidad de personal, costes, plazos, etc. Las zonas comunes deben representar al sistema final en un 90-95% de su totalidad. El área común representa la arquitectura global del sistema.
- Utilizar la focalización progresiva de dominios sobre la arquitectura global del sistema para conseguir parte de la funcionalidad del sistema en cualquier nivel del proceso.
- Finalmente, se deben completar las partes del sistema que no fueron contempladas en su arquitectura global y que representan entre un 5-10% de su funcionalidad

Referencias

[1] Bass L., Clements P. and Kazman R., 'Software Architecture in Practice',SEI Series in Software Engineering, Addison Wesley 1998.

[2]Sanz R., Alarcón I., Segarra M., Clavijo J. And Antonio de A. 'Progressive Domain Focalization in Intelligent Control Systems' To appear in "Control Engineering Practice".

[3] Boehm, B.W., 'Software Engineering Economics'. Prentice-Hall: Englewood Cliffs, NJ, 1981.

[4]Antonio de A. 'Arquitectura de control inteligente: Un enfoque basado en componentes para sistemas de control' Tesis doctoral. Universidad Politécnica de Madrid 1999

[5]Ian Sommerville 'Software Engineering' Fifth Edition Addison-Wesley 1995

[6] Putman, L.H., 'A general empirical solution to the macro software sizing and estimating problem'. *IEEE Trans. on Softw. Eng*., 4(4), 345-61, 1978.

[7] Shepperd, M.J., 'Foundations of Software Measurement'. Prentice Hall: Hemel Hempstead, UK, 1995.

[8] Banker, R.D. and C.F. Kemerer, 'Scale economies in new software development', *IEEE Trans. on Softw.Eng.*, 15(10), 199-204, 1989

[9]Alarcón, I. 'Modelo para el análisis y conceptualización de sistemas inteligentes para control de procesos industriales en tiempo real' Departamento de Inteligencia Artificial. Facultad de Informática. Universidad Politécnica de Madrid. Mayo, 1995.

[10] Erich Gamma, et al. 'Design Patterns: Elements of Reusable Object-Oriented

Software’ Addison-Wesley Professional
Computing, 1994.