ENGINYERIA DE LA QUALITAT DEL PROGRAMARI: ESTRATÈGIA, ACTIVITATS I EVOLUCIÓ

Albert Tort
Director tècnic (CTO) a Sogeti España, Capgemini, i coordinador del postgrau Software Quality Assurance de la Universitat Politècnica de Catalunya. albert.tort@sogeti.com

Resum: La qualitat del programari (*software*) inclou diversos atributs: des de la qualitat funcional fins a l'experiència d'usuari, passant per la seguretat, l'accessibilitat o el rendiment. Els defectes en el programari tenen conseqüències econòmiques en els negocis i també conseqüències socials. Per això és imprescindible que qualsevol procés de desenvolupament de programari prevegi una estratègia transversal que determini les activitats d'assegurament de la qualitat necessàries (anàlisi de codi, proves de programari, indicadors de qualitat, monitoratge, etc.) en funció dels riscos de cada aplicació. Aquest article té com a objectius: 1) presentar el valor social i de negoci de l'enginyeria de la qualitat; 2) descriure les principals activitats per a la detecció (com més anticipada millor) de defectes i la provisió de retroacció (*feedback*); 3) presentar l'enfocament de qualitat contínua en els processos de desenvolupament àgil, i 4) reflexionar sobre les perspectives i l'evolució de la qualitat del programari.

Paraules clau: qualitat del programari, proves de programari, experiència d'usuari, desenvolupament, rendiment, seguretat.

SOFTWARE QUALITY ENGINEERING: STRATEGY, ACTIVITIES AND EVOLUTION

Abstract: Software quality addresses a wide range of attributes: functional quality, user experience, security, accessibility, performance, etc. Software defects have business and social consequences. Therefore, software development projects need to define and implement a cross-cutting strategy in order to focus on activities (code analysis, testing, quality metrics, monitoring, etc.) aimed at assuring such quality attributes, depending on the risks of each application. This paper has the following goals: (1) to present the business and social value of software quality engineering; (2) to describe the main activities required to detect defects as early as possible and to provide feedback; (3) to present the continuous quality approach which should be embedded in agile software development, and (4) to reflect on the expectations and evolution of software quality.

Keywords: software quality, software testing, user experience, development, performance, security.

1. Introducció

Els principis de desenvolupament àgil del programari (Beck, 2001) promouen una entrega iterativa de programari que aporti valor de manera progressiva i afavoreixi la retroacció continuada. Dit d'una altra manera, l'objectiu no és només assolir entregues de programari més ràpides, sinó que aquestes aportin valor social i de negoci de manera incremental. En aquest context, els enfocaments seqüencials de desenvolupament d'aplicacions han evolucionat cap a una entrega freqüent de versions incrementals que, al seu torn, requereixen retroacció continuada sobre la seva qualitat per tal de guiar-ne el desenvolupament. Aquesta retroalimentació inclou totes aquelles activitats que acaben capturant informació sobre la qualitat del programari (enginyeria de requisits, proves de programari a diferents nivells i de diferents tipologies, anàlisi de codi, proves de rendiment, proves d'accessibilitat, anàlisi de l'experiència d'usuari, etc.). Fins i tot l'anàlisi de comentaris i valoracions de les plataformes que fan accessible el programari als usuaris contribueix a governar la qualitat en un context com l'actual, amb una gran quantitat i diversitat d'aplicacions de programari que coexisteixen a la societat i a les organitzacions. La determinació de les activitats d'assegurament de la qualitat adequades per a cada context i la seva implementació constitueixen el que anomenem *enginyeria de la qualitat del programari*.

Òbviament, l'enginyeria de la qualitat es pot aplicar en menor o major mesura en els projectes, però sempre s'hauria d'associar als riscos de la no qualitat i les seves conseqüències. Podríem dir que l'enginyeria de la qualitat és com una inversió en forma d'assegurança en els projectes de desenvolupament de programari. Parteix de la base que el programari no està exempt de defectes de naturalesa diversa a mesura que es va desenvolupant i evolucionant. Si aquests defectes arriben als entorns de producció (on ja no són professionals els qui hi

interactuen, sinó els mateixos usuaris) es revelen en forma d'errors o simplement en forma d'experiències de baixa qualitat, cosa que deriva en riscos socials i de negoci. Un exemple d'un estudi recent assegura que el cost dels defectes de qualitat als Estats Units d'Amèrica ascendeix a més de dos mil milions de dòlars (Consortium for Information & Software Quality, 2021). D'altra banda, segons el *World quality report 2021-22* (Sogeti, Capgemini i Microfocus, 2021), els objectius executius principals de l'assegurament de la qualitat i les proves són els següents: 1) la custòdia de la qualitat en els projectes i la detecció de defectes abans de la posada en producció de canvis en el programari (*go-live*); 2) l'increment de la freqüència de les entregues de programari amb qualitat; 3) el suport transversal als equips en una visió àmplia de la qualitat com a objectiu compartit; 4) la contribució als objectius de negoci i la satisfacció final dels usuaris (*digital happiness*), i 5) la protecció de la mateixa marca i imatge corporativa en les organitzacions.

Es tracta, en definitiva, d'implementar el model VOICE (Marselis, Geurts, Veenendaal i Ruigrok, 2020): la implantació d'un model de definició d'objectius i valor esperat, d'indicadors per mesurar el valor i els riscos associats i la comparació amb el valor real experimentat com a retroacció per a la millora contínua del programari (figura 1).

Així doncs, cada aplicació de programari té uns riscos diversos de la no qualitat que s'han d'avaluar. A partir d'aquí, és necessari integrar una estratègia de qualitat en els processos de desenvolupament i entrega de programari amb tres objectius: 1) mesurar, fer seguiment i aportar transparència sobre els riscos de la no qualitat; 2) aportar retroacció sobre l'evolució dels diferents atributs de qualitat que cal tenir en compte, i 3) cooperar amb els diferents rols implicats en els processos de desenvolupament i entrega de programari per governar la qualitat i controlar-ne els riscos. Aquesta és la missió i el valor de l'enginyeria de la qualitat.

Aquest article s'estructura en sis seccions: en la primera secció s'ha introduït el concepte d'enginyeria de la qualitat del programari i el seu valor per al negoci i la societat; en la segona secció, que tenim a continuació, es presenten els principals atributs que determinen la qualitat del programari segons els estàndards i models de qualitat existents; en la tercera secció, es descriuen el conjunt d'estratègies, activitats, metodologies i tècniques per a la validació dels atributs de qualitat, així com la necessitat de definir una estratègia de qualitat en qualsevol projecte de programari; en la quarta secció, es presenta l'enfocament de la qualitat contínua en entorns de desenvolupament àgil; en la cinquena secció, es reflexiona sobre l'evolució i les enginyeries de la qualitat del programari i, finalment, en la sisena secció, es resumeixen les conclusions de l'article.

2. Atributs de qualitat del programari

La primera pregunta que cal fer-nos, des que concebem i definim una aplicació fins que en retirem l'ús, és «què entenem per qualitat del programari en la nostra aplicació?». De fet, el concepte de qualitat ha evolucionat. Si fa uns anys la qualitat s'associava principalment a la correcció funcional del programari (i per això l'activitat principal que se'n derivava eren les proves funcionals), avui en dia aquesta visió és molt més àmplia i posa també el focus en aspectes com el rendiment, la usabilitat, l'accessibilitat o la seguretat, com a atributs rellevants del que acaba sent l'objectiu final: assolir una bona experiència d'usuari en un món de canvis i d'expectatives que evolucionen.

Existeixen molts estàndards de qualitat que sustenten el marc d'actuació de l'enginyeria de la qualitat. L'ISO 9000 (sistemes de gestió de la qualitat) defineix un marc general que inclou la necessitat de «sistemes de gestió de la qualitat» en qualsevol organització. Això es concreta més en l'ISO/IEC 90003 (guies per a l'aplicació de l'ISO 9000 en l'àmbit del programari) i en l'ISO/IEC 25000 (requisits de qualitat del programari i avaluació).

Una de les parts d'aquest estàndard (25010) definia ja l'any 2005 una sèrie d'atributs de qualitat que calia considerar (figura 2), els quals aborden aspectes més enllà de la correcció funcional:

— *Compatibilitat*: capacitat del programari per operar en diferents plataformes i dispositius.

— *Portabilitat*: capacitat del programari per ser installat, substituït i adaptat a diferents entorns.

— *Mantenibilitat*: capacitat del programari per ser modificat de manera eficient. Això implica posar el focus en as-

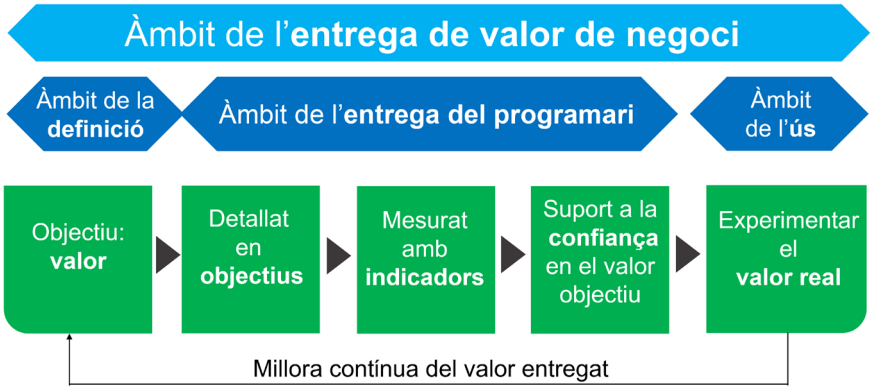

Figura 1. Model VOICE.
Font: Adaptat de Marselis, Geurts, Veenendaal i Ruigrok, 2020.

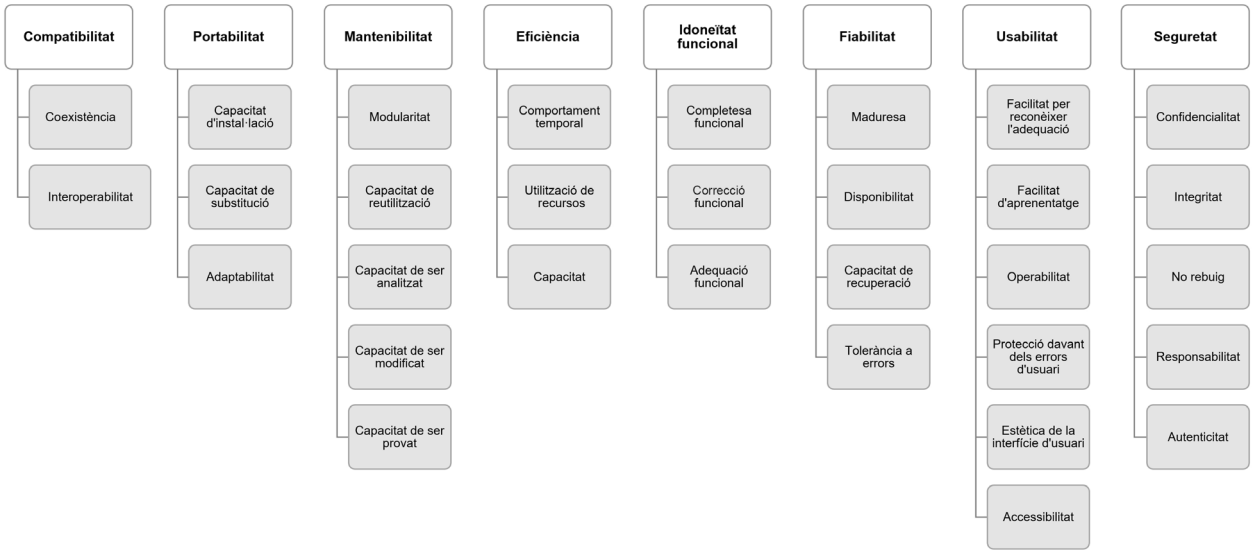

FIGURA 2. Model de qualitat del programari.
FONT: ISO/IEC 25000 - SQuaRE.

pectes com la modularitat, la reutilització i la capacitat de ser analitzat, modificat i provat.

— *Eficiència*: capacitat del programari perquè les funcionalitats operin de manera eficient en el temps, utilitzant els recursos de manera raonable i amb capacitat de resposta a la demanda d'ús dels usuaris.

— *Idoneïtat funcional*: capacitat del programari per donar resposta als requisits funcionals (completesa) de manera correcta i adequada.

— *Fiabilitat*: capacitat del programari d'oferir un comportament robust, disponible per als usuaris quan ho requereixin, i amb capacitat de tolerància a errors o recuperació.

— *Usabilitat*: capacitat del programari per oferir als usuaris una experiència senzilla i agradable d'utilitzar i d'aprendre. També s'hi inclou l'accessibilitat, que té per objectiu facilitar l'accés al programari de persones amb discapacitats sensorials.

— *Seguretat*: capacitat d'accedir i operar amb el programari amb confidencialitat, integritat, responsabilitat i autenticitat.

3. Estratègia, activitats, metodologies i tècniques d'enginyeria de la qualitat

En general, la complexitat del programari que es desenvolupa actualment va augmentant. Hi influeixen les funcionalitats i característiques que es requereixen, el nombre i la diversitat d'usuaris, i la compatibilitat requerida amb multitud de dispositius i plataformes, entre d'altres. Tot plegat s'ha de contextualitzar també en les tendències actuals del mercat del programari, que respon a la visió àgil de posar en producció noves versions de programari de manera freqüent. Això fa que els escenaris possibles de prova per assegurar la qualitat creixin exponencialment i s'hagin d'executar també més freqüentment, amb la qual cosa és impossible (amb els recursos limitats de què es disposi) de cobrir tots els escenaris de prova. Cal, doncs, prioritzar les proves que es fan i controlar el risc derivat de no provar-ho tot. Per això és imprescindible dotar-nos d'una estratègia.

A les organitzacions hi coexisteixen diferents tipus de projectes de desenvolupament, manteniment i evolució d'aplicacions de programari. Òbviament, no totes les aplicacions tenen la mateixa missió, ni els mateixos requisits, ni els mateixos objectius de qualitat, ni tampoc no s'utilitza la mateixa metodologia o enfocament del desenvolupament i entrega. Per exemple, una aplicació de venda en línia durant el Black Friday s'enfocarà més en el rendiment i la usabilitat. En aquest context, acostuma a haver-hi un marc general de qualitat en l'àmbit de l'organització, que es concreta, al seu torn, per a cada aplicació.

3.1. *Estratègia d'enginyeria de la qualitat*

Una estratègia d'enginyeria de la qualitat (figura 3) ha d'estar integrada, almenys, per: 1) una avaluació de riscos depenent del tipus d'aplicació i de l'entorn de desenvolupament; 2) la definició d'una sèrie d'activitats d'assegurament de la qualitat destinades a mitigar aquests riscos i fomentar majors nivells d'assoliment dels atributs de qualitat més rellevants al llarg del procés; 3) un enfocament d'implementació de les activitats que esculli les tècniques més adequades; 4) la definició d'indicadors i els mecanismes d'implementació de quadres de comandament de qualitat; 5) la determinació del grau de cobertura de les activitats i valors mínim (no s'admet el progrés d'una aplicació si no s'assoleix aquest mínim) i valors objectiu; 6) un procés de gestió dels defectes detectats, i 7) la integració de les activitats d'enginyeria de la qualitat en la metodologia i processos de desenvolupament, integració i posada en producció.

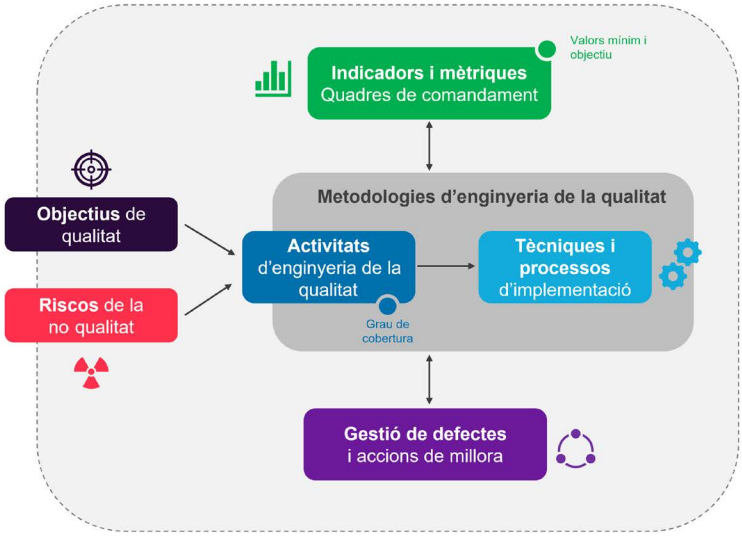

FIGURA 3. Visió general de components d'una estratègia d'enginyeria de la qualitat.
FONT: Elaboració pròpia.

3.2. *Activitats*

La concreció de les diverses activitats d'enginyeria de la qualitat aplicades en els projectes es basa en quatre eixos:

— *Tipus d'activitat*. Determina la tipologia de les activitats. Hi ha activitats d'anàlisi (per exemple, l'analítica de codi o la gestió de mètriques de qualitat), proves (que requereixen la interacció amb el sistema i la comprovació de resultats esperats), monitoratge (sondeig continuat i obtenció de mètriques) i elements de suport a la metodologia de desenvolupament i entrega del programari (informe de defectes, col·laboració en l'enginyeria de requisits, etc.).

— *Propòsit de qualitat*. Es determina en funció dels atributs de qualitat objectiu de cada activitat. Discriminem, per exemple, entre proves funcionals, proves de rendiment, proves d'usabilitat, proves de seguretat, anàlisi estàtica del codi per analitzar la seguretat o la mantenibilitat, proves de compatibilitat (per assegurar la qualitat d'una aplicació en diferents dispositius i/o plataformes), etc. Al mateix temps, podem discriminar entre proves de progressió (dirigides a provar noves funcionalitats) o proves de regressió (dirigides a provar de forma contínua aquelles funcionalitats clau del sistema per assegurar-ne la qualitat tot i els canvis en l'aplicació).

— *Nivell d'aplicació*. Les proves funcionals es poden aplicar en el codi (proves unitàries) per provar la correcció de funcionalitats específiques, en els serveis interns o externs utilitzats en l'aplicació a través de les interfícies de programació d'aplicacions (API, de l'anglès *application programming interfaces*), o en la interfície d'usuari per provar un procés d'extrem a extrem tal com ho faria un usuari.

— *Grau d'automatització*. Les diferents activitats poden executar-se manualment o bé automatitzar-se totalment o parcialment. Aquest grau d'automatització acostuma a dependre del tipus d'activitats, el seu propòsit i el nivell d'aplicació. Així doncs, en l'àmbit de les proves, acostuma a haver-hi un grau d'automatització més gran en les proves unitàries o d'integració, que no pas en les que són d'extrem a extrem, que són més costoses i, per tant, se centren en els processos més crítics de negoci.

Les proves de programari són el tipus d'activitat d'enginyeria de la qualitat més rellevant, amb diferents tipologies i graus. La figura 4 mostra un exemple de classificació de les activitats de proves en forma de quadrant. A l'eix horitzontal s'especifica el grau de millora interna o externa que es persegueix com a objectiu. A l'eix vertical s'especifica el grau tecnològic o de negoci de les proves. En funció del tipus de proves, el grau d'automatització recomanat és diferent.

Com a resultat de la unificació de diferents estàndards anteriors, el 2016 va aparèixer l'ISO/IEC/IEEE 29119 (proves de *software*), que estructura, en l'àmbit de les proves de programari (com a activitat principal de l'enginyeria de qualitat), els conceptes i les definicions, els processos, la documentació i les tècniques per a la realització de les proves.

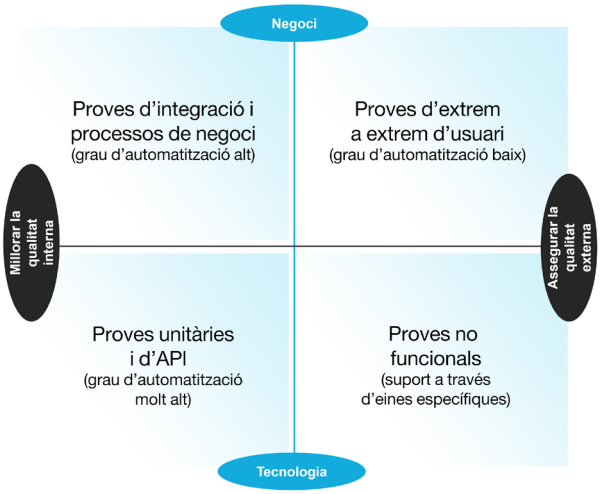

FIGURA 4. Exemple de quadrant de les proves de programari.
FONT: Adaptat de Sogeti, 2020.

3.3. *Metodologies i tècniques*

Cada activitat té tècniques associades per implementar-les de manera eficient i aplicant bones pràctiques. A més, cada tipus d'activitat disposa de metodologies i enfocaments que descriuen les tècniques associades i els processos per executar-les.

A tall d'exemple, en aquest article ens referirem a algunes tècniques rellevants en l'àmbit de les proves de programari incloses en la metodologia TMap (Sogeti, 2020): tècniques de disseny de proves, assegurament de la qualitat amb moviment de les proves a l'esquerra (*shift-left*), tècniques d'automatització de proves, procés de gestió de defectes, informes i indicadors de qualitat. Cadascuna d'aquestes tècniques pot estar suportada per eines i entorns de treball específics que existeixen en el mercat.

3.3.1. Tècniques de disseny de proves

Més enllà de les proves exploratòries (aquelles que es fan sense estructura prèvia), la necessitat de garantir una cobertura adequada d'acord amb els recursos existents i poder obtenir una retroacció estructurada sobre el veredicte de les proves implica necessàriament un procés estructurat per al disseny de proves.

El disseny estructurat de proves respon al repte de dissenyar el conjunt de proves que permeti cobrir el màxim possible d'escenaris potencials de prova. Existeixen diferents tècniques de disseny:

— *Orientades al procés*. Es dissenyen les proves a partir de l'anàlisi dels diferents passos dels processos de negoci.

— *Orientades a les condicions*. Es dissenyen les proves amb tècniques d'anàlisi detallada de les condicions d'una determinada funcionalitat.

— *Orientades a les dades*. Es dissenyen les proves amb tècniques d'anàlisi de les dades involucrades en un objectiu de prova.

— *Orientades a l'experiència d'usuari*. Es dissenyen les proves amb focus en aspectes de qualitat no funcionals.

3.3.2. Assegurament de la qualitat amb moviment de les proves a l'esquerra (*shift-left*)

Un dels enfocaments clau aplicable a diverses activitats d'enginyeria de la qualitat és el que es coneix com a *canvi a l'esquerra* (*shift-left*). L'objectiu és detectar defectes en el procés de desenvolupament i entrega del programari com més aviat millor, fins i tot des de les etapes inicials del procés. Això significa definir i executar proves en etapes inicials del desenvolupament, ja sigui a través de simulacions o utilitzant, si cal, tècniques de virtualització per a serveis encara no desenvolupats o inaccessibles en temps de realització de les proves.

L'enfocament *shift-left* té com a objectiu reduir els efectes de la regla del 10 (The Standish Group International, 2014),

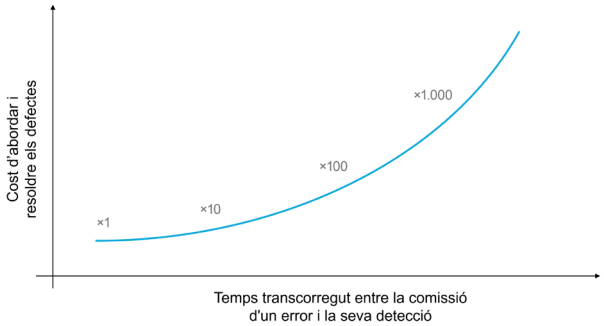

FIGURA 5. La regla del 10 (cost dels defectes).
FONT: Adaptat de The Standish Group, 2014.

que assegura que el cost per resoldre un defecte es multiplica de l'ordre de 10 a mesura que anem avançant en fases més tardanes fins a la posada en producció del programari, quan el cost d'un defecte és elevat, perquè s'ha anat multiplicant exponencialment a mesura que el detectem més tard (figura 5).

3.3.3. Tècniques d'automatització de proves

L'automatització de proves és una tècnica clau per fer més eficient l'execució de proves de regressió (aquelles que es repeteixen més d'una vegada davant de canvis freqüents en el programari). Fonamentalment consisteix a desenvolupar un sistema que prova automàticament un altre sistema i, per tant, robotitza les interaccions amb el sistema, compara els resultats reals amb els esperats i reporta els veredictes de qualitat automàticament. Existeixen dues aproximacions: la basada en gravació/reproducció (eines que permeten gravar interaccions amb la pantalla i reproduir-les) i els entorns de treball d'automatització modulars amb una aproximació d'enginyeria del programari (amb codi associat i biblioteques / eines d'automatització) que fomenten la reutilització i la reducció del manteniment. Els entorns de treball d'automatització han d'incloure el repositori de proves (codificació), un motor d'execució (que permet la interpretació i l'execució de les proves), la tecnologia d'interacció amb el sistema depenent del nivell (el codi, les API o els elements de pantalles), i el mecanisme d'informe dels veredictes.

L'automatització de proves és aplicable als diferents nivells de proves (proves unitàries, proves d'integració o proves d'extrem a extrem dels processos de negoci), però la complexitat d'automatització és més gran en les proves de processos de negoci que en les proves tècniques unitàries. Per aquest motiu, tal com representa la piràmide de la figura 6, la majoria d'estratègies d'automatització de proves busquen un balanç entre l'esforç d'automatització i el retorn. D'aquesta manera, es fomenta la realització de més proves unitàries com a base, mentre que s'aconsella focalitzar l'automatització de proves de processos de negoci a les proves més prioritàries.

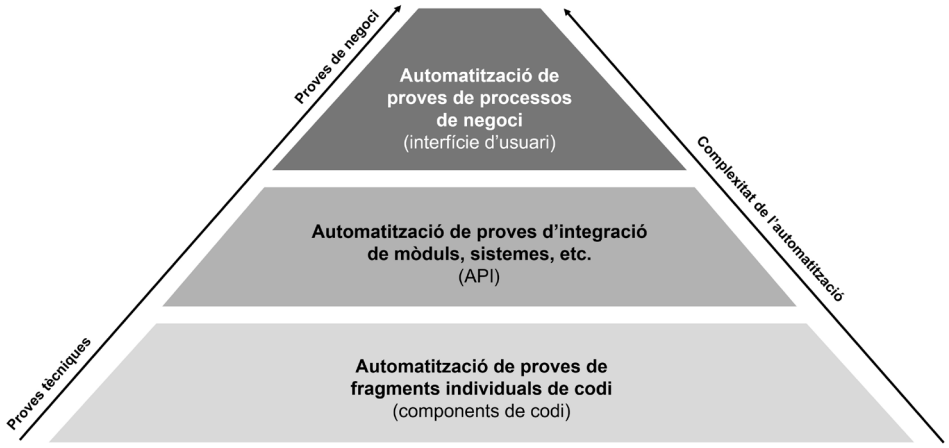

FIGURA 6. Piràmide d'automatització de proves.
FONT: Elaboració pròpia.

3.3.4. Procés de gestió de defectes

Un defecte és el resultat d'un error en el procés de desenvolupament o entrega del programari, que resideix en el codi, en la infraestructura o en la mateixa documentació.

Un dels objectius de la major part d'activitats de proves de programari és la detecció de defectes com més aviat millor (*shift-left*). Una vegada detectats els defectes potencials, aquests s'han de reportar i emmagatzemar en algun sistema de gestió de proves i defectes accessible per a la resta de parts interessades del projecte. A partir d'aquí, s'han de gestionar a través d'un procés de validació i correcció, si s'escau. La figura 7 representa un procés marc de gestió de defectes. Quan un enginyer de qualitat informa d'un defecte nou, s'obre un procés de consulta entre les parts interessades per tal de posposar, rebutjar o confirmar aquest defecte. Quan s'informa de la seva correcció, cal iniciar un procés per provar de nou i confirmar que s'ha solucionat.

3.3.5. Informes i indicadors de qualitat

Existeix una gran diversitat d'activitats d'enginyeria de la qualitat que acaben recollint dades (sobre execucions de casos de prova, defectes, anàlisi de codi, monitoritzacions, etc.). A més, a les organitzacions acostumen a coexistir múltiples fonts de dades i d'eines de suport diferents, així com múltiples projectes en desenvolupament. Aquestes dades són molt útils per a tres objectius: 1) obtenir retroacció continuada a través de mètriques, indicadors i acords de nivell de servei (SLA, de l'anglès *service level agreements*) de qualitat; 2) aportar transparència sobre les activitats d'enginyeria de la qualitat dutes a terme, i 3) disposar d'una capa de govern de la qualitat en els projectes que integri les dades de diferents fonts.

La figura 8 mostra la visió general d'un enfocament per a la implementació d'un sistema de quadres de comandament de qualitat. En primer lloc, cal implementar connectors a les diferents fonts de dades. En segon lloc, cal realit-

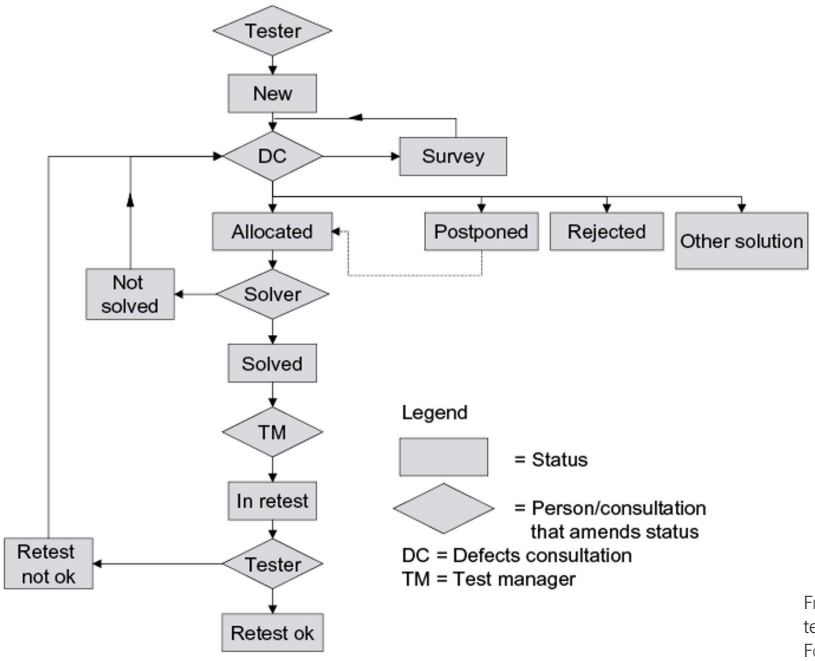

FIGURA 7. Exemple d'un procés marc de gestió de defectes del programari.
FONT: Sogeti, 2020.

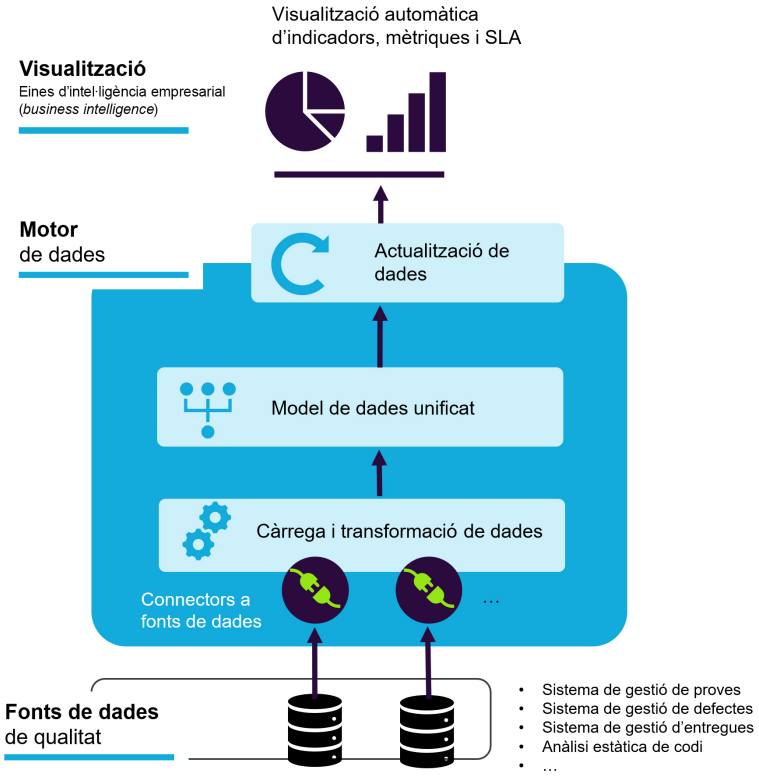

FIGURA 8. Visió general de components d'un sistema de quadres de comandament de qualitat.
FONT: Elaboració pròpia.

zar transformacions per adaptar les dades, relacionar-les i emmagatzemar-les en una base de dades que segueixi un model de dades de qualitat unificat. En tercer lloc, cal establir un procés de càrrega, transformació i actualització de les dades. Finalment, s'ha d'implementar la visualització dels quadres de comandament, utilitzant característiques d'intel·ligència de negoci ja existents al mercat. La figura 9 mostra alguns exemples de quadres de comandament de qualitat.

4. Enfocament de qualitat contínua en aproximacions de desenvolupament àgil

Existeixen diferents aproximacions de desenvolupament i entrega de programari: des dels models seqüencials fins als models àgils (Agile, DevOps...), passant per models híbrids aplicables a grans organitzacions.

En els models seqüencials tradicionals, les activitats d'assegurament de qualitat eren considerades una fase (final) específica per validar funcionalment el programari un cop construït. Amb l'arribada de metodologies de desenvolupament i entrega contínua de programari més iteratives, aquest és un enfocament que ha perdut la vigència. L'enginyeria de qualitat no és un simple punt de control final basat en l'experiència de les cadenes de muntatge industrial dels anys seixanta, sinó una activitat de control de qualitat i, sobretot, de provisió de retroacció continuada per ajudar i contribuir a millorar la qualitat final del programari en col·laboració amb tots els agents implicats en el seu desenvolupament.

Per aquest motiu, no es pot entendre la qualitat en entorns de desenvolupament Agile i DevOps[1] (Kim, Debois, Willis, Humble i Forsgren, 2021), sense un conjunt d'activitats de qualitat que han d'abraçar la planificació, la codificació, la integració, el desplegament i l'operació de les aplicacions de programari. La figura 10 classifica les diferents activitats d'assegurament de la qualitat en activitats organitzatives (estratègia, polítiques, responsabilitats, monitoratge, processos, estimacions, mètriques...) i operatives (aquelles activitats que implementen l'enginyeria de la qualitat i proveeixen dades).

En aquests contextos de desenvolupament, és encara més rellevant posar el focus en la transparència (quadres de comandament) i l'automatització (de proves i tasques periòdiques). Complementàriament, les aproximacions DevOps per a l'entrega contínua de programari es basen en cadenes d'integració i de desplegament continuat del codi, en les quals cal integrar també les activitats d'enginyeria de la qualitat com a part del procés.

5. Perspectives i evolució de l'enginyeria de qualitat

La qualitat del programari és un concepte en evolució, ja que ha d'adaptar-se a les expectatives i necessitats d'una

1. DevOps és un conjunt de pràctiques que combina el desenvolupament de programari (Dev) i les operacions (Ops). Té com a objectius la reducció del cicle de vida del desenvolupament de programari i l'entrega contínua de programari amb qualitat (*Viquipèdia*).

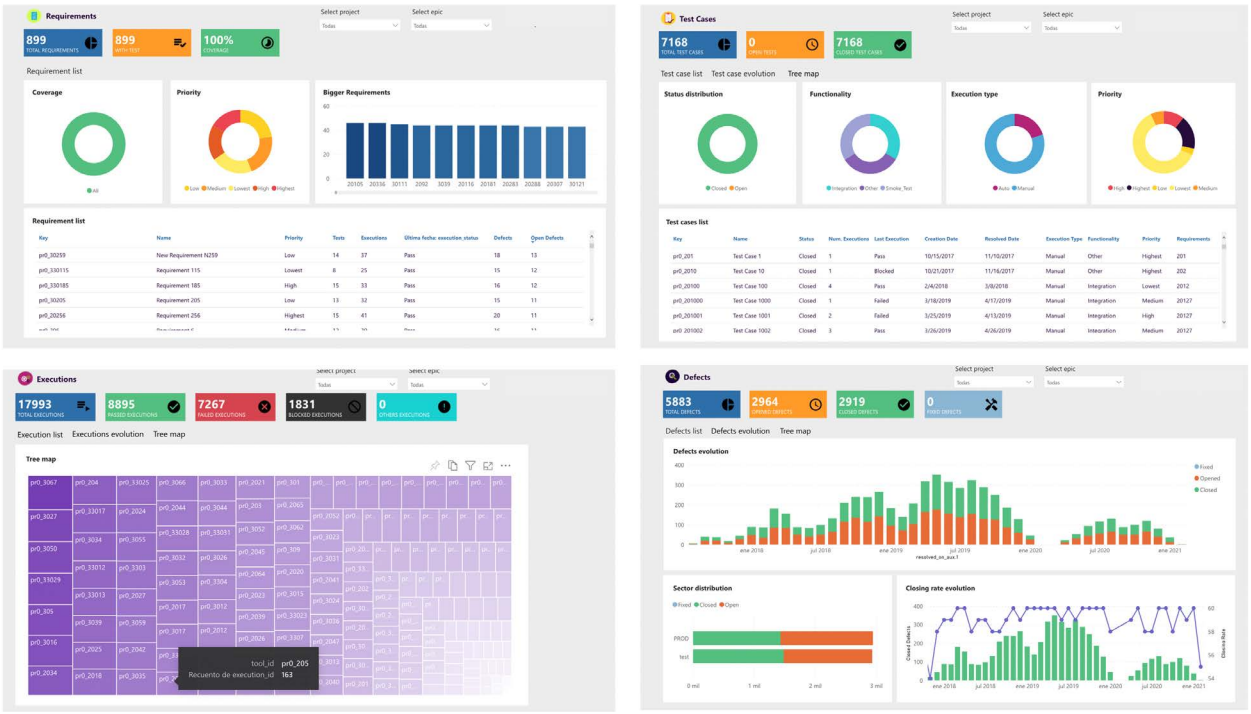

FIGURA 9. Exemples de demostració de SogetiLabs España.
FONT: SogetiLabs España.

societat canviant. Si fa uns anys el focus de la qualitat s'entenia fonamentalment des del punt de vista funcional, avui en dia els atributs de qualitat valorats pels usuaris, i per tant, factors d'èxit de les aplicacions de programari, tenen un espectre més ampli. Fins i tot és necessari començar a incloure com a atributs de qualitat aspectes com la inclusió i la sostenibilitat social (són les aplicacions de programari suficientment inclusives atenent la diversitat social que existeix i, per tant, la diversitat d'usuaris potencials?).

Tot plegat implica també una evolució clara en els rols professionals que participen en els projectes amb una visió d'enginyeria de la qualitat. Actualment, no n'hi ha prou amb perfils de negoci que fan proves funcionals manualment, sinó que són necessaris perfils d'enginyeria amb coneixements i habilitats transversals (infraestructura, desenvolupament, tècniques d'automatització, enginyeria de requisits, metodologia, analítica, etc.) per tal de cobrir diferents activitats de qualitat al llarg del procés de desenvolupament i entrega.

És evident que la gran quantitat de desenvolupaments i manteniments d'aplicacions en multitud d'àmbits i amb un ampli espectre de tecnologies de desenvolupament fa que l'enginyeria de qualitat comporti constantment nous reptes. Un d'aquests reptes, actualment, és la capacitat de gestió de diferents entorns (desenvolupament, proves, producció...) i la gestió de les dades en aquests diferents entorns amb seguretat i eficiència (*test data management*). També existeix l'oportunitat d'aprovisionar i gestionar en-

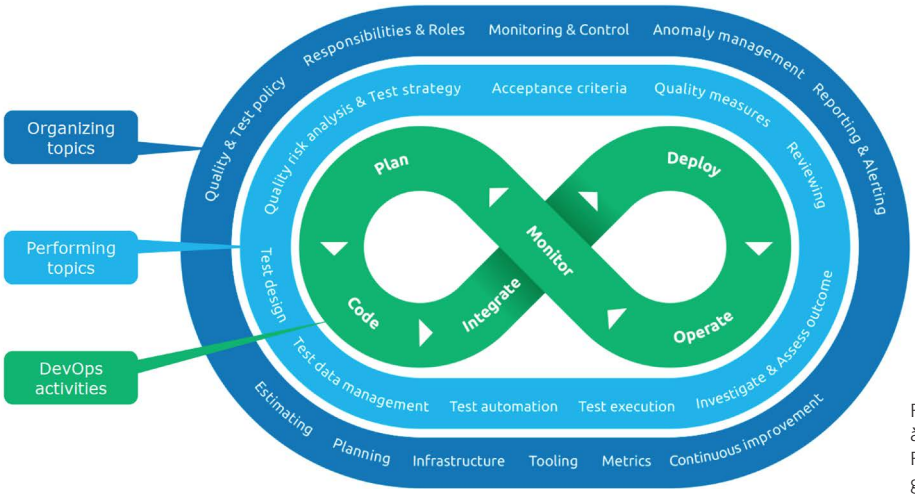

FIGURA 10. Activitats de qualitat en entorns àgils i DevOps.
FONT: Marselis, Geurts, Veenendaal i Ruigrok, 2020.

torns de proves al núvol, que poden incrementar la flexibilitat i l'optimització de costos, ja que es poden crear, redimensionar i eliminar de manera eficient d'acord amb les necessitats de realització de proves.

Finalment, és important destacar les aplicacions que ja està començant a tenir la intel·ligència artificial (aprenentatge automàtic, classificació, predicció…) per tal d'assistir la presa de decisions (priorització de proves, selecció automàtica de proves de regressió, classificació de defectes, etc.) en l'enginyeria de la qualitat (Tort, 2018). Diverses eines de suport ja inclouen funcionalitats amb intel·ligència artificial, i s'espera que sigui un camp de recerca i de millora creixent.

La figura 11 mostra un esquema de components per a la gestió integrada de la qualitat en entorns DevOps, a través de la implementació agregada d'un sistema d'intel·ligència que aporti retroacció i dades per a la computació de pas a través de portes de qualitat intel·ligents (*smart quality gates*) (Tort, 2021) integrades en les cadenes DevOps. Les portes de qualitat (*quality gates*) són elements de comprovació al llarg de la cadena de desenvolupament que, a través d'indicadors amb valors mínims, determinen si una nova contribució de programari pot progressar cap a producció. D'aquesta manera, es redueixen progressivament els riscos de possibles defectes de qualitat, a mesura que les noves versions de programari van progressant en la cadena de desenvolupament. De la mateixa manera, si no es compleixen els mínims indicats a les portes de qualitat, és necessari retirar la contribució de programari i tornar-la a processar a través de la cadena de portes de qualitat quan tingui una qualitat suficient.

La implementació de portes de qualitat pot enriquir-se amb informació, prediccions i models analítics que permetin comprovacions més intel·ligents (basades en dades prèvies, aprenentatge automàtic…). En aquests casos, és quan parlem de *portes de qualitat intel·ligents*. Aquesta visió es considera actualment una aproximació avançada de gestió de la qualitat per a les aplicacions de programari.

6. Conclusions

En aquest article s'ha introduït el concepte d'enginyeria de la qualitat i el seu valor per al negoci i la societat. Així mateix, s'han presentat les activitats d'enginyeria de la qualitat enfocades a la detecció anticipada de defectes, a la reducció de riscos i a la generació de retroacció en els projectes, especialment de manera continuada en entorns de desenvolupament àgil. Finalment, s'ha exposat una visió de les perspectives de la gestió de la qualitat del programari, que adquireixen cada vegada més un enfocament transversal, ampli i en evolució, d'acord amb les expectatives socials i de negoci actuals.

Agraïments

Vull agrair a Antoni Olivé, que en el seu dia va introduir-me en el món de la qualitat del programari, l'oportunitat d'escriure aquest article, així com els seus comentaris per millorar-lo. D'altra banda, també vull agrair a tot l'equip professional de Sogeti i del grup Capgemini la seva contribució

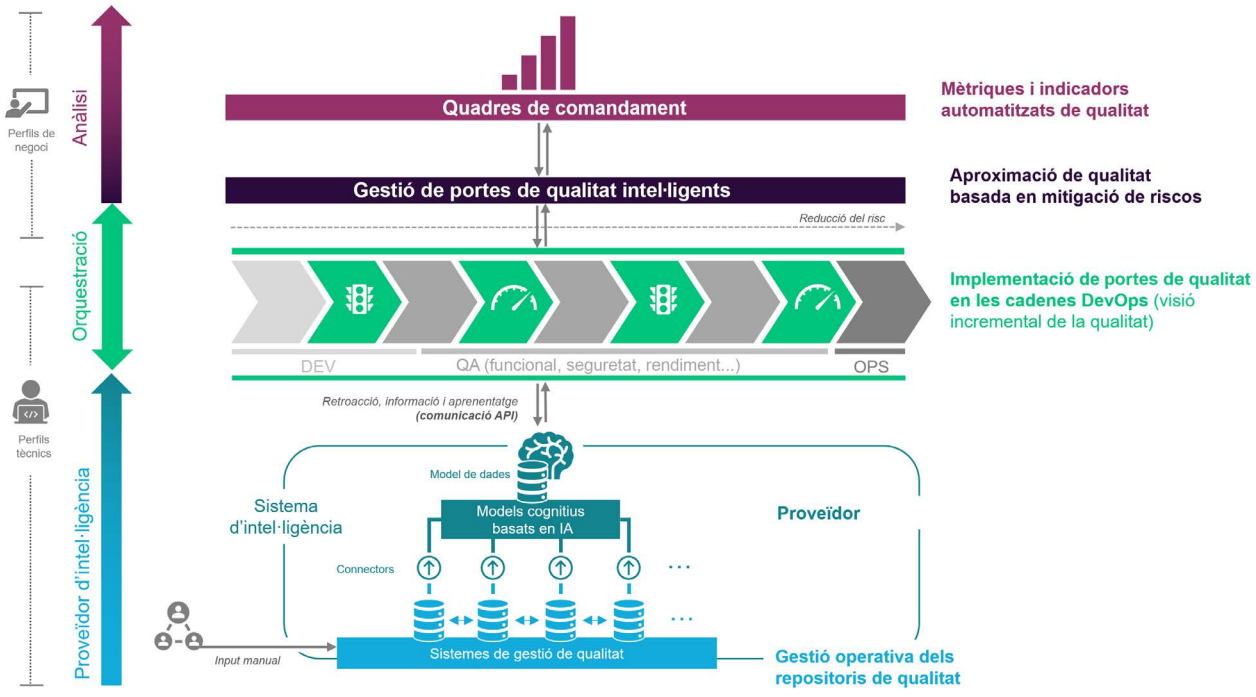

FIGURA 11. Components d'un model de portes de qualitat intel·ligents integrades en un entorn de desenvolupament Agile/DevOps.
FONT: Elaboració pròpia.

i aprenentatges sobre l'enginyeria de la qualitat del programari.

Bibliografia

Beck, K. E. (2001). *Manifesto for agile software development* [en línia]. <https://agilemanifesto.org/> [Consulta: 20 abril 2022].

Consortium for Information & Software Quality (2021). *The cost of poor software quality in the US: A 2020 report* [en línia]. <https://www.it-cisq.org/pdf/CPSQ-2020-report.pdf> [Consulta: 20 abril 2022].

Kim, G.; Debois, P.; Willis, J.; Humble, J.; Forsgren, N. (2021). *The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations*. 2a ed. Portland, EUA: IT Revolution.

Marselis, R.; Geurts, D.; Veenendaal, B. van; Ruigrok, W. (2020). *Quality for DevOps teams*. Holanda: Sogeti.

Sogeti (2009). *TPI next: Business driven test process improvement*. Holanda: UTN Publishers.

Sogeti (2020). *TMap methodology portal* [en línia]. <www.tmap.net> [Consulta: 2 maig 2022].

Sogeti; Capgemini; Microfocus (2021). *World quality report: 2021-22* [en línia]. 13a ed. <https://www.sogeti.es/explora/publicaciones/world-quality-report-2021-22/> [Consulta: 30 abril 2022].

The Standish Group International (2014). *Rule of ten* [en línia]. <https://www.standishgroup.com/sample_research_files/RuleTen.pdf> [Consulta: 1 maig 2022].

TMMi Foundation (2018). *Test maturity model integration (TMMi®): Guidelines for test process improvement. Release 1.2* [en línia]. Irlanda. <https://tmmi.org/tm6/wp-content/uploads/2018/11/TMMi-Framework-R1-2.pdf> [Consulta: 15 abril 2022].

Tort, A. (2018). «Is (artificial) intelligence needed for testing?» [en línia]. *SogetiLabs blog* (12 juny). <https://labs.sogeti.com/is-artificial-intelligence-needed-for-testing/> [Consulta: 30 abril 2022].

— (2021). *Smart quality gates: State of AI applied to quality engineering* [en línia]. <https://www.sogeti.com/ai-for-qe/> [Consulta: 30 abril 2022].