Octavo Taller Internacional sobre Especificación y Diseño de Software, Alemania, marzo de 1996

Una encuesta sobre los lenguajes de descripción de la arquitectura

Paul C. Clements

Instituto de Ingeniería de Software

Universidad de Carnegie mellon

Pittsburgh, PA 1521

Abstracto

Los lenguajes de descripción de arquitectura (ADL) están surgiendo como herramientas viables para representar formalmente las arquitecturas de los sistemas. Aunque crecen en número, varían ampliamente en términos de las abstracciones que admiten y las capacidades de análisis que proporcionan. Además, muchos lenguajes que no se diseñaron originalmente como ADL sirven razonablemente bien para representar y analizar arquitecturas de software. Este artículo resume una encuesta taxonómica de las AVD que está en curso. La encuesta caracteriza a las ADL en términos de (a) las clases de sistemas que soportan; (b) las propiedades inherentes de las lenguas mismas; y (c) el proceso y el apoyo tecnológico que brindan para representar, refinar, analizar y construir sistemas a partir de una arquitectura. Los resultados preliminares nos permiten sacar conclusiones sobre lo que constituye una AVD y cómo las AVD contemporáneas se diferencian entre sí.

1. Introducción

Lenguajes de descripción de arquitectura (ADL) son lenguajes formales que se pueden utilizar para representar la arquitectura de un sistema intensivo en software. A medida que la arquitectura se convierta en un tema dominante en el desarrollo y la adquisición de grandes sistemas, los métodos para especificar sin ambigüedades una arquitectura se volverán indispensables.

Por *arquitectura*, nos referimos a los componentes que componen Valorar un sistema, las especificaciones de comportamiento de esos componentes y los patrones y mecanismos de interacción entre ellos. Tenga en cuenta que un solo sistema suele estar compuesto por más de un tipo de componente: módulos, tareas, funciones, etc. Una arquitectura puede elegir el tipo de componente más apropiado o informativo para mostrar, o puede incluir múltiples vistas del mismo sistema, cada uno ilustrando diferentes componentes.

Hasta la fecha, las arquitecturas se han representado en gran medida mediante dibujos informales de círculos y líneas en los que la naturaleza de los componentes, sus propiedades, la semántica de las conexiones y el comportamiento del sistema en su conjunto están mal (si es que lo están) definidos. . Aunque estas cifras a menudo dan una imagen intuitiva de la construcción del sistema, por lo general no responden a preguntas como:

- ¿Cuáles son los componentes? ¿Son módulos que existen solo en tiempo de diseño, pero se compilan juntos antes del tiempo de ejecución? ¿Son tareas o procesos agrupados desde diferentes módulos, ensamblados en tiempo de compilación y forman unidades de tiempo de ejecución? ¿O son algo tan nebuloso como "áreas funcionales", como en los diagramas de flujo de datos, o algo completamente diferente?
- ¿Qué hacen los componentes? ¿Cómo se comportan? ¿En qué otros componentes se basan?
- ¿Qué significan las conexiones? ¿Se refieren a "envía datos a", "envía control a", "llama", "es parte de", alguna combinación de estos o algo más? ¿Cuáles son los mecanismos utilizados para cumplir con estas relaciones?
- Las ADL son el resultado de un enfoque lingüístico de la representación formal de arquitecturas y, como tales, abordan las deficiencias de las representaciones informales. Además, como se mostrará, las ADL sofisticadas permiten el análisis temprano y las pruebas de viabilidad de las decisiones de diseño.1.

Las ADL tienen sus raíces en los lenguajes de interconexión de módulos de la década de 1970. Hoy en día, las AVD se encuentran en una fase de maduración, pero existen varias. Los ejemplos actuales incluyen Rapide [12], UniCon [18], ArTek [19], Wright [1] y Meta-H [21].

Este documento describe una encuesta de ADL contemporáneas que está actualmente en progreso. Utilizando las técnicas de análisis de dominio, se elaboró un cuestionario que caracteriza a una ADL individual en términos

de los sistemas y arquitecturas que puede soportar, el análisis o el desarrollo automatizado que puede facilitar o proporcionar, y las cualidades intrínsecas de la propia ADL. El cuestionario se ha aplicado a más de una docena de AVD hasta la fecha y los datos resultantes permiten comparar y contrastar las AVD. Los datos también brindan información sobre la cuestión de cuándo un lenguaje es una ADL en comparación con algún otro tipo de lenguaje, como un lenguaje de requisitos, programación o modelado.

2. Arquitectura y ADL

Una arquitectura juega varios roles en el desarrollo del proyecto, todos ellos importantes, y todos ellos facilitados por una representación formal de la arquitectura, como con una ADL. Es más probable que se mantenga y se siga una representación de arquitectura formal que una informal, se puede consultar y tratar más fácilmente como autoritaria y se puede transferir más fácilmente a otros proyectos como un activo central. Los roles incluyen:

- Base para la comunicación: los miembros del equipo del proyecto, los gerentes y los clientes recurren a la arquitectura como base para comprender el sistema, su desarrollo y cómo funciona durante la ejecución.
- Anteproyecto: La elección de los componentes arquitectónicos se institucionaliza en la estructura del equipo de la organización en desarrollo, las asignaciones de trabajo, las unidades de gestión, el cronograma y las estructuras de desglose del trabajo, los planes de integración, los planes de prueba y los procesos de mantenimiento. Una vez que se toma, una decisión de arquitectura tiene una vida útil extremadamente larga y sobrevive incluso fuera del software que describe.
- Plan para el desarrollo de la línea de productos. Una arquitectura puede reutilizarse en otros sistemas para los que sea apropiada. Si se gestiona con cuidado, se puede producir una familia de productos completa utilizando una única arquitectura. En este caso, la importancia de una arquitectura adecuada se magnifica en todos los proyectos a los que servirá.
- Realización de las primeras decisiones de diseño: la arquitectura representa el primer mapeo de los requisitos a los componentes computacionales. La selección de componentes y conexiones, así como la asignación de funcionalidad a cada componente, es una codificación de las primeras decisiones de diseño de un proyecto. Todas las decisiones de diseño posteriores deben ser coherentes con las opciones arquitectónicas. Como tal, las decisiones arquitectónicas son las más difíciles de cambiar y tienen las consecuencias de mayor alcance.
- Primer enfoque para lograr atributos de calidad: una arquitectura puede permitir o impedir el logro de la mayoría de los atributos de calidad objetivo de un sistema. La modificabilidad, por ejemplo, depende en gran medida de la modularización del sistema, que refleja las estrategias de encapsulación. La reutilización de los componentes depende de qué tan fuertemente acoplados estén con otros componentes del sistema. El rendimiento depende en gran medida del volumen y la complejidad de la comunicación y coordinación entre componentes, especialmente si los componentes son procesos distribuidos físicamente. Por lo tanto, una arquitectura incorpora decisiones sobre prioridades y compensaciones de calidad, y representa la primera oportunidad para evaluar esas decisiones y compensaciones.

Algunas ADL brindan una oportunidad para el análisis a nivel de arquitectura, como la generación automática de simulación, el análisis de capacidad de programación y similares. Sin embargo, incluso en ausencia de capacidades de análisis automatizadas, se pueden aplicar otras estrategias de evaluación a la arquitectura [5]. Por lo tanto, estas decisiones de diseño tempranas y las compensaciones de atributos de calidad se pueden probar antes de que sean demasiado costosas para cambiar.

3. ADL y su relación con otros idiomas

¿En qué se diferencian las ADL de los lenguajes de programación, los lenguajes de requisitos, los lenguajes de modelado y similares? Dado un lenguaje para expresar propiedades o comportamientos de un sistema, ¿cuáles son los criterios para decidir si es una ADL o no? Desafortunadamente, no está claro.

En principio, las ADL se diferencian de los lenguajes de requisitos porque los últimos describen espacios de problemas, mientras que los primeros están arraigados en el espacio de soluciones. En la práctica, los requisitos a menudo se dividen en fragmentos de comportamiento para facilitar la presentación, y los lenguajes para representar esos comportamientos a veces son adecuados para representar componentes arquitectónicos, aunque ese no era el original.

---

1. La encuesta establece una distinción entre el idioma, el análisis que *pueden* realizarse sobre la información arquitectónica representable por el lenguaje, y las herramientas que realmente existen para realizar dicho análisis. Sin embargo, así como nadie usaría Ada sin un compilador de Ada, no es probable que alguien adopte una ADL para un proyecto sin adoptar también las herramientas de edición, refinamiento, análisis o creación de sistemas que vienen con él. Por lo tanto, en una encuesta diseñada para ayudar a los profesionales a elegir una AVD, vemos una AVD como el idioma más su conjunto de herramientas de apoyo, el *suma* de los cuales los profesionales utilizarán como base para la selección, el uso y la evaluación posterior a los hechos. Sin embargo, señalamos dónde el lenguaje captura información que podría estar, pero quizás aún no ha sido, explotada por una herramienta de análisis.

objetivo del idioma. Por ejemplo, Modechart [10], un lenguaje de requisitos similar a Statechart [7], exhibió capacidades analíticas más fuertes que cualquier otra ADL en nuestra encuesta debido a la presencia de un verificador de verificación de modelos. Se consideró que Modechart era una ADL porque sus componentes (máquinas de estado) *interpretado* como componentes arquitectónicos. Pero Modechart no fue diseñado para ser una ADL, por lo que es fácil producir artefactos en Modechart que, bajo ninguna interpretación semántica razonable, corresponden a una vista arquitectónica de un sistema.

En principio, las ADL difieren de los lenguajes de programación porque estos últimos vinculan todas las abstracciones arquitectónicas a soluciones puntuales específicas, mientras que las ADL suprimen o varían intencionalmente dicha vinculación. En la práctica, la arquitectura está incorporada y se puede recuperar del código, y muchos lenguajes proporcionan vistas del sistema a nivel de arquitectura. Por ejemplo, Ada ofrece la capacidad de ver un sistema solo en términos de las especificaciones de su paquete, que son las interfaces de los componentes. Sin embargo, Ada ofrece poca o ninguna capacidad analítica a nivel de arquitectura, ni proporciona información a nivel de arquitectura sobre cómo se "conectan" los componentes entre sí.

En principio, las ADL se diferencian de los lenguajes de modelado porque estos últimos se preocupan más por los comportamientos del todo que de las partes, mientras que las ADL se concentran en la representación de componentes. En la práctica, muchos lenguajes de modelado permiten la representación de componentes cooperantes y pueden representar arquitecturas razonablemente bien.

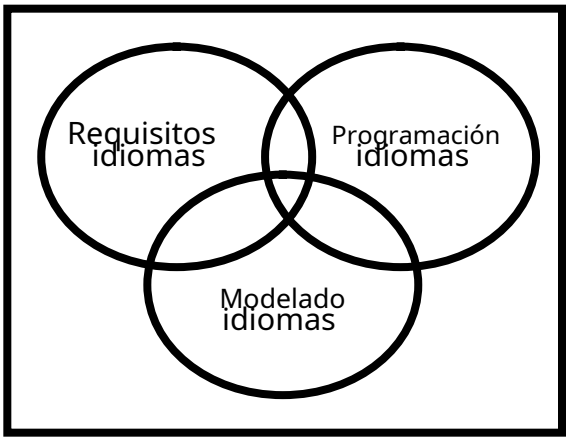

Figura 1: Lenguajes de requisitos, programación lenguajes y lenguajes de modelado han pectos en común con las AVD.

Dos investigadores líderes en AVD ofrecen sus deseos para las AVD. Shaw enumera las siguientes propiedades importantes erties que deben exhibir las ADL [18]:

- capacidad para representar componentes (primitivos o compuestos) junto con afirmaciones de propiedad, interfaces e implementaciones;
- capacidad para representar conectores, junto con protocolos, afirmaciones de propiedad e implementaciones
- abstracción y encapsulación
- tipos y verificación de tipos
- capacidad para acomodar herramientas de análisis abiertamente

Luckham enumera los siguientes requisitos para una ADL [13]:

- abstracción de componentes
- abstracción de la comunicación
- integridad de la comunicación (limitando la comunicación a aquellos componentes conectados entre sí arquitectónicamente)
- capacidad para modelar arquitecturas dinámicas
- capacidad de razonar sobre la causalidad y el tiempo
- soporte de refinamiento jerárquico
- relatividad, el mapeo de comportamientos a arquitecturas (posiblemente diferentes), como primer paso para verificar la conformidad.

Estas listas ilustran los diferentes puntos de vista sobre lo que constituye una AVD. No existe una línea clara entre las ADL y las no ADL. Sin embargo, los idiomas pueden discriminarse entre sí de acuerdo con la cantidad de información arquitectónica que representan, y nuestra encuesta ha intentado capturar esto. Los lenguajes que nacieron como ADL muestran una clara ventaja en esta área sobre los lenguajes construidos para algún otro propósito y luego cooptados para representar arquitecturas. En la Sección 6, volveremos a examinar este tema a la luz de los resultados de la encuesta.

4. La encuesta ADL

Esta sección describe el propósito, la forma, el contenido y la metodología de la encuesta ADL.

4.1: Propósito

Nuestra encuesta de ADL tenía como objetivo proporcionar información a tres comunidades:

- arquitectos, que deben elegir una ADL. Nuestra encuesta tiene como objetivo resaltar las capacidades y cualidades de las ADL actualmente disponibles. No es una tarjeta de puntuación ni siquiera una evaluación; Las AVD no son mejores ni peores que

mutuamente. Más bien, presentan diferentes capacidades y cualidades, y la mejor opción es el uso específico.

• patrocinadores de tecnología, que financian el desarrollo de ADL y herramientas de ADL. Nuestra encuesta está destinada a permitirles detectar y redirigir el trabajo duplicado.

• Creadores de ADL. Nuestra encuesta tiene como objetivo permitir que los creadores identifiquen áreas de capacidad que, hasta la fecha, han pasado por alto en gran medida por las ADL.

4.2: Forma

A partir de esfuerzos anteriores para encuestar las ADL [20], así como otros tipos de lenguajes de especificación [2], elaboramos un *análisis de características* de AVD. El análisis de características es una herramienta de ciertos métodos de análisis de dominio, como el método de Análisis de dominio orientado a características (FODA) [11]; procede catalogando las características del sistema visibles para el usuario de una manera estructurada. En nuestra encuesta, el dominio consistió en el conjunto de idiomas que podrían considerarse ADL; "Visible para el usuario" significa aparente para un usuario del idioma.

El análisis de características tomó la forma de un cuestionario de encuesta; el cuestionario es la manifestación de un marco de características importantes que una ADL en particular puede tener o no. Una ADL en la encuesta se caracteriza por responder una serie de preguntas sobre sus capacidades, características e historial de uso. Por lo tanto, un conjunto de cuestionarios completados proporciona una base para comparar y contrastar las AVD entre sí, y las AVD con idiomas que no se considerarían AVD.

Las características están estructuradas en las siguientes tres categorías: características orientadas al sistema, características orientadas al lenguaje y características orientadas a procesos.

Funciones orientadas al sistema

Las características orientadas al sistema están relacionadas con el sistema de aplicación derivado de la descripción de la arquitectura. Por ejemplo, es posible que ciertas ADL no puedan expresar restricciones en tiempo real sobre los componentes arquitectónicos de un sistema, mientras que otras sí. Todas las características de esta categoría son atributos de un sistema final; sin embargo, reflejan la capacidad de la ADL para expresar o describir esos atributos a nivel arquitectónico.

Las preguntas específicas de la encuesta incluyen[2]:

Aplicabilidad: ¿Qué tan adecuada es la ADL para representar un tipo particular de sistema de solicitud?

- **Estilos arquitectónicos: ¿Qué tan bien permite la ADL la descripción de estilos arquitectónicos, como los enumerados en [6]? Los estilos incluyen tubería y filtro, pizarra, etc.**
- **Clase de sistema: ¿Qué clases amplias de sistemas pueden tener sus arquitecturas representadas con la ADL? Las clases incluyen: arquitecturas dinámicas en tiempo real duro, tiempo real suave, integradas, distribuidas, sistemas de componentes importados, otros.**
- **Dominios: ¿Qué dominios de aplicación está diseñada específicamente para admitir la ADL, si corresponde, y cómo y en qué grado?**

Funciones orientadas al idioma

Las características orientadas al lenguaje son características de la propia ADL, independientemente del sistema o sistemas para los que se esté utilizando. Estos atributos incluyen el tipo de información que generalmente se encuentra en un manual de referencia de idiomas. Un ejemplo es cuán formalmente especificadas están la sintaxis y la semántica de la ADL y qué abstracciones arquitectónicas incorpora la ADL.

Las preguntas orientadas al idioma incluyen:

Calidad de la definición del lenguaje

- **Formalidad: ¿Cuán formalmente se definen la sintaxis y la semántica de la ADL?**
- **Integridad: ¿Qué tan bien se define la integridad para las descripciones de una arquitectura? ¿Cómo trata la ADL una descripción de arquitectura incompleta?**
- **Coherencia: ¿Se define la autoconsistencia para la descripción de una arquitectura? ¿Se define entre dos descripciones de arquitectura diferentes o entre una descripción de arquitectura y alguna otra representación del sistema, como una especificación de requisitos o una implementación codificada? ¿Las reglas de coherencia están integradas o definidas por el usuario?**

Alcance del lenguaje: ¿Cuánta información no relacionada con la arquitectura puede representar la ADL?

Historial de diseño: ¿Qué tan bien proporciona la ADL para registrar la información de diseño arquitectónico?

Vistas: ¿Qué tan bien soporta la ADL diferentes puntos de vista que resaltan diferentes aspectos / perspectivas de la arquitectura?

- **Lista de vistas sintácticas: ¿Qué vistas sintácticas son compatibles? Gráficos, textuales, etc.**
- **Lista de vistas semánticas: ¿Qué vistas semánticas son compatibles? Flujo de datos, flujo de control, vista de proceso, etc.**
- **Referencia cruzada de entrevistas: ¿La ADL prevé la traducción entre vistas?**

---

2. Por brevedad, se ha omitido una gran cantidad de aclaraciones, al igual que la explicación de las escalas de respuesta. Por ejemplo, la pregunta "¿Cómo trata la ADL una descripción de arquitectura incompleta?" se ejemplifica en el cuestionario real como varias preguntas detalladas que tratan de las operaciones que se pueden realizar en una descripción incompleta, si existen o prevalecen reglas de completitud integradas o definidas por el usuario, si el lenguaje presenta un comodín o un token de incompletitud, etc. Pida al lector que crea que las preguntas que suenan subjetivas presentadas en este documento están respaldadas en el cuestionario real con subpreguntas detalladas que abordan cada tema de manera mucho más objetiva.

• Contenido arquitectónico de las vistas: ¿La ADL proporciona vistas que muestran principalmente información arquitectónica?

Legibilidad

• Comentarios incrustados

• Control de presentación

Características de los usuarios previstos

• Usuarios objetivo: ¿ingeniero de dominio, ingeniero de aplicaciones, analista de sistemas, administrador de software?

• Experiencia requerida: experiencia en el dominio, experiencia en diseño de software, experiencia en lenguajes de programación.

Modificabilidad de la descripción de la arquitectura del software

• Facilidad de cambio: ¿Qué tan bien soporta la ADL la facilidad de cambio de la arquitectura y su representación?

• Escalabilidad: el grado en que la ADL puede representar sistemas grandes y / o complejos. Jerarquías ¿Referencia cruzada? ¿Capacidad de subconjunto? ¿Composición? ¿Múltiples instancias de plantillas?

Variabilidad: ¿Qué tan bien representa la ADL las variaciones en los sistemas de aplicación que se pueden derivar de una arquitectura?

Poder expresivo de la ADL.

• Se presentan poderosos primitivos

• Extensibilidad

• ¿Qué abstracciones apoya o proporciona la ADL? ¿Las abstracciones se basan en la arquitectura, el comportamiento o la implementación?

Funciones orientadas al proceso

Las características orientadas a procesos son características de un proceso relacionadas con el uso de ADL para crear, validar, analizar y refinar una descripción de arquitectura y construir un sistema de aplicación a partir de ella. Se incluyen atributos que miden o describen cómo o hasta qué punto una ADL permite la evaluación predictiva del sistema de aplicación basada en información a nivel de arquitectura. Estos atributos miden si la ADL contiene suficiente información para hacer que una arquitectura sea analizable, independientemente de si existen o no herramientas que explotan esa capacidad. Además, el cuestionario proporciona un lugar para describir las herramientas existentes.

Por ejemplo, una ADL puede permitir que se proporcione suficiente información de tiempo para respaldar el análisis de planificación. Un analizador de programación de análisis monótono de tasas (si existe para la ADL) sería un ejemplo de una herramienta que explota dicha información.

Las áreas de análisis se extraen principalmente de IEEE Std 1061, "Metodología de métricas de calidad del software" [9]. Muchos de estos atributos no se abordan en ninguna ADL existente.

Las preguntas orientadas al proceso son:

Soporte de creación de arquitectura: editor de texto, editor gráfico, herramienta de importación?

Soporte de validación de arquitectura: verificador de sintaxis, verificador de semántica, verificador de integridad, verificador de coherencia?

Compatibilidad con el refinamiento de la arquitectura: navegador, herramienta de búsqueda, herramienta de refinamiento incremental, control de versiones, comparación de arquitectura.

Soporte de análisis de arquitectura: ¿Qué soporte proporciona la ADL para analizar información a nivel de arquitectura con el fin de predecir o proyectar las cualidades del sistema final?

• Análisis de economía de tiempo y recursos: programabilidad, rendimiento, otras economías de tiempo, utilización de la memoria, ¿otras economías de recursos?

• Analizando la funcionalidad: ¿integridad, corrección, seguridad, interoperabilidad?

• Análisis de la capacidad de mantenimiento: ¿corregibilidad, capacidad de expansión, capacidad de prueba?

• Análisis de portabilidad: ¿independencia del hardware, independencia del software?

• Análisis de confiabilidad: ¿tolerancia a errores, capacidad de operación degradada, disponibilidad?

• Análisis de usabilidad: ¿comprensibilidad, facilidad de aprendizaje, operabilidad?

Creación de aplicaciones: creación de un sistema de software compilable (o ejecutable) a partir de un diseño de sistema específico.

• Composición del sistema: la composición o integración de los cuerpos de los componentes, con el fin de producir un sistema de software compilable o ejecutable para: destino de un solo procesador, sistema distribuido homogéneo, sistema distribuido heterogéneo, componentes escritos en más de un idioma?

• Soporte de generación de aplicaciones: generación de código de componentes, generación de código de envoltura, generación de casos de prueba, generación de documentación.

Madurez, disponibilidad y soporte de la herramienta.

Soporte de proceso: ¿Existe un manual de usuario? ¿Existe un curso de formación?

5. Recopilación de datos

El cuestionario se ha distribuido a los propietarios de más de una docena de ADL hasta la fecha. La Tabla 1 contiene la lista de aquellos que han respondido y han validado sus resultados hasta la fecha. Otras encuestas en curso incluyen las de Rapide [12], Gestalt [17], ACME [4] y FR [8].

Para ganar confianza en el cuestionario, lo aplicamos a algunos idiomas, como Modechart, que no se considerarían ADL convencionales. La intención era observar cómo puntuaban estos lenguajes de "cúspide" para tratar de comprender las cualidades que distinguen a las ADL de las no ADL.

Tabla 1: AVD encuestadas (orden alfabético)

ADL	Fuente	Citación
Artek	Teknowledge	[19]
CÓDIGO	Univ. of Texas en Austin	[15]
Demeter	Northeastern Univ.	[dieciséis]
Modechart	Univ. de Texas en Austin	[10]
PSDL / TAPAS	Postgrado Naval. Colegio	[14]
Resolver	Univ. Del Estado de Ohio	[3]
UniCon	Carnegie Mellon Univ.	[18]
Wright	Carnegie Mellon Univ.	[1]

La intención del cuestionario era hacer que la mayor cantidad posible de preguntas fueran objetivas, en el sentido de que sería muy probable que dos observadores independientes respondieran de la misma manera a la pregunta. Cuando se solicitó subjetividad (por ejemplo, “¿qué tan bien el lenguaje soporta las representaciones arquitectónicas que son fáciles de cambiar?”), Le pedimos al encuestado que proporcionara características específicas del lenguaje y escenarios de uso que justificaran la respuesta. Sin embargo, ha sido necesario realizar una entrevista de validación para cada cuestionario con el fin de agregar credibilidad a las respuestas.

6. Conclusiones

Esta sección presentará los resultados de la encuesta hasta la fecha y postulará algunas conclusiones sobre las AVD en general.

6.1: Resultados de la encuesta

La Tabla 3 resume muchas de las preguntas detalladas en declaraciones resumidas sobre las capacidades de los idiomas. La Tabla 2 explica los símbolos de respuesta en la Tabla 3. La Tabla 3 sirve como una guía de referencia rápida para los lectores interesados en encontrar un idioma para una aplicación en particular. La intención es que la guía de referencia rápida proporcione al lector un pequeño conjunto de idiomas candidatos que probablemente se adapten a su propósito. Para detalles completos y selección final, deben consultarse las encuestas completas de los idiomas (no reproducidas en este documento).

Para crear la guía de referencia rápida, las encuestas completas se destilaron utilizando las siguientes heurísticas:

- Si fue posible combinar un conjunto de preguntas en una general sobre la capacidad del idioma sin perder detalles significativos, se hizo. La calificación de la capacidad general de un idioma es el promedio aritmético de sus calificaciones en las preguntas del componente (si la escala de calificación está ordenada, como Alto / Medio / Bajo), o la respuesta que apareció con más frecuencia en las preguntas del componente (si el la escala de calificación no está ordenada). Si las preguntas del componente fueron Sí / No, entonces la calificación resumida refleja la proporción de respuestas Sí.
- Las preguntas en las que todos los idiomas puntuaron igual o casi igual tendían a ser suprimidas, ya que no servían como discriminadores útiles entre los idiomas.

La información de sombreado se usa en las celdas para aumentar el contenido textual. Cuanto más claro sea el sombreado, más capaz será la calificación del idioma sobre el tema en particular.

6.2: ADL versus otros idiomas

Los idiomas que generalmente se consideran ADL "convencionales" comparten los siguientes aspectos:

- Las abstracciones que proporcionaron al usuario fueron de naturaleza arquitectónica. Todos los componentes y conexiones representados (aunque Rapide representa conexiones solo especificando comportamientos de comunicación entre componentes).
- La mayoría de las vistas proporcionadas por las ADL contenían principalmente información arquitectónica. Esto contrasta con un lenguaje de programación o un lenguaje de requisitos que tiende a mostrar otros tipos de información.
- El análisis proporcionado por el lenguaje se basa en información a nivel de arquitectura. El simulador de eventos discretos de Rapide, por ejemplo, utiliza información de comportamiento sobre cada componente para generar conjuntos de eventos parcialmente ordenados. La interfaz de UniCon para el analizador monótono de tasas utiliza información de rendimiento de caja negra sobre cada componente para calcular la capacidad de programación. Esto contrasta con un analizador de rendimiento que identifica los cuellos de botella en función de la información de implementación (es decir, el código).

6.3: Discriminadores

Las siguientes áreas revelaron diferencias interesantes entre las AVD que encuestamos:

- Las ADL diferían notablemente en su capacidad para manejar construcciones en tiempo real a nivel arquitectónico. Áspero-

ly la mitad afirmó lidiar con construcciones estrictas en tiempo real, como los plazos; sólo un pequeño número se ocupó de construcciones en tiempo real suave, como prioridades de tareas.

• Las ADL variaron en su capacidad para soportar la especificación de estilos arquitectónicos particulares. Todas las ADL podrían representar arquitecturas de tubería y filtro, ya sea directa o indirectamente. A otros estilos no les fue tan bien. Todos proporcionaban una estructura jerárquica de componentes y podían representar objetos, pero solo unos pocos podían manejar la herencia de clases orientada a objetos. Solo dos manejaron arquitecturas dinámicas.

• El conjunto de ADL encuestadas se divide en partes iguales en función de su capacidad para permitir que un usuario defina nuevos tipos de componentes y conectores, defina nuevas declaraciones en la ADL y represente información no arquitectónica (como requisitos o casos de prueba) utilizando la ADL. La extensibilidad y el alcance fueron buenos discriminadores.

• Si bien todos los lenguajes proporcionaron reglas de consistencia e integridad internas integradas para los artefactos renderizados en esos lenguajes, solo unos pocos permitieron al usuario definir lo que se entendía por consistencia, y solo unos pocos se ocuparon de la consistencia entre diferentes artefactos (p. Ej., Entre una arquitectura y diseños de componentes).

• Las ADL variaron ampliamente en su capacidad para respaldar el análisis. Rapide presenta un simulador de eventos discretos basado en conjuntos de comportamientos de eventos parcialmente ordenados. Modechart presenta un verificador de verificación de modelos que toma como entrada una aserción lógica e informa si la descripción del sistema garantiza, prohíbe o es simplemente compatible con esa aserción. UniCon interactúa directamente con un analizador de programación monótono de tasas. Una capacidad de análisis más general (como en el caso de un verificador) apareció en el cuestionario como la capacidad de analizar muchos atributos de calidad diferentes. Por ejemplo, se podría usar un simulador para analizar la usabilidad permitiendo a los usuarios observar el comportamiento simulado para ver si cumple con sus expectativas.

• Las ADL diferían en su capacidad para manejar la variabilidad o diferentes instancias de la misma arquitectura. Todos admitieron la variabilidad de los componentes a través de la capacidad de reescritura simple, pero pocos admitieron el mantenimiento de diferentes instancias de la misma arquitectura simultáneamente.

• Al ofrecer más de una vista arquitectónica, las ADL variaron ampliamente en su capacidad para traducir entre esas vistas. La intercambiabilidad de vistas es un fuerte discriminador entre las AVD que encuestamos.

6.4: Puntos en común

Se notaron los siguientes puntos en común:

• Todas las ADL encuestadas tenían una sintaxis gráfica; todos menos uno también tenían forma textual. Todos los idiomas menos uno presentaban una sintaxis formal, y la gran mayoría presentaba una semántica definida formalmente.

• Todos los lenguajes afirmaron ser capaces de modelar sistemas distribuidos.

• Las ADL tienden a no brindar mucho apoyo para capturar el fundamento y / o el historial del diseño, salvo a través de mecanismos de anotación genéricos o de propósito general.

• Todas las ADL afirmaron manejar el flujo de datos y controlar el flujo como mecanismos de interconexión. Modechart fue el más débil en este sentido, debido a su capacidad para tratar solo con predicados de estado (como "datos enviados") en lugar de valores de datos reales.

• Todas las ADL proporcionaron ayuda con la creación y validación y el refinamiento de arquitecturas, incluso si la validación solo se realizó en el contexto de las propias reglas del lenguaje para la integridad o legalidad.

• Todos presentaban la capacidad de representar niveles jerárquicos de detalle y manejar múltiples instancias de una plantilla como una forma rápida de realizar copias de subestructuras durante la creación.

• Todas las ADL respaldan al ingeniero de aplicaciones y la mayoría respalda al ingeniero de dominio, aunque solo sea indirectamente. La mayoría apoya al analista de sistemas al proporcionar capacidades analíticas iniciales. Ninguno afirmó apoyar directamente la gestión de proyectos.

• Finalmente, un rasgo común evidente fue la falta de experiencia profunda y aplicación en el mundo real que las ADL ofrecen actualmente. Es de esperar que a medida que se comprendan mejor los beneficios de la arquitectura, que los beneficios de las representaciones formales sean igualmente apreciados y que las ADL se conviertan en tecnologías viables para el desarrollo de sistemas complejos.

6.5: ¿Qué constituye una ADL?

Después de examinar una amplia selección de idiomas, todos los cuales pretenden ser ADL, ¿qué podemos concluir sobre lo que hace que un idioma sea ADL? Lo siguiente parece ser un conjunto mínimo de requisitos para que un idioma sea ADL:

• Una ADL debe respaldar las tareas de creación, perfeccionamiento y validación de la arquitectura. Debe incorporar reglas sobre lo que constituye una arquitectura completa o consistente.

• Una ADL debe proporcionar la capacidad de representar (aunque sea indirectamente) la mayoría de los estilos arquitectónicos comunes enumerados en [6].

• Una ADL debe tener la capacidad de proporcionar vistas del sistema que expresen información arquitectónica, pero al mismo tiempo suprimir información de implementación o no arquitectónica.

• Si el lenguaje puede expresar información a nivel de implementación, entonces debe contener capacidades para hacer coincidir más de una implementación con las vistas a nivel de arquitectura del sistema. Es decir, debe admitir la especificación de familias de implementaciones que satisfagan una arquitectura común.

• Una ADL debe admitir una capacidad analítica, basada en información a nivel de arquitectura, o una capacidad para generar rápidamente implementaciones de prototipos.

6.6: Tendencias futuras

Comunicar una arquitectura a una parte interesada se convierte en una cuestión de representarla en una forma legible e inequívoca que contiene la información apropiada para esa parte interesada. Las tendencias actuales en el desarrollo de ADL parecen centrarse en mejorar las capacidades de análisis y generación de sistemas de los lenguajes. Después de todo, la arquitectura es solo un medio para un fin, y la información que los desarrolladores pueden inferir sobre el sistema final es más valiosa que la información solo sobre la arquitectura. Ser capaz de desarrollar rápidamente un sistema o gestionar las cualidades del producto final es la verdadera recompensa de las ADL. Si bien el desarrollo de los lenguajes de arquitectura avanza a buen ritmo, se presta menos atención a las siguientes áreas:

- **Infraestructuras para apoyar el desarrollo de ADL.**La mayoría de las ADL comparten un conjunto de conceptos comunes. La creación de herramientas para respaldar una ADL implica la resolución de un conjunto común de problemas. El desarrollo de un entorno de desarrollo de ADL facilitaría la producción rápida de ADL y herramientas de apoyo, lo que permitiría que las buenas ideas lleguen al mercado más rápidamente. El trabajo de Garlan Aesop / ACME representa una contribución temprana e importante a esta área [4].

- **Integración de la información de ADL con otros productos de ciclo de vida.** A medida que las ADL maduren, asumirán un papel más destacado en la letanía de productos del ciclo de vida (como documentos de diseño detallados, casos de prueba, etc.). Debe fomentarse la consideración temprana de la relación que una descripción de la arquitectura tendrá con estos otros documentos. Por ejemplo, ¿qué casos de prueba se pueden generar para un sistema en una descripción de sus componentes y mecanismos de interconexión? ¿Qué tipo de código ejecutable y cuánto se puede generar automáticamente? ¿Cómo se puede establecer la trazabilidad de la arquitectura a los requisitos? Este trabajo podría culminar en la integración completa de las descripciones de la arquitectura en el entorno de desarrollo, dando lugar a una especie de "arquitectura". Esto se puede considerar como un entorno de exploración en el que se redactan arquitecturas, se validan mediante el mapeo de los requisitos, se exploran sus implicaciones mediante el análisis o la creación rápida de prototipos, se sugieren alternativas de una manera similar a un sistema experto y las infraestructuras del proyecto necesarias para el desarrollo (p. Ej., Se generan plantillas de horarios de trabajo, bibliotecas de control de configuración basadas en componentes, planes de prueba, etc.).

Es de esperar que los investigadores se sientan motivados a proporcionar ADL que aborden las brechas en la capacidad actual, particularmente en el análisis y el apoyo a la familia de programas.

7. Agradecimientos

Paul Kogut de Loral es cocreador del análisis de características de ADL y coautor del informe SEI que lo describe. Muchas de las ideas de este artículo son suyas. Agradecemos a los muchos creadores de ADL que completaron pacientemente nuestro cuestionario y nos permitieron venir a visitarnos y discutir los resultados de la encuesta en profundidad. Este trabajo no hubiera sido posible sin ellos. Un agradecimiento especial para David Garlan, David Luckham, Bruce Weide, John Hartman, Mary Shaw, JC Browne, y sus respectivos equipos de construcción de ADL, generalmente con exceso de trabajo.

El SEI está patrocinado por el Departamento de Defensa de EE. UU.

Tabla 2: Clave de respuesta

Símbolo	Sentido
Y	sí
norte	No
H	Alta capacidad: el lenguaje proporciona características explícitas para respaldar esta capacidad.
METRO	Medio: el lenguaje proporciona características genéricas a través de las cuales esta capacidad puede lograrse indirectamente.
L	Bajo: el lenguaje ofrece poco apoyo
T	La herramienta desarrollada específicamente para la ADL admite esta capacidad
mi	La herramienta externa proporciona esta capacidad
PAG	El idioma proporciona suficiente información para respaldar la capacidad, pero actualmente no existe soporte para herramientas.

Tabla 3: Resultados de la encuesta de ADL

Atributos	ARTmiK	CODmi	DmiMETROmiTmiR	METROODmiCHART	PAGSDL/CAPAGS	RmiSOLVmi	UnorteICOnorte	WRIGRAMOHT
Aplicabilidad								
Capacidad para representar estilos Capacidad para	METRO	METRO	METRO	METRO	H	METRO	METRO	H
manejar problemas en tiempo real	METRO		METRO	H	H	L	H	L
Capacidad para manejar problemas de sistemas distribuidos	H	H	H	H	H	METRO	METRO	METRO
Capacidad para manejar arquitecturas dinámicas	L	H	H	L	METRO	L	?	L
Calidad de la definición del lenguaje								
Atención a la integridad del arco. Especificaciones.	METRO	L	H	METRO	METRO	L	METRO	H
Atención a la consistencia del arco. Especificaciones.	L	L	METRO	L	METRO	H	L	H
Alcance del lenguaje; usuarios previstos								
Requisitos	H	L	?	H	H	METRO	L	L
Diseño detallado / algoritmos	L	METRO	H	METRO	H	METRO	L	L
Código	METRO	L	H	L	L	H	H	L
Ingeniero de dominio	Y	Y	Y	norte	Y	norte	Y	Y
Ingeniero de aplicaciones	Y	Y	Y	Y	Y	Y	Y	Y
Analizador de sistemas	Y	norte	Y	Y	Y	Y	norte	Y
Captura de vistas del historial de	?	L	?	L	?	?	?	?
diseño								
Textual	Y	norte	Y	Y	Y	Y	Y	Y
Gráfico	Y	Y	Y	Y	Y	norte	Y	norte
Riqueza de vista semántica	L	METRO	H	H	H	L	?	L
Referencia cruzada de la entrevista	METRO	L	L	METRO	METRO	L	METRO	L
Soporte para variabilidad Potencia expresiva,	METRO	L	H	L	H	H	L	H
extensibilidad Soporte para creación de	H	L	METRO	METRO	H	H	METRO	METRO
arquitectura Soporte para validación de	T	T	PAG	T	T	mi	T	
arquitectura Soporte para refinamiento de	T	T	mi	T	T	PAG	T	mi
arquitectura Soporte para análisis de	T	PAG	PAG	T	T	PAG	PAG	PAG
arquitectura Soporte para construcción de	mi		PAG	T	T	PAG		
aplicaciones Madurez de herramienta		T	PAG		PAG			
Disponible como CUNAS?	norte	norte	?	norte	norte	norte	norte	norte
Años de edad)	3	?	10	7	5	?	2	3
Número de sitios en uso	11	?	?	4	12	4	1	1
Atención al cliente disponible	Y	Y	norte	norte	norte	Y	norte	norte

8. Referencias

1. Allen, R., Garlan, D. "Más allá de la definición / uso: interconexión arquitectónica", *Proceedings, Workshop on Interface Definition Languages,* Portland, Oregón, 20 de enero de 1994l.

2. Clements, P., Gasarch, C., Jeffords, R. “Criterios de evaluación para lenguajes de especificación en tiempo real”, Informe del Memorando del Laboratorio de Investigación Naval 6935, febrero de 1992.

3. Edwards, S., Heym, W., Long, T., Sitarman, M. y Weide, B .; “Especificación de componentes en RESOLVE”, Notas de ingeniería de software, vol. 19, no. 4, octubre de 1994.

4. Garlan, D., Allen, R., Ockerbloom, J. "Exploiting Style in Architectural Design Environments", *Actas de SIGSOFT '94: Fundamentos de la ingeniería de software,*Diciembre de 1994, ACM Press.

5. R. Kazman, L. Bass, G. Abowd y M. Webb, "SAAM: un método para analizar las propiedades de las arquitecturas de software". En*Actas de la 16a Conferencia Internacional de Ingeniería de Software*, (Sorrento, Italia), mayo de 1994, págs. 81-90.

6. Garlan, David; y Shaw, Mary.*Introducción a la arquitectura de software*. (CMU / SEI-93-TR-33). Pittsburgh, Pa .: Software Engineering Institute, Carnegie Mellon University, diciembre de 1993. También en Ambriola, V .; y Tortora, G. (eds.), Advances in Software Engineering and Knowledge Engineering, Volume I. Singapur: World Scientific Publishing, 1993.

7. Harel, D .; Lachover, H .; Naamad, A .; Pnueli, A .; Politi, METRO.; Sherman, R .; Shtul-Trauring, A .; “STATEMATE: un entorno de trabajo para el desarrollo de sistemas reactivos complejos” Actas de la 10ª Conferencia Internacional sobre Ingeniería de Software en Singapur, abril de 1988;

8. Hartman, J., Chandrasekaran, B., "Representación funcional y comprensión del software: tecnología y aplicación", *Actas, Conferencia de 1995 sobre tecnologías y aplicaciones de doble uso,* 1995.

9. IEEE Std 610.12, *Glosario estándar IEEE de terminología de ingeniería de software,* Septiembre de 1990.

10. Jahanian, F. y Mok, A. "Modechart: A Specification Language for Real-Time Systems", *Transacciones IEEE sobre ingeniería de software,* vol. 20, no. 12, diciembre de 1994, págs. 933-947.

11. Kang, Kyo C .; Cohen, Sholom G .; Hess, James A .; Novak, William E .; Y Peterson, A. Spencer.*Estudio de viabilidad del análisis de dominio orientado a características (FODA)*(CMU / SEI-90-TR-21, ADA235785). Pittsburgh, PA: Instituto de Ingeniería de Software, Universidad Carnegie Mellon, noviembre de 1990.

12. Luckham, David, John J. Kenney, Larry M. Augustin, James Vera, Doug Bryan, Walter Mann. “Especificación y análisis de la arquitectura del sistema mediante Rapide”, informe técnico de la Universidad de Stanford, 1993.

13. Luckham, David, Vera, James. "Un lenguaje de definición de arquitectura basada en eventos"*Transacciones IEEE sobre ingeniería de software,* a aparecer.

14. Luqi, Shing, M., Barnes, P. y Hughes, G. “Creación de prototipos de sistemas Ada en tiempo real en un entorno de clase”, *Actas del Séptimo Simposio Anual de Educación y Capacitación en Ingeniería de Software de Ada (ASEET)*, Monterey, 12-14 de enero de 1993.

15. Newton, P., Browne, J. "El lenguaje de programación gráfica paralela CODE 2.0", *Actas, ACM International Conference on Supercomputing,* Julio de 1992.

16. Palsberg, J., Xiao, C., Lieberherr, K. “Efficient Implementation of Adaptive Software (Summary of Demeter Theory)”, Northeastern University, Boxton, 10 de enero de 1995.

17. Schwanke, R., “Arquitectura de software industrial con Gestalt”, informe técnico, Siemens Corporate Investigación, Princeton NJ.

18. Shaw, DeLine, Klein, Ross, Young, Zelesnik "Abstracciones para arquitecturas de software y herramientas para respaldarlas", Universidad Carnegie Mellon, informe inédito de febrero de 1994

19. Terry, Hayes-Roth, Erman, Coleman, Devito, Papanagopoulos, Hayes-Roth "Descripción general del programa DSSA de Teknowledge", *Notas de ingeniería del software ACM SIGSOFT*, Octubre de 1994.

20. Vestal, S. *Una descripción general y una comparación de cuatro lenguajes de descripción arquitectónica*, informe técnico, Honeywell, febrero de 1993

21. Vestal, S. “Cambios de modo en un lenguaje de descripción de arquitectura en tiempo real”, Actas, Proc. Taller internacional sobre sistemas distribuidos configurables: Honeywell Technology Center y el Universidad de Maryland.