Journal of Engineering Sciences and Information Technology
Volume (3), Issue (4): 30 Dec 2019
P: 55 - 68

ISSN: 2522-3321

مجلة العلوم الهندسية وتكنولوجيا المعلومات
المجلد (3)، العدد (4): 30 ديسمبر 2019 م
ص: 55 - 68

Architecture Description Language – Systematic Literature Review

Tahani Elfatih Babeker
Faculty of Computer Science and information Technology || Sudan University for Science and Technology || Sudan
Hany Ammar
Lane Department of Computer Science and Electrical Engineering || West Virginia University

Abstract: increase numbers and complexity of software development requires that learners of software engineering and software architecture or software architects who do not have sufficient practical experience must have the skills and abilities to perform their jobs. But there is a gap between academia and practical so the question why does not use one of the architecture description languages in teaching and learn software engineering and software architecture to fill this gap?
The objective of this study to classification ADLs according to their domain, domain specific or general purpose languages, doing these systematic literature review flowing the steps from Ketchenham.
As the result we find that most ADLs are Domain Specific ADLs (Aviation Systems, Distribution Systems, Mobile Systems, Product Lines ...etc.) none of these ADLs is used in the field of software architecture education, but most intended to deal with experts in the field, except general purpose ADLs, like ACME, which we make as start point to design Arabic ADL, so as to enrich Arabic content, also to be as helping language in teaching Software Architecture course, because most of ADLs need experience and high level of understanding to use, moreover, when student think and design with same language subject more understandable.
So using software architecture description languages in educating purpose, help on understanding high level of abstraction of software architecture and software engineering courses, there are some studies that aimed to reduce the complexity of these courses using different methodologies and approaches, but none of them using architecture description languages as helping tool. So the use of architecture description languages is helpful in teaching software architecture and software engineering courses.

Keywords: Software Architecture, Architecture Description Languages, Software Architecture Education, Component Based Architecture.

دراسة استقصائية حول لغات توصيف معمارية البرمجيات

تهاني الفاتح بابكر
كلية علوم الحاسوب وتقانة المعلومات || جامعة السودان للعلوم والتكنولوجيا || السودان
هاني عمار
قسم علوم الحاسوب والهندسة الكهربائية || جامعة وست فرجينيا

الملخص: أن الزيادة المضطردة في إعداد البرمجيات وزيادة درجة تعقيدها، تحتم على دارسي هندسة البرمجيات ومعمارية البرمجيات أو على معماريي البرمجيات الذين ليس لديهم الخبرة العملية الكافية أن يكونوا على قدر من المهارات والمقدرات التي تمكنهم من أداء وظائفهم. ولكن على أرض الواقع هناك فجوة بين المعرفة الأكاديمية وبين المقدرة العملية المطلوبة في سوق العمل فكان السؤال لماذا لا

تستخدم إحدى لغات توصيف المعمارية كلغة مساعدة حتى يتمكن المتلقي من ربط المفاهيم المجردة لهندسة البرمجيات ومعمارية البرمجيات؟

تهدف هذه الدراسة الاستقصائية إلى تصنيفات لغات توصيف معمارية البرمجيات (Software Architecture Description Languages) أو (ADLs) الحالية والتعرف على مدى فاعليتها في تعليم معمارية البرمجيات. وذلك باستخدام طريقة الدراسة الاستقصائية المنتظمة (systematic Literature Review)، متبعين التوجهات المقترحة من (Kitchenham).

لغرض تصميم لغة لتوصيف معمارية البرمجيات باللغة العربية لكي تكون مادة مساعدة في تدريس مقرر معمارية البرمجيات، معتمدين على مفهوم البرمجيات القائمة على المكونات (Components).

خلصت النتائج إلى أن معظم لغات توصيف المعمارية الحالية موجه إلى مجال معين (Domain Specific)، مثل (أنظمة الطيران، الأنظمة الموزعة، الهواتف النقالة، خطوط الإنتاج،...الخ) أضف إلى ذلك أنه ليس هناك واحدة من هذه اللغات مستخدمة في مجال تعليم معمارية البرمجيات، بل إن معظمها إن لم يكن كلها موجهة للتعامل مع خبراء في المجال، عدا بعض اللغات ذات الأغراض العامة والتي قد تكون بسيطة إلى حد ما مثل أكمي (ACME)، والتي كانت النواة للغة المراد تصميمها. وهي لغة لتوصيف معمارية البرمجيات، والتي تستخدم العبارات والصياغة باللغة العربية لإثراء محتوى اللغة العربية التقني وكذلك لتكون معين للطلاب الذين يدرسون مادة معمارية البرمجيات في دراستهم، نسبة لأن كل لغات توصيف المعمارية تعتمد على أن الذين يستخدمونها من الذين لديهم خبرة في معمارية البرمجيات، أي انها تعتمد على مستوى علمي وعملي عالٍ. أضف إلى ذلك عندما ينفذ الطالب معمارية البرمجيات بنفس اللغة التي يفكر بها يصبح ذلك بالنسبة إليه أوضح. أما من ناحية تعلم معمارية البرمجيات فإن استخدامه أداة للتعلم يساعد في تقليل المستوى العالي من التجريد (Abstraction) التي تتميز به مادة معمارية البرمجيات. فكانت هناك دراسات هدفت إلى اقتراح طرق وأساليب مختلفة لتسهيل تعلم معمارية البرمجيات وهندسة البرمجيات، ولكن ليست هناك واحدة منها اقترحت بناء لغة معمارية تخدم هذا الغرض. لذلك فإن استخدام لغة توصيف المعمارية للأغراض التعليمية يساهم إلى حد ما في استيعاب مقررات معمارية البرمجيات وهندسة البرمجيات.

الكلمات المفتاحية: معمارية البرمجيات، لغات توصيف معمارية البرمجيات، تعليم معمارية البرمجيات.

1- مقدمة

هناك حوالي 150 لغة لتوصيف معمارية البرمجيات ولا توجد تصنيفات واضحة لهذه اللغات من قبل مجتمع هندسة البرمجيات فقد كانت هناك دراسة استقصائية سابقه قامت بتصنيف هذه اللغات إلى لغات ذات طبيعة أكاديمية أي أنها كانت نتيجة بحوث ودراسات أكاديمية، ولغات أخرى ذات طبية صناعية، أي أنها تعتبر منتج.

إن التزايد المستمر والمتسرع في صناعة البرمجيات أدى إلى الحاجة إلى تصميم البرمجيات بطريقة واضحة ومرتبه كما وأن الزيادة الكثيرة في أعداد لغات توصيف المعمارية مع وجود فارق بسيط في القدرات والاهتمام أو المجال الذي تعمل فيه تلك اللغات نتج عن تعدد المستفيدين والمستخدمين لهذه اللغات واهتماماتهم. تمثل معمارية البرمجيات عدة وجهات نظر وظيفية وتشغيلية تسهم في اتخاذ القرار على حسب متطلبات البرمجيات، من أهم متطلبات لغة توصيف معمارية البرمجيات أن تفي وتغطى الاحتياجات المطلوبة.

كما وأن المستوى العالي من التجريد الذي تحققه لغات وصف المعمارية يسهل فهم عمل المتغيرات في النظام.

2.1 خلفية الدراسة:

الزيادة الكبيرة في أعداد لغات توصيف معمارية البرمجيات مع وجود اختلافات طفيفة في الإمكانيات والاهتمام [1][2] وذلك نتيجة لاختلاف المستخدمين واحتياجاتهم، فقد خلق هذا سوء فهم في خصائص لغات توصيف معمارية البرمجيات.

لتوصيف معمارية البرمجيات هناك طرق منهجية (Formal Methods) وذلك باستخدام لغات توصيف معمارية البرمجيات أو غير منهجية وذلك باستخدام الرموز والأشكال. الطرق المنهجية توفر سهولة التحليل والتعليل وكذلك الصيانة والمراجعة للمعمارية كما انها مفهومه لكل المشاركين من معماريين ومصممين وكذلك المستخدمين، أضف إلى ذلك تعرف خارطة الطريق للبرمجيات وتأكد من كل المتطلبات قد تم تحقيقها. الطرق غير المنهجية غير مجدية لأنها لا تعطى المعلومات الكافية عن وحدات المعمارية.

1.2.1 لغات توصيف معمارية البرمجيات

هناك عدة تعريفات للغات توصيف معمارية البرمجيات منها:

التعريف الأول:

هيكلة وحدات النظام أو التطبيق، والعلاقات بين هذه الوحدات مع وجود مبادئ وقواعد ارشادية تتحكم في التصميم والتطوير[3].

التعريف الثاني:

لغة حاسوب لتوصيف معمارية البرمجيات مثل العمليات، البيانات والبرامج الفرعية. أو لتوصيف معمارية المكونات أو العتاد مثل المعالجات، الناقل والذاكرة.

عموما تم تقسيم لغات توصيف هيكلة البرمجيات إلى لغات الجيل الأول من العام 1990 حتى العام 2000، ولغات الجيل الثاني من العام 2000 إلى الان.

وهناك بعض الدراسات للوصول للغات الجيل الثالث[4][5]

2.1. 2. أسئلة الدراسة

1- كيف يمكن تقييم لغات وصف معمارية البرمجيات الحالية؟.

2- إلى أي مدى للغات وصف معمارية البرمجيات أن تساهم في تعلم مادة معمارية البرمجيات؟.

3.2.1 أهداف الدراسة

تهدف هذه الدراسة إلى:

1. البحث في الدراسات والمقالات المنشورة والتي تتحدث عن لغات معمارية البرمجيات وتصنيفها من حيث المجال وفاعليتها في التعليم أو استخدامها كمادة مساعدة في تدريس مادة معمارية البرمجيات.
2. معرفة تصنيفات أنواع لغات معمارية البرمجيات، هل هي لغات مرتبطة بمعمارية مجال معين أو هل هي لغات معمارية ذات أغراض عامه تصلح لوصف برمجيات مجالات عدة.
3. معرفة إمكانية استخدام لغات معمارية البرمجيات في تعلم مادة معمارية البرمجيات.

2- منهجية الدراسة

تم تقسيم الورقة كما يأتي: القسم 2 يتم فيه توضيح الخطوات المتبعة في الدراسة الاستقصائية وأسئلة البحث وشرح كيفية البحث في قواعد البيانات ومحركات البحث وهيكلة الأسئلة المستخدمة في عملية البحث وشرح كيفية تضمين أو استبعاد الدراسات المتحصل عليها نتيجة البحث مع وضع معايير لجودة كل دراسة.

في القسم 3 نتعرف على نتيجة الدراسة الاستقصائية بطريقة تتوافق وأسئلة البحث.

القسم 4 مناقشة النتائج وفيه نستعرض الدراسات التي أسهمت في تسهيل تعلم معمارية البرمجيات وأيضا بعض الدراسات المتعلقة بلغات توصيف معمارية البرمجيات. القسم 5 الخلاصة ثم التوصيات في القسم 6 تليها المراجع ومن ثم جدول المصطلحات المستخدمة في الدراسة.

3- طريقة البحث

استخدمنا طريقة الدراسة الاستقصائية المنتظمة متبعين التوجيهات المقترحة من Kitchenham في [6]، وهي عملية تعريف وتفسير لجميع الدراسات المتاحة والمتعلقة بأسئلة البحث.

2.3 عملية البحث

تمت عملية البحث في الأوراق العلمية والمقالات وبحوث الماجستير والدكتوراه الموجودة في قواعد بيانات ومحركات البحث [6][7]، كما موضح في جدول (1).

جدول (1) مصادر البحث

المصدر	الاسم
قواعد البيانات	ACM Digital Library
	IEEE Explore
	INCPEC
	Science Direct
	Springer link
	Research Gate
محركات البحث	Google Scholar
	CiteSeer

وذلك بعد أن تم وضع عبارات البحث في صورة (PICOC) [6][7]. فكانت هيكلة الأسئلة كما في جدول (2)

جدول (2) عبارات البحث في صورة (PICOC)

المجموعة	مهندسو البرمجيات، معماري البرمجيات، الطلاب
المنهجيات أو التقنيات	لغات معمارية البرمجيات المختلفة، لغات معمارية البرمجيات المستخدمة في التعليم، لغات معمارية البرمجيات المستخدمة في تدريس تصميم البرمجيات، بناء معمارية البرمجيات باستخدام لغات توصيف معمارية البرمجيات، تأثير لغات معمارية البرمجيات في تصميم وبناء البرمجيات
المقارنة	طرق لمقارنة لغات معمارية البرمجيات
النتيجة	تصنيف لغات معمارية البرمجيات، مدى نجاح لغات توصيف معمارية البرمجيات في تعلم مادة معمارية البرمجيات
السياق	في مجال هندسة البرمجيات، لغات توصيف معمارية البرمجيات(مجال معين، أغراض عامه)، تدريس معمارية البرمجيات، دراسة تحليلية

3.3 معايير التضمين والاستبعاد

بعد قراءة العنوان والملخص لكل منشور ظهر في نتيجة البحث، تم تضمين الأوراق العلمية والمقالات ذات الصلة بلغات وصف معمارية البرمجيات من حيث تعريف اللغة وشرح مكوناتها وإمكانياتها وتوضيح نوع المجال الذي تستخدم فيه اللغة، لم يتم التقييد بفترة زمنية وذلك بأن هناك العديد من اللغات كانت لها اصدارة أولى وتعتبر من لغات الجيل الأول بالإضافة للغات الحديثة.

تم استبعاد الأوراق والمقالات التي تتحدث عن لغات معمارية العتاد، والأوراق التي اشارت إلى بعض لغات معمارية البرمجيات دون تفصيل نوع اللغة ومجالها، كذلك الأوراق التي لم يتم الحصول على نصها كاملا، بالإضافة للأوراق المكررة لنفس الدراسة.

4.3 معايير جودة الدراسة

حتى نتجنب الحياد في الدراسة أوضحت kitchenham في أن هناك معايير إضافة إلى معايير التضمين وذلك لإعطاء تفاصيل أكثر عن الدراسة وجودتها وضمان أن الدراسة تسلك السلوك الصحيح لاستخلاص الحقائق، هذه المعلومات ضرورية عند تجميع البيانات وتفسير النتائج.

جدول (3) يوضح العوامل التي تؤخذ في الاعتبار لتقييم الدراسة

جدول (3) قائمة تقييم جودة الدراسة

مسلسل	الرمز	السؤال	الاجابة
1	مع1	هل معايير التضمين والاستبعاد واضحة ومناسبة؟	نعم / لا
2	مع2	هل معايير التضمين غطت كل الدراسات ذات الصلة؟	نعم / لا / جزئيا
3	مع3	هل تم اخذ جودة الدراسات المتضمنة في الاعتبار؟	نعم / لا / جزئيا

يوضح جدول (4) تقييم للإجابات

جدول (4) تقييم الإجابات

الإجابة	القيمة
نعم	1
لا	0
جزئيا	0.5

5.3 تجميع وتحليل البيانات

بعد تطبيق معايير التضمين والاستبعاد على الدراسات المتحصل عليها نتيجة البحث في المصادر المذكورة في جدول (2)، تم تجميع البيانات وفقا ل:

1- المصدر: مصدر الدراسة المعنية.

2- مجال أو نطاق الدراسة (هل اللغة لمجال معين أو أغراض عامه، الجداول (5)و(6) و(7)).

جدول (5) لغات توصيف معمارية البرمجيات (مجال محدد أم ذات أغراض عامة)

اللغة	مجال محدد	أغراض عامه	المصدر	العام
AADL	✓		[8]	2006
ABC/ADL		✓	[9]	2002
ACME		✓	[12][1] [11][10]	1997

اللغة	مجال محدد	أغراض عامه	المصدر	العام
AC2_ADL		✓	[13]	2008
ABACUS		✓	[14]	2005
ADAGE	✓		[15]	1995
ADLARS	✓		[16]	2005
ADLV		✓	[17]	2008
ADR	✓		[18]	2008
AEMILIA		✓	[19]	2002
AESOP	✓		[1]، [10]، [20]، [21]	1994
AIL-Transport	✓		[22]	2002
ALI	✓		[23]	2008
Ambient-PRISMA	✓		[24]	2005
AO-ADL	✓		[25]	2007
Arch Face	✓		[26]	2010
Armani	✓		[27]	2000
ASDL	✓		[28]	1994
Aspectual Acme		✓	[29]	2006
Autosar	✓		[30]	2010
DAOP-ADL		✓	[31]	2003
Darwin		✓	[20][37][36][1][35][34][33][11][32]	1995
KOALA	✓		[38]	2000
KADL	✓		[34]	2006
MetaH	✓		[40][39]	2000
Rapid		✓	[42][41]	1997
xADL		✓	[37][7][43][11]	2002
Wright		✓	[20]	1997
Weaves		✓	[12]	2000
πADL	✓		[44]	2004
UniCon		✓	[35][21][20][33]	1997

جدول (6) ملخص لغات توصيف معمارية البرمجيات

اللغات التي شملتها الدراسة	مجال محدد	ذات أغراض عامه
31	17	14

جدول (7) ملخص الدراسات التي اسهمت في تعلم معمارية البرمجيات

مسلسل	العام	الطريقة المتبعة	المصدر
1	2009	مجتمع المتعلمين (Community of learners)	[45]
2	2017	منهج المشاركة	[46]
3	2006	الكتابة	[47]
4	2006	UML	[48]

مسلسل	العام	الطريقة المتبعة	المصدر
5	2004	لغة توصيف المعمارية	[49]
6	2015	HUSACCT	[50]
7	2010	مفهوم التجريد	[51]

4. مميزات لغات توصيف معمارية البرمجيات

الهدف من هذه الدراسة الاستقصائية التعرف على لغات معمارية البرمجيات المعروفة، ومدى فاعليتها في تعليم معمارية البرمجيات كلغات مساعده.

نجد أن في [52] تم حصر كل لغات معمارية البرمجيات الموجودة حاليا. في هذه الدراسة الاستقصائية تم تضمين اللغات التي تم الحصول على مصدرها، جدول(5).

كل من هذه اللغات لها ميزاتها الخاصة كما وأنها تعنى بتصميم معمارية البرمجيات من أوجه نظر مختلفة، مثلا نجد لغات ADLARS،KOALA، ALI تعنى بتصميم معمارية برمجيات خطوط الإنتاج. AADL لغة لتوصيف معمارية الأنظمة المدمجة المقيدة بالزمن الحقيقي (Embedded real time systems) وأيضا مناسبة للأنظمة المعقدة، ABC/ADL لغة ذات أهداف عامة تدعم البرمجيات القائمة على المكونات، وأيضا إعادة استخدام المكونات. ACME تعتبر لغة توصيف معمارية ذات أغراض عامة وتستخدم كأداة للتحويل بين لغات معمارية البرمجيات. AC2_ADL لتصميم معمارية واجه المستخدم باستخدام نمط المعمارية القائم على الرسائل فهي مناسبة لمعمارية الأنظمة الموزعة والمتغيرة والمتطورة. ABACUS تقوم بهيكلة عدة حلول بعد التحليل ومحاكاة خيارات المعمارية. ADAGE تستخدم لمعمارية أنظمة الطيران.ADLARS تعنى بمعمارية خطوط الإنتاج لسد الفجوة بين نموذج الخصائص (Pattern model) ونموذج المعمارية. ADLV تسمح بتعريف سلوك وخصائص النظام في الأنظمة القائمة على المكونات. ADR مناسبة لإعادة تصميم المعمارية وكذلك للأنظمة المتحركة. AEMILIA تعنى بالنمذجة الهرمية والتجميعية لمعمارية البرمجيات ومعرفة عدم التطابق في المعمارية من خلال التحليل الوظيفي لأداء المعمارية. AESOP تدعم استخدام الإنماط المعمارية. AIL-Transport لتصميم معمارية برمجيات الأنظمة المدمجة للسيارات. ALI مبنية على أساس ADLARS لتغطية بعض جوانب القصور في لغات معمارية البرمجيات من عدم مرونة ونقص في الأدوات المستخدمة.Ambient-PRISMA تختص بمعمارية الأجهزة المحمولة والنقالة وكذلك الشبكات الهرمية الموزعة. AO-ADL لحل مشكلة الخصائص والمميزات التي تكون مشتركة بين المكونات والروابط قامت هذه اللغة بإضافة نوع اخر من المكونات والروابط. Arch Face لسد الفجوة بين التصميم والتطبيق فهي تعتبر لغة توصيف تعتمد على المكونات والروابط ولغة برمجة لمقابلة التطور في النظام والواجهات. Armani لغة معمارية ذات أغراض عامة توفر بيئة للتحليل. ASDL لمعمارية الأنظمة الصناعية الموزعة. Aspectual Acme أضافت إلى ACME نوع خاص من الروابط. Autosar تعنى بتصميم معمارية برمجيات السيارات. DAOP-ADL هناك بعض الميزات التي تكون مشتركه بين المكونات، تعنى بالأنظمة المعقدة الموزعة.Darwin تدعم التركيب الهرمي للنظام بحيث يمكن أن يكون المكون يحتوى على عدة مكونات أخرى مما يدعم الأنظمة الموزعة.KOALA لمعمارية البرمجيات القائمة على المكونات. MetaH تعنى بمعمارية أنظمة التحكم في الطيران.

Rapid تعنى بمعمارية البرمجيات القائمة على نمط الحدث (Event Based) فهي تقوم بعمل محاكاة للأحداث ومن ثم تحليل النتائج لبناء المكونات والوصلات(Connectors). xADL هذا الاطار يتعامل مع لغات معمارية البرمجيات المختلفة كأنها وحدات مستفيد من خاصية التمديد في، فهناك وحدة خاصة بمعمارية برمجيا الطيران وآخر خاصة

بمركبات الفضاء وهكذا يمكن أن تكون كل وحدة لغة قائمة بذاتها.Wright تدعم التفاعل بين مكونات البرمجيات، حيث يتم تقسيم النظام إلى وحدات كل وحدة بها مجموعة من المكونات والوصلات ومن ثم دراسة وتحليل التفاعل بينهم. Weaves من لغات الجيل الأول تستخدم في بناء معمارية البرمجيات لها واجه تطبيق. πADL عند بناء معمارية لبرمجيات متغيرة يجب أن يتضح هذا التغير عند التشغيل، فكانت هذه اللغة التي تعني ببناء المعمارية من ناحية التركيب وسلوك البرمجيات.UniCon تمتاز ببناء معمارية من مكونات و روابط مختلفة الأنواع.

5. الدراسات السابقة:

بما أن هذه الدراسة حول لغات معمارية البرمجيات، لتصميم لغة تعليمية لتوصيف معمارية البرمجيات، فإن الجزء الاول من الدراسات السابقة يتحدث عن الدراسات الاستقصائية السابقة والجزء الثاني يتحدث عن الدراسات التي قامت بتطوير أو تغيير في بعض لغات معمارية البرمجيات لحل مشكلة ما أو لكى تصبح اللغة ملائمه للعمل في مجال ما، اما الجزء الثالث فيه الدراسات التي تتحدث عن تعلم هندسة البرمجيات ومعمارية البرمجيات وذلك باستخدام طرق ومنهجيات جديدة تمكن من زيادة استيعاب الطلاب لمقرر هندسة البرمجيات ومعمارية البرمجيات.

1.5 الدراسات الاستقصائية

تمت مقارنة بعض لغات توصيف معمارية البرمجيات و نماذج هيكلة البرمجيات وطرق تحليل البيانات [31]، وفي [4] حيث تمت مقارنة لغات تعتبر من لغات الجيل الأول Darwin, Wright, Rapide, UniCon, C2, LEDA, Koala بلغات مستخدمه حديثا SOFA, Pilar, PRISMA, COSA, AADL, CONNECT في بعض الخصائص التي تعتبر مهمة جدا لأى لغة توصيف معمارية وهى الدلالات(Semantics) الرسمية والقابلية للاستخدام ومدى جودتها.

وخلصت المقارنة إلى أن كل اللغات التي تمت دراستها صعبة الاستخدام وخاصة عند توصيف الأنظمة المعقدة، وعزا ذلك لثلاث أسباب:

- عدم قدرة تلك اللغات في إنتاج أو إعطاء تحليل رسمي أو واضح للمعمارية المصممة.
- الرموز المستخدمة في توصيف معمارية الأنظمة المعقدة غير مفهومه وصعبة.
- مما نتج عنه احتمال عدم استخدام المعمارية المتحصل عليها.

نتيجة لذلك قام بتطوير XCD لغة جديده لتلافى أوجه القصور التي سبق ذكرها.

الجدير بالذكر أن هذه الدراسة ركزت على ثلاث خصائص دون البقية، كما انها لم تأخذ عدد أكبر من اللغات في الاعتبار.

قام [7] في بحثه بإجراء دراسة استقصائية للغات معمارية البرمجيات، درس معظمها من ناحية دورها هل هي لغات تستخدم لأغراض اكاديمية بحثية ام هل هي ذات أغراض تجارية أي منتجات تجارية. وقام أيضا بدراسة الفوائد وجوانب القصور فيها، وكذلك عوامل جودة البرمجيات عند استخدامها في معمارية البرمجيات.

وتوصل إلى أن معظم لغات معمارية البرمجيات ذات أغراض اكاديمية والقليل ذات طبية تجارية، كما أوضح أن لزيادة اللغات للأغراض التجارية هناك حاجه لفهم الاستخدام الجيد لهذه اللغات، مما يشير إلى أن هناك جوانب قصور في فهم طبيعة وعمل لغات معمارية البرمجيات.

لم تتطرق الدراسة لكيفية استخدام لغات معمارية البرمجيات في أنماط معمارية مختلفة، وأيهم يمكن أن يفي بمتطلبات نمط معين.

واخرين [5] قاموا بدراسة حول لغات معمارية البرمجيات وذلك لسد الثغرة بين ما تقدمه البحوث والدراسات الاكاديمية وبين الاحتياجات الحقيقية في سوق العمل من ناحية ما تقدمه تلك اللغات. وذلك بعمل استطلاع تضمن ثلاث فئات وهم الباحثين الأكاديميين في مجال لغات معمارية البرمجيات ومعماري البرمجيات والباحثين الصناعيين الذين يستخدمون لغات معمارية البرمجيات.

وخلصت الدراسة إلى أن هناك بعض الخصائص والمميزات التي يجب أن تحققها لغات معمارية البرمجيات وهي أمكانية على التصميم، مقدرة اللغة على تغطية احتياجات كل المشاركين في النظام والتحليل، إضافة إلى ذلك اجمع معظم المشاركين في الاستطلاع على انهم يستخدمون اللغات تجارية المنشأ، مما شير إلى أن لغات المعمارية اكاديمية المنشأ بها قصور ولا تغطى احتياجات معماري البرمجيات.

كذلك يجب أن تكون لغة المعمارية بسيطة وواضحة حتى يتم توصيف احتياجات المشاركين كما وأيضا أن تكون لها أدوات كي تجعل عملية التوصيف المعمارية سهلة، لان اللغات المعقدة يتم استخدامها في نطاق ضيق.

2.5 الدراسات ذات الصلة بلغات معمارية البرمجيات

بالإشارة إلى [1] الذى تحدث باختصار عن بعض لغات معمارية البرمجيات المعروفة وقتها مع إعطاء امثلة لبعض منها وكذلك بعض المؤتمرات التي اختصت بلغات معمارية البرمجيات.

لم يذكر كل اللغات ولم يذكر مواصفات كل لغة ولا المجالات التي تعنى بها تلك اللغات.

[36] قارن بين لغات معمارية البرمجيات في نطاق دورها في تصميم وبناء معمارية البرمجيات القائمة على الوحدات(معمارية البرمجيات القائمة على الوحدات) ومدى فاعليتها في هذا المجال، لم تذكر الورقة أي أنماط أخرى كذلك لم تأخذ كل اللغات في الاعتبار.

وتوصلت الدراسة إلى الدور الرئيسي والمهم للغات معمارية البرمجيات في تصميم البرمجيات القائمة على المكونات لأنها تبنى المعمارية معتمدة أساسا على المكونات والوصلات والشروط (Constraint) التي تحكم تلك العلاقة بينهم وهي التهيئة(Configuration). كما أعتبر أن الUML لغة توصيف معمارية بينما يعتبرها البعض لغة أداة لتصميم المعمارية، وأشار إلى أن الحل لتعدد لغات معمارية البرمجيات يجب أن تكون قياسية وعالمية. اعتبر أن ACME يمكن أن تكون لغة تحويلية بين جزء كبير من لغات معمارية البرمجيات.

وضع [12] إطار لتصنيف و مقارنة لغات معمارية البرمجيات اخذا في الاعتبار اللغات (Wright،Weaves،Unicon،SADL،Rapide،MttaH،C2،Darwin،Aesop،Acme)، قام بمقارنتها من ناحية وحدات بناء المعمارية وهى المكونات،الوصلات و التهيئة أو ترتيب المعمارية. ووضع الخصائص الضرورية لكل من هذه الوحدات فمثلا خصائص المكونات والوصلات هي الواجهة(Interface) المراد بها واجهة التعامل بين المكونات والوصلات والدلالات والمقصود بها تصميم سلوك المكون، والشروط أو القيود التي تحكم تصميم البرمجيات، وأيضا قابلة اللغة لمقابلة التطور في المتطلبات كذلك مدى دعمها للخصائص غير الوظيفية.

هذه الدراسة اعتمدت على نمط واحد وهو نمط البرمجيات القائمة على المكونات كما انها لم تأخذ في الاعتبار المجالات المعنية بكل لغة اخذتها مجردة، بالإضافة للعدد المحدود للغات قيد الاختبار.

كتبت ماري شو وباول كلمنت في المقال [53]، عن تأثير إنماط معمارية البرمجيات في لغات معمارية البرمجيات متسائلين عن ماهي الإمكانيات المطلوبة من اللغات لتصميم الإنماط المعمارية والى أي مدى يمكن لها أن تغطى المعلومات المطلوبة عن النمط المعين، داعين مجتمع معمارية البرمجيات إلى المشاركة لوضع نقاط و تصميم اطار واضح لتلك التساؤلات.

تحدث [54] في جزيئه من بحثه عن لغات معمارية البرمجيات وتحديدا اللغات المختصة بالتفاعل بين مكونات النظام مثل (Darwin،Wright،SOFA،PcMSEFF،Nets p Fractal) والتي أطلق عليها لغات التفاعل بين المكونات(Components Interchange Language).

[55] قام في ورقته باقتراح طريقة لتعريف خاصية الوراثة(Inheritance) في لغة SOFA، موضحا أن هناك بعض اللغات توجد بها هذه الخاصية مثل C2 التي تسمح بتعريف نوع فرعى من أي وحدة من وحدات البرمجيات وكذلك (Darwin،Rapid،XADL،Wright،OMG CORBA).اهتمت دراسته بتفعيل هذه الخاصية في لغة SOFA فقط دون باقي اللغات، ولم تكن هناك رؤية واضحة لكيفية تطبيقها في بقية لغات توصيف معمارية البرمجيات.

في سبيل الوصول إلى تحويل بين (Hypermedia Authoring Languages) ولغات توصيف معمارية البرمجيات قام [56] باقتراح نموذج هيكلي(Structural_Meta_Modle) اخذا في الاعتبار بعض اللغات مقارنا بينهم في بعض الصفات الهيكلية، مثل بعض اللغات تحدد نقاط التعامل الخارجي للمكونات قد تكون نقطة واحدة أو عدة نقاط مثل (CL،C2،Darwin،SADL) كما أن هناك بعض اللغات تسمح بالتعامل مع اكثر من مكون خلال رابط واحد مثل (Wright،UniCon، C2،Aesop،Acme).كما وانه ركز فقط بال (Hypermedia Domain)

على ملى واخرين في [57] قاموا بتطوير لغة اكمى إلى +ACM التي تأخذ في الاعتبار الصفات غير الوظيفية، و تحليلها.

لم تأخذ الدراسة كل الصفات غير الوظيفية اخذت بعضها مثل زمن الاستجابة، احتمالات الفشل والإنتاجية.

3.5 تعلم معمارية البرمجيات

نموذج مجتمع المتعلمين يعتمد على نظرية أن التعلم هو عملية تحويل ومشاركة للمعرفة أكثر من تحويل وتبادل للمعرفة أو من اكتشاف واستحواذ على المعرفة.

ليس هناك تصميم صحيح للمعمارية أو بصورة أدق ليس هناك معمارية صحيحة أو خطأ وإنما يمكن أن تكون جيدة أو ضعيفة على حسب الاحتياجات المطلوب تحققها في المعمارية، لذلك نجد أن تدريس مقرر معمارية البرمجيات فيه بعض التعقيدات، قام [45] بتطبيق نظرية مجتمع المتعلمين في تدريس مقرر معمارية البرمجيات أستاذ وطالب في نفس الوقت وبذلك تتوزع الخبرة والمعرفة بينهم فلا يصبح الأستاذ هو المصدر الوحيد للمعرفة في المحاضرة. ونتيجة لذلك زادت نسبة الحضور من 50% إلى 90% لما يجده الطلاب من متعة في المشاركة. غير أن الدراسة لم توضح تلك النسب بصورة واضحة كما لم تأخذ نسبة التحصيل في الاعتبار.

وقام أيضا [46] باستخدام منهج المشاركة في تدريس مقرر معمارية البرمجيات حيث أوضح أن هناك العديد من التحديات التي تواجه الدارسين منها المستوى العالي للتجريد الذى يتسبب في صعوبة استيعاب المادة، كما وأن الدارسين يجب أن يتحلوا بمهارة المخاطبة والتواصل بين المشاركين، لذلك كانت فكرة منهج المشاركة حتى يتمكن الدراسين من التحاور والتشارك في الأفكار والمعرفة. اعتمدت الدراسة على تطبيق الجزء النظري من المقرر عمليا من خلال نظام حقيقي، ولكن ليس هناك أدوات اعتمد فيها على شرح المفاهيم الأساسية لمادة معمارية البرمجيات.

الكتابة أداة لتعلم معمارية البرمجيات، اعتمدت في [47] كطريقة لتدرس مقرر معمارية البرمجيات، حيث يقوم كل طالب بالكتابة في موضوع المحاضرة بعد المقدمة التي يلقيها عليهم المحاضر، ومن ثم تتم المناقشة في صورة مجموعات وبعدها تتم مناقشة جماعية، يستفيد الطالب من هذه الطريقة في استخدام الكتابة للتفكير والكتابة

للإخبار والتوضيح، مما يجعل الطلاب يستفيدون اكثر حيث إنه المناقشة والحوار بينهم من جهة وبينهم وبين الأستاذ من جهة أخرى توضح كثير من المفاهيم ويصبح الجميع مشاركين ومتحدثين.

هناك من استخدم لغة وصف المعمارية [49] ولكن لتعليم معمارية الحاسوب. وهناك من استخدم أدوات التصميم [48] كمادة مساعدة في تعلم معمارية البرمجيات.

دراسة أخرى لتعلم معمارية البرمجيات ولكن ليس لمستوى الماجستير وليس بتفعيل المشاركة وإنما باستخدام أداة يطلق عليها اسمHUSACCT [50] لتدريس معمارية البرمجيات لمستوى البكالوريوس على الرغم من المستوى العالي للتجريد في هذه المادة ولكن باستخدام هذه الأداة يتم تغطية المفاهيم الأساسية للمادة، في حين اعتبر [58] أن التجريد من المفاهيم الرئيسية التي يجب أن تدرس لطلاب علوم الحاسوب وهندسة البرمجيات في سنوات الدراسة الأولى، لما له من دور كبير في تغير التفكير على مستوى التفاصيل والتطبيق إلى التفكير في المفاهيم الأساسية.

6. المراجع

[1] S. Björnander, "Architecture Description Languages," Mrtc.Mdh. Se, 2011.

[2] P. Kogut and P. Clements, "Features of architecture description languages," Proc. Eighth Int. ..., no. April, pp. 1–13, 1994.

[3] D. Garlan and M. Shaw, "An Introduction to Software Architecture," Knowl. Creat. Diffus. Util., vol. 1, no. January, pp. 1–40, 1994.

[4] M. Ozkaya and C. Kloukinas, "Are We There Yet ? Analysing Architecture Description Languages for Formal Analysis, Usability, and Realisability."

[5] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What Industry needs from Architectural Languages : A Survey," pp. 1–25, 2012.

[6] B. Kitchenham, "Procedures for performing systematic reviews," Keele, UK, Keele Univ., vol. 33, no. TR/SE-0401, p. 28, 2004.

[7] S. Hussain, "Investigating Architecture Description Languages (ADLs): A Systematic Literature Review," 2013.

[8] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The Architecture Analysis & Design Language (AADL): An Introduction," no. February, 2006.

[9] H. Mei, F. Chen, Q. Wang, and Y. Feng, "ABC / ADL : An ADL Supporting Component Composition," pp. 38–47, 2002.

[10] D. Garlan and R. T. Monroe, "ACME : An Architecture Description Interchange Language Acme : An Architecture Description Interchange Language," 1997.

[11] E. M. Dashofy, A. van der Hoek, and R. N. Taylor, "An infrastructure for the rapid development of XML-based architecture description languages," in Proceedings of the 24th international conference on Software engineering - ICSE '02, 2002, p. 266.

[12] N. Medvidovic and R. N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," Softw. Eng. IEEE Trans., vol. 26, no. 1, pp. 70–93, 2000.

[13] J. Wen, S. Ying, L. Zhang, and Y. Ni, "AC2-ADL: Architectural description of aspect-oriented systems," Proc. 2008 Adv. Softw. Eng. Its Appl. ASEA 2008, vol. 3, no. 1, pp. 147–152, 2008.

[14] K. Dunsire, T. O'Neill, M. Denford, and J. Leaney, "The ABACUS architectural approach to computer-based system and enterprise evolution," 12th IEEE Int. Conf. Work. Eng. Comput. Syst., pp. 62–69, 2005.

[15] D. Batory, L. Coglianese, S. Shafer, and W. Tracz, "The ADAGE avionics reference architecture," 10th Comput. Aerosp. Conf., 1995.

[16] R. Bashroush, T. J. Brown, I. Spence, and P. Kilpatrick, "ADLARS: An architecture description language for software product lines," 2005 29th Annu. IEEE/NASA Softw. Eng. Work. SEW'05, vol. 2005, pp. 163–172, 2005.

[17] A. Bouzoualegh, D. Marcadet, F. Boulanger, and C. Jacquet, "An Architecture Description Language for Verification in Component-Based Software," Comput. Softw. Appl. 2008. COMPSAC '08. 32nd Annu. IEEE Int., pp. 365–368, 2008.

[18] R. Bruni, A. Lluch-Lafuente, U. Montanari, and E. Tuosto, "Architectural design rewriting as an architecture description language," 2008.

[19] S. Balsamo, M. Bernardo, and M. Simeoni, "Combining stochastic process algebras and queueing networks for software architecture analysis," Proc. Int. Work. Softw. Perform., pp. 190–202, 2002.

[20] R. J. Allen, "A Formal Approach to Software Architecture," Architecture, vol. 28, no. May, pp. 1–7, 1997.

[21] N. Medvidovic and R. N. Taylor, "A Framework for Classifying and Comparing Architecture Description Languages," 1997.

[22] J. Elloy, F. Simonot-lion, B. P. N. C. France, and F. De Haye, "AN ARCHITECTURE DESCRIPTION LANGUAGE Architecture description," no. 1, 2002.

[23] R. Bashroush, I. Spence, P. Kilpatrick, T. J. Brown, W. Gilani, and M. Fritzsche, "ALI: An extensible Architecture Description Language for industrial applications," Proc. - Fifteenth IEEE Int. Conf. Work. Eng. Comput. Syst. ECBS 2008, pp. 297–304, 2008.

[24] N. Ali, I. Ramos, and C. Solís, "Ambient-PRISMA: Ambients in mobile aspect-oriented software architecture," J. Syst. Softw., vol. 83, no. 6, pp. 937–958, 2010.

[25] M. Pinto and L. Fuentes, "AO-ADL: An ADL for describing aspect-oriented architectures," Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 4765 LNCS, pp. 94–114, 2007.

[26] N. Ubayashi, J. Nomura, and T. Tamai, "Archface: A Contract Place Where Architectural Design and Code Meet Together," Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. - ICSE '10, vol. 1, p. 75, 2010.

[27] R. Monroe, "Capturing Software Architecture Design Expertise with Armani," Tepper Sch. Bus., 2000.

[28] G. Georg and S. Seidman, "Use of architecture description languages to describe a distributed measurement system," Proc. Int. Symp. Work. Eng. Comput. Based Syst., pp. 185–193, 2000.

[29] C. Chavez, A. Garcia, C. Lucena, and U. Kulesza, "Aspectual Connectors: Supporting the Seamless Integration of Aspects and ADLs," Proc. Brazilian Symp. Softw. Eng. SBES, pp. 17–32, 2006.

[30] S. Fürst, "Challenges in the Design of Automotive Software," Des. Autom. Test Eur. Conf. Exhib. (DATE), 2010, pp. 6–8, 2010.

[31] M. Pinto, L. Fuentes, and J.-M. Troya, "DAOP-ADL : An Architecture Description Language for Dynamic Component and Aspect-Based Development," 2nd Int. Conf. Gener. Program. Compon. Eng. (GPCE '03), pp. 118–137, 2003.

[32] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer, "Specifying distributed software architectures," Esec, vol. 989, no. September 1995, pp. 137–153, 1995.

[33] N. Medvidovic, "A Classification and Comparison Framework for Software Architecture Description Languages," 1996.

[34] P. Poizat, "A Formal Architectural Description Language based on," vol. 12, no. 12, pp. 1741–1782, 2006.

[35] S. Zhang and S. Goddard, "3CoFramework: A Component-based Framework for Distributed Applications," Proc. Int. Conf. Softw. Eng. Res. Pract., vol. 1, no. June, pp. 398–404, 2003.

[36] A. Grau, B. Shihada, and M. Soliman, "Architectural Description Languages and their Role in Component Based Design," Architecture, no. August, pp. 1–27, 2002.

[37] E. M. Dashofy and R. N. Taylor, "A Highly-Extensible, XML-Based Architecture Description Language."

[38] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The Koala component model for consumer electronics software," Computer (Long. Beach. Calif)., vol. 33, no. 3, pp. 78–85, 2000.

[39] T. Overview, "Technical and Historical Overview of MetaH," 2000.

[40] J. H. McDuffie, "Using the architecture description language MetaH for designing and prototyping an embedded reconfigurable sliding mode flight controller," Digit. Avion. Syst. Conf. 2002. Proceedings. 21st, vol. 2, p. 8B1-1-8B1-17 vol.2, 2002.

[41] D. C. Luckham, "Rapide: a language and toolset for simulation of distributed systems by partial orderings of events," POMIV '96 Proc. DIMACS Work. Partial order methods Verif., pp. 329–357, 1997.

[42] S. Vestal and H. Systems, "A Cursory Overview and Comparison of Four Architecture Description Languages," Communication, 1993.

[43] M. Voelter, "A Family of Languages for Architecture Description."

[44] F. Oquendo, "π-ADL: an Architecture Description Language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures," ACM SIGSOFT Softw. Eng. Notes, vol. 29, no. 3, pp. 1–14, 2004.

[45] R. C. D. Boer, R. Farenhorst, and H. Van Vliet, "A community of learners approach to software architecture education," Proc. - 22nd Conf. Softw. Eng. Educ. Training, CSEET 2009, pp. 190–197, 2009.

[46] A. Van Deursen et al., A Collaborative Approach to Teaching Software Architecture. 2017.

[47] A. I. Wang and C.-F. Sorensen, "Writing as a Tool for Learning Software Engineering," 19th Conf. Softw. Eng. Educ. Train., no. May 2014, pp. 35–42, 2006.

[48] G. Engels, J. H. Hausmann, M. Lohmann, and S. Sauer, "Teaching UML Is teaching software engineering is teaching abstraction," Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 3844 LNCS, pp. 306–319, 2006.

[49] I. Talk et al., "Workshop on Computer Architecture Education," Forum Am. Bar Assoc., pp. 00–08, 2004.

[50] C. Köppe and L. Pruijt, "Teaching Software Architecture Concepts with HUSACCT - Tool Demo," Present. SPLASH-E Conf. SPLASH-E'15, no. October, 2015.

[51] H. Koppelman and B. van Dijk, "Teaching abstraction in introductory courses," Proc. Fifteenth Annu. Conf. Innov. Technol. Comput. Sci. Educ. - ITiCSE '10, p. 174, 2010.

[52] I. Malavolta, "Architectural languages." [Online]. Available: http://www.di.univaq.it/malavolta/al/#languages.

[53] M. Shaw and P. Clements, "How Should Patterns Influence Architecture Description Languages?," Citeseer, 1996.

[54] D. Thesis, "Modelling and Formal Analysis of Component-Based Systems in View of Component Interaction," no. May, 2008.

[55] R. Hilliard and T. Rice, "Expressiveness in Architecture Description Languages," Burns.

[56] D. C. Muchaluat-Saade and L. F. G. Soares, "Towards the convergence between hypermedia authoring languages and architecture description languages," Proc. 2001 ACM Symp. Doc. Eng. - DocEng '01, p. 48, 2001.

[57] I. Derbel, L. L. Jilani, and A. Mili, "ACME+ for software architecture analysis," ICSOFT 2013 - Proc. 8th Int. Jt. Conf. Softw. Technol., pp. 429–437, 2013.

[58] H. Koppelman and B. van Dijk, "Teaching abstraction in introductory courses," Proc. fifteenth Annu. Conf. Innov. Technol. Comput. Sci. Educ. - ITiCSE '10, p. 174, 2010.