MONOGRÀFIC
MÈTODE SCIENCE STUDIES JOURNAL (2020). Universitat de València.
DOI: 10.7203/metode.11.16112
Enviat: 18/11/2019. Acceptat: 10/03/2020.

EL LLARG I SINUÓS CAMÍ

Accidents i bricolatge en l'estandardització del programari

SERGI VALVERDE

El programari (*software*) es basa en principis universals, però no així el seu desenvolupament. Relacionar el programari amb el maquinari (*hardware*) no és mai automàtic ni senzill. Els intents d'optimitzar la producció de programari i reduir-ne el cost (com ocorre amb el maquinari) han estat molt limitats. L'èxit d'un projecte sol dependre de les accions d'individus altament qualificats i experimentats. El llarg i complicat camí cap a un programari fiable i útil sol estar farcit d'accidents idiosincràtics i d'una complexitat emergent. S'esperava que l'estandardització del programari eliminara aquestes fonts no desitjades de diversitat per tal d'aconseguir processos de desenvolupament controlables, llenguatges de programació universals i components reutilitzables. No obstant això, l'adopció limitada d'estàndards de desenvolupament suggereix que encara no comprenem les raons per les quals és tan difícil produir programari. L'estandardització de programari s'ha vist obstaculitzada per la nostra limitada comprensió del paper dels humans en l'origen de la diversitat tecnològica.

Paraules clau: estàndards de programari, desenvolupament de programari, llenguatge de programació, complexitat, evolució tecnològica.

Imaginem que, en escriure una carta d'amor, ens veiérem obligats a escriure oracions amb un nombre fix de caràcters. Oblidem la idea d'utilitzar una prosa complexa o de citar Shakespeare en la nostra obra mestra: si ens passem del límit, la frase es tallarà, haguera o no acabat. Això és el que vam viure tots i cadascun dels estudiants de programació informàtica en les dècades dels vuitanta i noranta del segle passat. En aquell moment era necessari trencar les cadenes de més de vuitanta caràcters en trossos més menuts per a ajustar-se a les limitacions dels editors de text. Això era encara més evident en escriure càlculs complexos en un prestigiós llenguatge de programació anomenat Fortran. A vegades, una equació no es podia escriure en una sola línia, i era necessari partir aquella llarga expressió matemàtica en múltiples trossos. La inserció de molestos salts de línia que interrompien els nostres pensaments semblava una complicació injustificada per a una tasca –la programació informàtica– que ja era (i continua sent) intrínsecament complexa. Per què 80 i no 166 o qualsevol altre nombre de caràcters?

«La tendència exponencial del maquinari no s'ha vist reflectida en innovacions paral·leles en programari»

L'origen d'aquesta qüestió és anterior a les pantalles de grandària fixa; radica en la grandària de les targetes perforades amb les quals es processava el cens dels EUA en la dècada de 1890. Aquestes targetes amb vuitanta caràcters per línia s'introduïen en les primeres computadores comercials d'IBM, en els anys cinquanta del segle xx. Els nostres ordinadors personals van heretar el format de columna de vuitanta caràcters, que es va convertir en l'estàndard *de facto* per a molts de nosaltres. Fins i tot avui dia, en un moment en el qual les pantalles d'alta resolució són tan comunes, els editors de text continuen sent compatibles amb relíquies de maquinari (*hardware*) que ja no tornarem a utilitzar.

Com hem vist en l'exemple anterior, una casualitat històrica pot deixar una petjada ben fonda en l'evolució tecnològica (Arthur, 1994). Algunes poden ser inofensives, però unes altres provoquen ineficiències associades a tasques subòptimes. Com podem garantir la idoneïtat d'una elecció tecnològica? Una manera de fer-ho és comprovar la llista de bones pràctiques i recomanacions. Un estàndard tecnològic elimina els elements innecessaris de les

pràctiques de l'enginyeria i conserva «el que val la pena». Aquests estàndards han estalviat molt d'esforç assegurant-se que els «materials, productes, processos i serveis siguen els adequats per al seu propòsit», tal com predica l'Organització Internacional per a l'Estandardització (ISO). Podem trobar molts exemples d'estàndards útils en la indústria i en enginyeria, on les comunitats d'experts defineixen i actualitzen les seues recomanacions perquè siguen consistents. En essència, la qualitat d'aquests estàndards depèn de la capacitat dels experts per a decidir si les normes i innovacions associades continuen sent útils o no. L'estandardització augmenta l'eficiència tecnològica, però també pot prolongar més del compte la vida de les tecnologies existents inhibint qualsevol inversió en innovació (Tassey, 1999). Aconseguir un equilibri òptim entre l'eficiència i la innovació és extremadament difícil, i les prediccions sobre innovació tecnològica han estat fins ara poc fiables. En particular, les innovacions complexes s'enfronten a més obstacles que les simples a l'hora d'aconseguir l'èxit comercial (vegeu Figura 1) (Schnaars i Wymbs, 2004).

Figura 1. Des de l'aparició del telèfon, els seus inventors van pronosticar l'arribada d'interaccions visuals a llarga distància. En 1924 Alexander Graham Bell va dir que arribaria el dia en què la persona al telèfon podria veure la persona amb la qual estava parlant. No obstant això, l'intent per part d'AT&T de comercialitzar el *picturephone* en els anys seixanta (en la imatge) va ser un fracàs comercial. AT&T va invertir tant en aquesta tecnologia que, si l'èxit comercial depenguera únicament del convenciment, el videòfon s'hauria convertit en un aparell tan comú com el telèfon fa molts anys.

EL COLL DE BOTELLA DEL PROGRAMARI

En l'últim segle hem presenciat una acceleració espectacular en el rendiment computacional, la capacitat d'emmagatzematge digital i les comunicacions electròniques a escala global. La famosa llei de Moore ha marcat l'evolució de les tecnologies de la informació. Això és conseqüència d'uns principis teòrics sòlids: els fonaments conceptuals de la computació continuen essent els mateixos des de la publicació dels textos clàssics d'Alan Turing. D'altra banda, la tendència exponencial del maquinari no s'ha vist reflectida amb innovacions paral·leles en programari (*software*), que es continuen mesurant en escales de temps humanes. Encara que tant maquinari com programari han crescut en complexitat, la seua evolució presenta una asimetria important (Valverde, 2016). L'actual coll de botella en tecnologies de la informació no és el cost del maquinari, sinó l'obtenció del programari necessari per a fer-lo funcionar (Ensmenger, 2010). El programari dicta la utilitat de les tecnologies de la informació i la seua demanda ha creat un enorme problema econòmic. Molts projectes de programari estan farcits d'errors, accidents i decisions idiosincràtiques (Brooks, 1975). El fracàs recurrent de projectes va portar el sector a definir enfocaments fiables per assegurar el desenvolupament de programari d'alta qualitat (Charette, 2005).

«Molts projectes de programari estan farcits d'errors, accidents i decisions idiosincràtiques»

Des de la invenció de la tecnologia de computació en la dècada de 1950, l'estandardització ha estat l'aspiració de moltes associacions d'usuaris i professionals de la informàtica. Com en tota disciplina emergent, els professionals estaven ansiosos per demostrar la seua utilitat social. En particular, els primers passos dels programadors en l'estandardització es van centrar en la interoperabilitat del programari informàtic; és a dir, en la possibilitat d'intercanviar i reutilitzar programari entre diferents ordinadors. Per exemple, l'estàndard de sistema operatiu MS-DOS va donar lloc a l'adopció generalitzada dels ordinadors personals (PC), que al seu torn van conduir a moltes innovacions posteriors, com l'aparició d'Internet i la World Wide Web. El creixement exponencial del mercat dels ordinadors i la proliferació d'ordinadors incompatibles va augmentar la competència en el món del programari. S'esperava que l'estandardització facilitara l'aparició de programari no sols compatible sinó també més econòmic. Per a això, els estàndards es van centrar princi-

palment en dos aspectes: escriure i mantenir codi (o «desenvolupament de programari») i les eines per a fer costat a aquest procés (per exemple, els llenguatges de programació i els jocs de components de programari reutilitzable). Però si bé la interoperabilitat dels programes informàtics va tenir un gran èxit, l'adopció tan escassa d'estàndards en desenvolupament de programari suggereix la presència de límits que encara no comprenem prou.

DESENVOLUPAMENT DE PROGRAMARI IMPREDICTIBLE

Les iniciatives del Departament de Defensa dels Estats Units són un bon exemple dels obstacles a què s'enfrontava el desenvolupament d'estàndards de programari. Des de la dècada dels setanta fins als vuitanta, el Departament va tractar de fer que les seues contractes compliren uns estàndards en aquest sentit (McDonald, 2010). El seu objectiu era reduir els costos tan elevats del desenvolupament de programari. Darrere d'això (i d'altres iniciatives paral·leles) hi havia l'aspiració (mai realitzada) de reemplaçar el component humà amb un procés de desenvolupament de programari completament automatitzat i a prova d'errors. En 1978 el Departament de Defensa va publicar una sèrie de normes de desenvolupament que totes les contractes havien de seguir. En primer lloc, el programari s'hauria de desenvolupar seguint un procés de disseny descendent, des de la definició global del sistema als seus components funcionals (vegeu Figura 2). D'altra banda, per a millorar la llegibilitat del codi de programació (reduint en conseqüència les possibilitats d'error), hi havia una grandària màxima per als components individuals de programari i es van prohibir instruccions «perjudicials» (com l'ordre «GO TO», que trenca la seqüència lògica de les operacions de programari). I, per últim, tot el codi programat hauria d'escriure's utilitzant una sèrie aprovada de llenguatges de programació d'alt nivell. Sorprenentment, el Departament es va enfrontar a una dura oposició quan va intentar que les contractes incorporaren aquestes normes. Encara que l'estàndard reflectia les convencions sobre com escriure bon codi, molts programadors pensaven que era innecessari i que havia quedat obsolet, per la qual cosa suposava una càrrega innecessària que limitava la seua llibertat. A causa de les crítiques creixents i de la pressió social, en els anys noranta es va abandonar qualsevol intent d'imposar estàndards de programari per contracte.

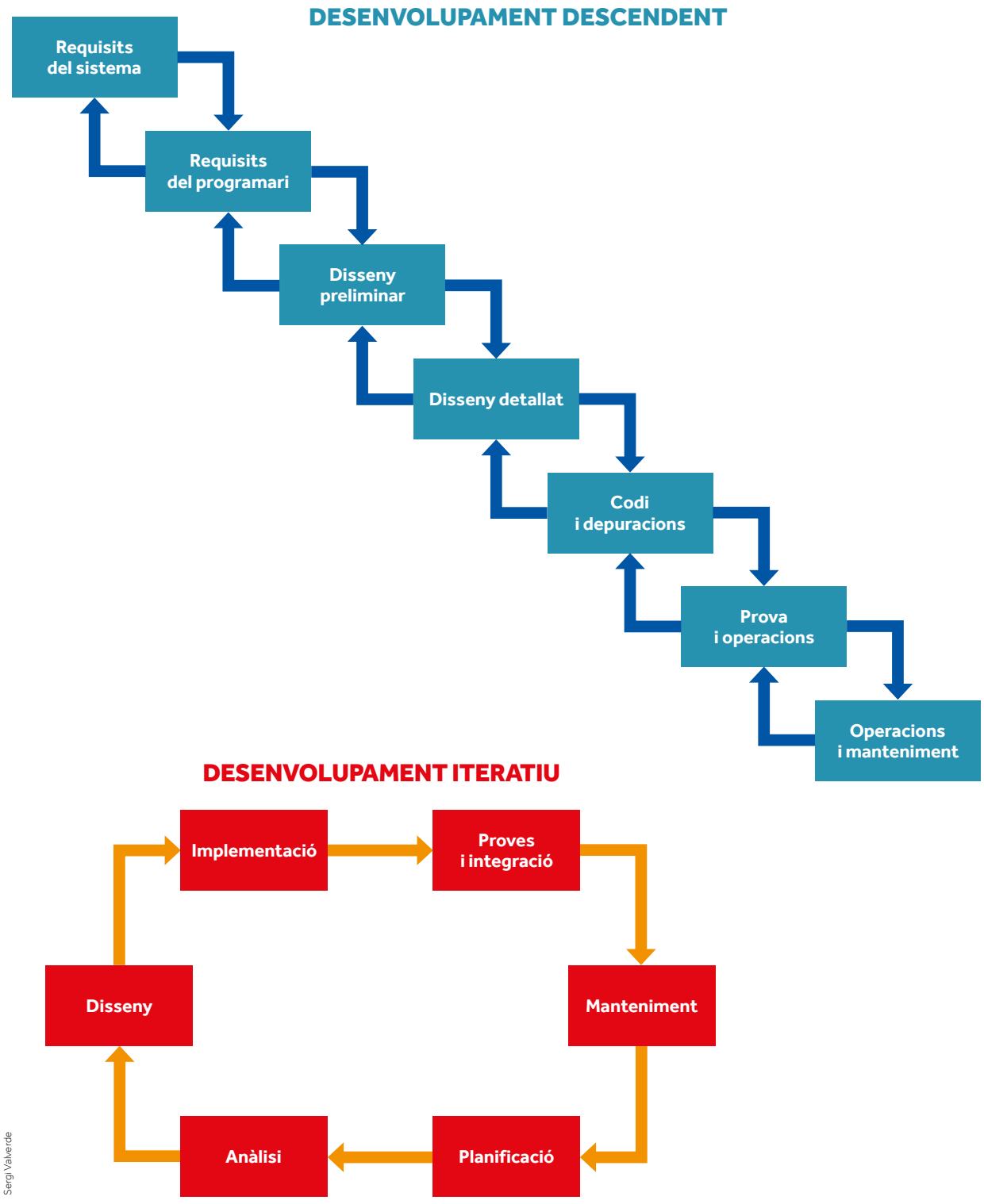
Un obstacle fonamental era la suposició poc realista per part del model descendent segons la qual es pot planificar la trajectòria dels projectes de programari. El codi real (com molts projectes més d'enginyeria complexos) sol implicar correccions costoses en etapes posteriors del projecte; és a dir, algunes de les eleccions inicials de disseny es converteixen en accidents històrics (McDonald, 2010). L'experiència amb els projectes de programari suggereix que resulta ben difícil complir els requisits funcionals; això és, determinar el conjunt de tasques que el programa ha de realitzar. En particular, qualsevol requisit no especificat en el disseny inicial es tradueix en modificacions costoses en etapes posteriors del projecte (Boehm, 1976). Per exemple, els usuaris tenen intuïcions sobre com haurien de funcionar els sistemes operatius (com Microsoft Windows), però els costa molt més descriure funcions de programari que encara no han utilitzat.

Un enfocament més pragmàtic conceptualitza el programari com un procés incremental i iteratiu, no gaire diferent de l'evolució natural. Amb això s'espera minimitzar la quantitat de decisions de disseny redundants. Per exemple, en el model basat en prototips, els usuaris i programadors cooperen activament construint un sistema de programari gran i estable. Ací, els canvis dels programadors són una font de variació natural en el projecte. Els usuaris actuen com l'entorn del prototip de programari seleccionant característiques segons les seues necessitats. Repetint aquest procés iteratiu, els usuaris i els programadors coevolucionen en un sistema que s'ajusta a les especificacions. El prototipat de programari accepta que, en un entorn canviant, l'adaptabilitat ràpida i bruta és preferible a una planificació descendent inadequada.

«Un enfocament més pragmàtic conceptualitza el programari com un procés incremental i iteratiu, no gaire diferent de l'evolució natural»

UNA TORRE DE BABEL ELECTRÒNICA

Les eines utilitzades en el desenvolupament de programari tampoc han aconseguit la uniformitat d'altres disciplines tecnològiques com l'enginyeria elèctrica. La diversitat tecnològica es troba en els repositoris públics de programari de codi obert, on no existeix un marc únic de desenvolupament, sinó una multitud. Hi podem trobar moltes maneres de resoldre el mateix problema en un programa per a diferents plataformes (per exemple, Windows, Mac OSX o Linux) escrit en llenguatges de programació incompatibles (com a C++, Python o Java) utilitzant una mescla de llibreries registrades de programari (OpenGL o DirectX). Aquestes solucions



Sergi Valverde

Figura 2. Els estàndards divideixen el desenvolupament de programari en diferents etapes, com el disseny, la construcció, les proves i el manteniment. En els anys seixanta, el Departament de Defensa dels Estats Units va intentar imposar en les seues contractes un model seqüencial (o descendent) de desenvolupament de programari, amb èxit escàs. El desenvolupament iteratiu de programari és més flexible i pot reduir els malentesos entre els usuaris i els programadors.

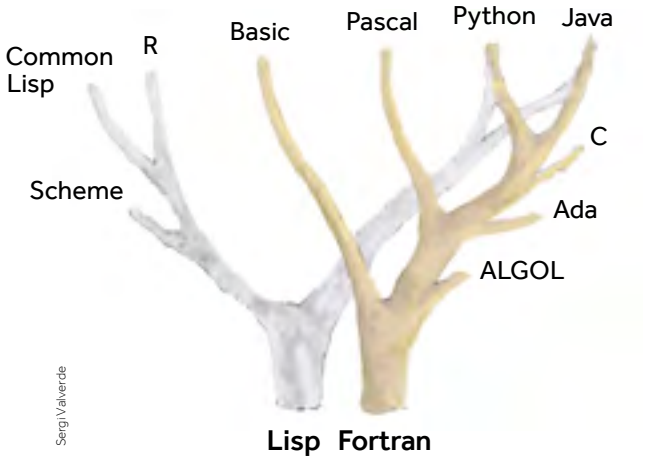


Figura 3. Aquest diagrama il·lustra l'evolució dels llenguatges de programació en forma d'arbre. L'objectiu dels primers llenguatges en els seixanta era principalment la indústria i els negocis. Els llenguatges d'aquella època, com ALGOL, van ser dissenyats per comitès privats d'experts. No obstant això, l'adopció general de les tecnologies de la informació va accelerar la diversificació dels llenguatges de programació. Alguns dels més populars, com ara C o Python, han estat desenvolupats per comunitats distribuïdes de programadors. Diferents branques de l'arbre influeixen en l'evolució contínua dels llenguatges de programació.

es basen en les mateixes idees i conceptes, però les seues tecnologies de base són incompatibles, la qual cosa en limita la reutilització. En aquest context, integrar programari amb èxit continua depenent de la bona voluntat i la col·laboració desinteressada entre programadors. La situació dels llenguatges de programació és particularment reveladora. En 1936, Alan Turing va demostrar que no hi havia barreres teòriques per a la unificació dels llenguatges de programació. És a dir, que existeix una computadora universal capaç de fer qualsevol tasca. No obstant això, la realitat és que tenim milers de llenguatges de programació diferents a la nostra disposició (vegeu Figura 3). Per què no podem simplement crear un llenguatge universal amb moltes funcionalitats?

Hi ha hagut múltiples intents d'estandarditzar els llenguatges de programació, però no n'hi ha hagut cap que s'haja acceptat universalment. Per exemple, l'objectiu del llenguatge de programació Ada era reemplaçar la miríada de llenguatges utilitzats en els projectes de programari del Departament de Defensa (en la dècada dels setanta, s'havien utilitzat més de 450 llenguatges diferents en projectes de l'exèrcit). A diferència de molts llenguatges dissenyats per consorcis privats (com COBOL), la creació d'Ada va ser el resultat d'un concurs internacional subjecte a revisió externa (a càrrec d'experts acadèmics en llenguatges de programació). Un dels objectius d'Ada era evitar els errors humans en el desenvolupament de programari, la qual cosa requereix ser estrictes amb la seguretat i la concurrència d'una manera que rares vegades es dona en altres llenguatges. Malgrat aquests avantatges, Ada no va aconseguir mai la popularitat d'altres llenguatges menys robustos com ara C++, que va aparèixer en 1985. Uns vint anys després, ISO va estandarditzar per primera vegada el llenguatge C++ (que es va convertir en un estàndard *de facto*). Això mostra, de nou, que l'èxit no es pot predir, sense importar l'esforç realitzat en el disseny inicial. Molts factors influeixen en l'èxit dels llenguatges de programació, incloent-hi la popularitat, la complexitat i l'economia.

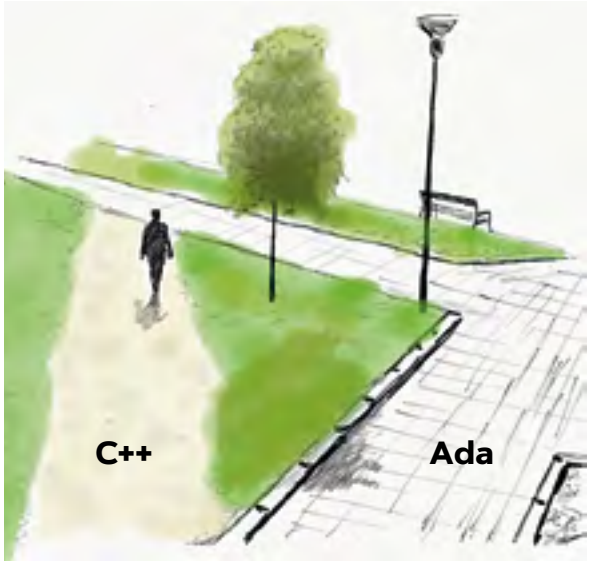
Des d'una perspectiva històrica, sembla que el camí des del llenguatge C fins a C++ va ser més fàcil que l'adopció d'un estàndard d'alta qualitat (però relativament desconegut) (Figura 4). El compromís amb l'opció «menor» no es va traduir en un bloqueig del mercat ni en la falta d'innovació perquè els llenguatges de programació evolucionen contínuament i s'influeixen mútuament. En algun moment la gran comunitat d'usuaris de C++ es va beneficiar de les innovacions proposades per Ada sense perdre la compatibilitat amb la tecnologia existent. Sembla que els programadors prefereixen viure amb un programari imperfecte que reconstruir-ho tot des de zero.

COMPLEXITAT EMERGENT

En els processos complexos d'enginyeria, com aquells que tenen a veure amb la programació, els enginyers no sempre poden decidir el millor curs d'acció. Més aviat els «impulsa» la complexitat emergent de les seues creacions. Ni l'evolució de la tecnologia ni la de la biologia poden evitar els accidents i el bricolatge quan la complexitat augmenta massa.

«Les eines utilitzades en el desenvolupament de programari tampoc han aconseguit la uniformitat d'altres disciplines tecnològiques com l'enginyeria elèctrica»

A començament dels noranta, l'heterogènia diversitat de maquinari, sistemes operatius i llenguatges de programació representava un obstacle per a la interoperabilitat del programari. Els enginyers i els gestors van veure en la reutilització de components la solució natural a aquest problema. En lloc de construir un programa des de zero, es podria utilitzar un repositori de blocs (o components) que incloguera les funcions de programari més comunes. Aquest enfocament requereix



Sergi Valverde

Figura 4. El camí cap a l'adopció de tecnologies depèn de molts factors, incloent-hi el cost. I el fet que alguns llenguatges de programació estiguen dissenyats acuradament per comissions d'experts no significa automàticament que acaben adoptant-se de manera generalitzada. El llenguatge estàndard Ada (dreta) gaudeix d'un reconeixement com a llenguatge de qualitat (amb moltes característiques úniques que no solen estar presents en altres llenguatges de programació). No obstant això, la popularitat del sistema operatiu Unix va catalitzar l'adopció del llenguatge C, així com el seu successor orientat a objectes, C++ (esquerra). Aquests dos llenguatges de programació –tots dos estandarditzats mitjançant normes ISO– han acabat dominant el mercat, mentre que Ada s'ha mantingut com una solució especialitzada.

definir un estàndard d'interoperabilitat de programari com CORBA (*Common Object Request Broker Architecture*; en català, "Arquitectura Intermediària de Petició de Components Comuns"). En CORBA, els components de programari escrit en diferents llenguatges, com Java i C, també poden intercanviar informació adherint-se a un protocol de programari comú. Es va definir aquest estàndard com «la següent generació de tecnologia per al comerç electrònic», i va guanyar molta popularitat al començament. No obstant això, les deficiències tècniques i els errors de disseny van dificultar que es consolidara en el mercat, i finalment CORBA va quedar desplaçat per tecnologies web com XML (Henning, 2008).

El fracàs de CORBA s'ha associat principalment a qüestions de qualitat. En un examen més profund, exemplifica la dificultat de la tasca de definir una interfície estàndard entre components de programari. El desenvolupament de programari basat en components és molt similar a la idea de les peces de Lego. Els blocs de Lego són interoperables gràcies a un sistema d'assemblatge patentat (Figura 5). És a dir, podem connectar qualsevol parell de blocs, independentment de la forma, color o funció que tinguen. Però això no ocorre amb la programació. Hi ha molts exemples catastròfics d'interaccions no desitjades entre components de programari. Per a evitar-ho, ens hem vist obligats a provar (i depurar) qualsevol interacció, la qual cosa requereix molt de temps i esforç. És inevitable i imposa un límit estricte al desenvolupament escalable de programari.

EVOLUCIÓ EN EL FUTUR

En un espai de temps molt curt, hem passat d'una visió de ciència-ficció amb computadores totpoderoses a un producte bàsic que és a les mans de tot el món. Aquesta adopció tan àmplia de la tecnologia de la informació va convertir el programari en un component clau de la nostra societat. Tenim una necessitat urgent d'aconseguir formes més fiables i econòmiques de desenvolupar programari a un ritme més ràpid. S'ha proposat l'estandardització del programari com a solució a aquest problema, assimilant-la als milers d'estàndards bàsics necessaris per al funcionament de la nostra societat.

Podem definir normes universals per a controlar el desenvolupament de programari? La història recent demostra que el desenvolupament de programari fiable continua sent un objectiu difícil d'aconseguir. Un dels principals problemes és la incertesa present en cada etapa del procés de desenvolupament de programari. A l'hora de dissenyar un sistema de programari, hi ha moltes opcions diferents. Decidir quina és la millor en cada etapa no és en absolut fàcil. No podem estar segurs de les funcionalitats de la tecnologia a llarg termini perquè l'èxit es basa en canvis impredictibles de l'entorn. La incertesa afecta l'evolució de moltes tecnologies (Petroski, 1992), però és fins i tot més greu en el cas del programari a causa de l'absència d'un entorn físic. El programari no es descompon ni s'avaria com altres tecnologies. Conèixer els principis que subjauen als sistemes físics va permetre a l'enginyeria i a la indústria realitzar avanços espectaculars. Però la situació és molt diferent en aquest cas, ja que l'enfocament científic del desenvolupament de programari va molt per darrere en comparació amb la pràctica. És necessari que el desenvolupament de programari madure, i això depén de la disponibilitat dels resultats científics (Glass, 2009).

Els obstacles per a l'estandardització de programari suggereixen l'existència d'un problema recurrent: l'enginy humà no es pot reemplaçar per peces estandarditzades. De moment, el cervell és un component essencial per a traduir els requisits humans al llenguatge de programació. Encara no entenem del tot com ho fem els humans per a programar els ordinadors. Desenvolupar programes útils i fiables requereix enginy i una experiència considerable. Els programadors inexperts no poden confiar en la seua intuïció per a avaluar si un programari

Domini públic

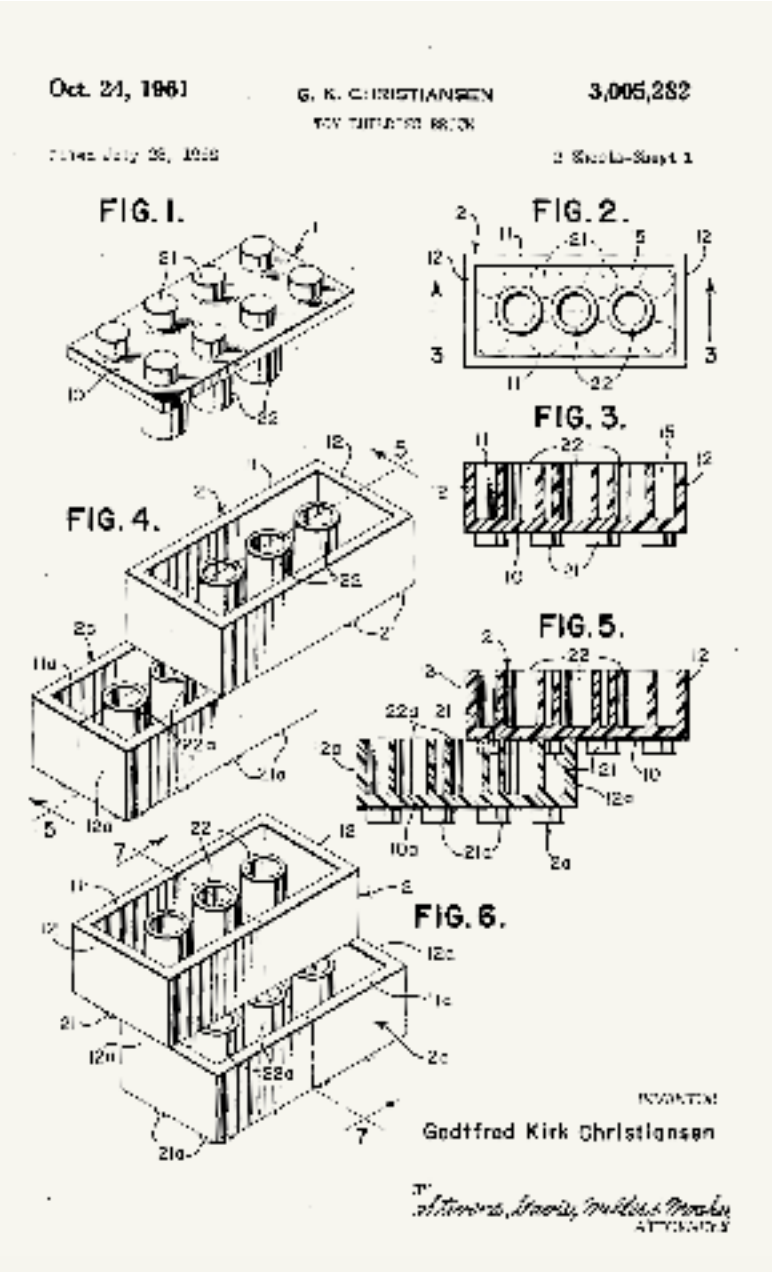


Figura 5. En 1961, la patent nord-americana 3005282A va proposar el disseny d'un joguet amb «blocs de construcció», coneguts popularment com *legos*. Aquest document descriu un enginyós mecanisme d'assemblatge que permet unir moltes estructures diferents. En el desenvolupament de programari no s'ha aconseguit mai crear una interfície universal com aquesta.

«Encara no entenem del tot com ho fem els humans per a programar els ordinadors»

és massa gran o massa petit, simple o complex, perquè és invisible. Només després de passar molt de temps programant (i sobretot depurant), podem començar a comprendre l'enorme complexitat del programari. Més enllà de la nostra intel·ligència i les nostres habilitats, només podem desenvolupar tecnologia complexa gràcies a tot el coneixement generat per la nostra societat amb el pas dels anys (Basalla, 1988; Messoudi, 2011). L'evolució de tecnologies complexes com el llenguatge de programació C++ ha estat el resultat de la informació acumulada per molta gent al llarg de molts anys en comunitats de codi obert. Hi ha qui defensa la idea que el desenvolupament de programari prompte quedarà obsolet, i que la intel·ligència artificial reemplaçarà comunitats senceres de programadors humans. Això sembla poc probable sense una comprensió completa de la manera com els humans programen les computadores. En qualsevol cas, una cosa sembla certa: el programari del futur no es dissenyarà, sinó que evolucionarà.

REFERÈNCIES

Arthur, W. B. (1994). *Increasing returns and path dependence in the economy*. Ann Arbor: Michigan University Press. doi: 10.3998/mpub.10029

Basalla, G. (1988). *The evolution of technology*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9781107049864

Boehm, B. W. (1976). Software engineering. *IEEE Transactions on Computers*, *25*(12), 1226–1241. doi: 10.1109/TC.1976.1674590

Brooks, F. (1975). *The mythical man-month: Essays on software engineering*. Boston: Addison-Wesley.

Charette, R. N. (2005, 2 de setembre). Why software fails. *IEEE Spectrum*. Consultat en https://spectrum.ieee.org/computing/software/why-software-fails

Ensmenger, N. L. (2010). *The computer boys take over. Computers, programmers, and the politics of technical expertise*. Cambridge: The MIT Press.

Glass, R. L. (2009). Doubt and software standards. *IEEE Software*, *26*(5), 104. doi: 10.1109/MS.2009.126

Henning, M. (2008). The rise and fall of CORBA. *Communications of the ACM*, *51*(8), 52–57. doi: 10.1145/1378704.1378718

McDonald, C. (2010). From art form to engineering discipline? A history of US military software development standards, 1974-1998. *IEEE Annals of the History of Computing*, *32*(4), 32–47. doi: 10.1109/MAHC.2009.58

Messoudi, A. (2011). *Cultural evolution: How Darwinian theory can explain human culture and synthesize the social sciences*. Chicago: University of Chicago Press.

Petroski, H. (1992). *To engineer is human: The role of failure in successful design*. Nova York: Vintage Books.

Schnaars, S., & Wymbs, C. (2004). On the persistence of lackluster demand: The history of the video telephone. *Technological Forecasting and Social Change*, *71*(3), 197–216. doi: 10.1016/S0040-1625(02)00410-9

Tassey, G. (1999). Standardization in technology-based markets. *Research Policy*, *29*(4-5), 587–602. doi: 10.1016/S0048-7333(99)00091-8

Valverde, S. (2016). Major transitions in information technology. *Philosophical Transactions of the Royal Society B*, *371*(1701), 20150450. doi: 10.1098/rstb.2015.0450

SERGI VALVERDE. Expert en sistemes complexos, doctorat en Física Aplicada i investigador de l'Institut de Biologia Evolutiva (UPF-CSIC) de Barcelona (Espanya), on lidera el Laboratori de l'Evolució Tecnològica (ETL). El seu grup de recerca és pioner en l'estudi de les grans transicions evolutives mitjançant la comparativa de sistemes biològics i artificials. La seua recerca multidisciplinària integra diverses àrees de coneixement, des de la teoria de xarxes fins a l'ecologia teòrica i la simulació computacional dels processos evolutius. És membre de la junta de la Xarxa Catalana per a l'Estudi dels Sistemes Complexos (complexitat.cat). ✉ sergi.valverde@ibe.upf-csic.es