Зборник радова Факултета техничких наука, Нови Сад

УДК: 004.45:005.93
ДОИ: https://doi.org/10.24867/34OI01Okiljevic

Идентификација и анализа техничког дуга у софтверу за заказивање термина

Identification and Analysis of Technical Debt in Appointment Scheduling Software

Милица Окиљевић, *Факултет техничких наука, Нови Сад*

Студијски програм – ИНЖЕЊЕРСТВО ИНФОРМАЦИОНИХ СИСТЕМА

Кратак садржај – *Технички дуг је метафора која описује креирање компромиса у изради изворног кода, где се одлуке доносе ради бржег развоја у кратком року, али касније стварају додатне трошкове кроз одржавање и дораду. Овај концепт обухвата не само кодирање, већ и шири оквир планирања рокова, испорука и функционалности пројекта. Циљ рада је да објасни основу техничког дуга и његову категоризацију, укаже на утицај у развоју софтвера и уради анализу појаве техничког дуга у реалном систему софтвера за заказивање термина. Резултати показују да су најизраженији архитектонски, дизајнерски и дефектни дуг, док су промењиви захтеви и временски притисак кључни фактори њиховог настанка.*

Кључне речи: *Технички дуг, развој софтвера, софтвер за заказивање термина*

Abstract – *Technical debt is a metaphor that describes making compromises during software development to speed up short-term progress, which later leads to extra costs for maintenance and improvements. This concept includes not only coding but also project planning, delivery deadlines, and functionality. The aim of this paper is to explain the basics and categorization of technical debt, show its impact on software development, and analyze its occurrence in a real appointment scheduling software system. The results show that the most common types of technical debt are architectural, design, and defect debt, while changing requirements and time pressure are key factors behind their appearance.*

Keywords: *Technical debt, Software development, Appointment scheduling software*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Дарко Стефановић, ред. проф.

1. УВОД

Технички дуг је појам који је први увео *Ward Cunningham* [1] 1992. године, описујући га као „не баш исправан код“ (енг. *not-quite-right code*) који развојни тим свесно оставља у систему како би убрзао краткорочни напредак. Касније је ова дефиниција проширена и данас обухвата све развојне активности које се свесно одлажу, иако постоји ризик да у будућности изазову проблеме у пројекту [2].

Развој софтвера захтева балансирање између брзе испоруке и имплементације решења, услед чега често долази до компромисних одлука. Иако такве одлуке могу бити корисне у кратком року, њихов кумулативни ефекат може значајно отежати одржавање и даљи развој система. Управо због тога расте потреба за препознавањем, мерењем и управљањем техничким дугом, како би се спречило његово дуготрајно негативно деловање на квалитет софтвера и развојне процесе.

Циљ рада је да укаже на значај детектовања и мерења техничког дуга током развоја софтвера на примеру реалног система из окружења. Истакнута је важност препознавања различитих врста техничког дуга и њихово решавање, како би се побољшала одрживост и могућност надоградње система, као и на важност квалитетног дефинисања захтева и документације као кључних фактора у превенцији његовог настанка.

У раду је примењен квалитативни приступ који обухвата анализу релевантне литературе и идентификацију појава техничког дуга у реалном систему. Методологија се заснива на систематском прегледу литературе и студији случаја кроз анализу *Petify* апликације као представника система за заказивање термина.

Структура рада је следећа. Поглавље два дефинише основне појмове за разумевање и појаву техничког дуга. Поглавље три објашњава коришћење методологије и опис реалног система. Поглавље четири представља конкретне примере техничког дуга у софтверу. Поглавље пет се односи на дискусију и резултате анализе и рада. На крају, у закључку су сумирани кључни налази и предложени даљи развој ове теме.

2. ТЕОРИЈСКЕ ОСНОВЕ

У оквиру овог поглавља су описани основни појмови везани за технички дуг и његову класификацију.

2.1. Појам техничког дуга

Технички дуг је метафора први пут дефинисана 1992. године и представља интерне задатке за које тим одлучи да се прескочи њихово извршење са ризиком да ће потенцијално створити проблем касније. Метафора дефинише дуг који настаје када развојни тим изабере краткорочно решење, које носи велику могућност негативног утицаја и последица дугорочно [3].

Суштина ове метафоре огледа се у поређењу са финансијским дугом где развојни тим „задужује" пројекат бржим напретком у садашњем тренутку, али тај дуг касније мора бити „отплаћен с каматом" кроз додатне трошкове, сложеније одржавање и рефакторисање система. Технички дуг не обухвата само лош код, већ и шири спектар развојних активности које су привремено заобиђене. Може се односити на, на пример, застарелу или непотпуну документацију, тестове који нису спроведени, превише сложен код који би требало реструктурирати или познате дефекте који остају неисправљени. Последице таквих одлука најчешће се манифестују кроз неочекивана кашњења, проблеме у имплементацији нових функционалности и потешкоће у одржавању дефинисаних критеријума квалитета [2].

Технички дуг се јавља у софтверским пројектима када је потребно бирати између одржавања високог квалитета система и брзе испоруке уз ограничене ресурсе. Ове компромисне одлуке често су неизбежне у раним фазама развоја, али њихово игнорисање може довести до озбиљних последица на дужи рок.

2.2 Класификација техничког дуга

Martin Fowler [4] је категорисао технички дуг пратећи две главне карактеристике: необазривост/промишљеност и намерно/ненамерно. Недуго након његове класификације, *Steve McConnell* [5] је категорисао технички дуг у намерне и ненамерне. Временом, развијен је нов начин класификације техничког дуга који се заснива на природи проблема из којих различите врсте дуга настају. Овај приступ омогућава јасније разумевање сложености техничког дуга у софтверским пројектима и идентификацију специфичних индикатора за сваку врсту дуга током развоја софтвера. На тај начин се олакшава препознавање и практична примена одговарајућих метода и алата за њихову детекцију у реалним пројектима. У наставку ће бити објашњена свака врста техничког дуга понаособ [6].

Архитектонски дуг – Односи се на проблеме који настају у архитектури софтверског пројекта, као што су нарушавање модуларности и смањење квалитета унутрашњих структура. Овај дуг негативно утиче на архитектонске захтеве попут перформанси, робусности и скалабилности, а његово отклањање захтева додатно време и напор у процесу рефакторисања архитектуре.

Дизајнерски дуг – Односи се на дуг који се може открити анализом изворног кода и идентификовањем пракси које крше принципе доброг објектно-оријентисаног дизајна. Овакав дуг отежава проширење, тестирање и рефакторисање система.

Дефектни технички дуг – Односи се на познате грешке (дефекте) у софтверском систему које су уочене током тестирања или пријављене од стране корисника, а које су евидентиране у системима за праћење багова. Њихово решавање се често одлаже због ограничених ресурса и конкурентских приоритета унутар пројекта и честа је појава нагомилавања дефектних дугова.

Дуг у процесу изградње – Технички дуг који се односи на проблеме у процесу изградње (енг. *build*), који тај процес чине непотребно сложеним и спорим. Процес изградње може садржати беспотребан код или зависности које нису јасно дефинисане, што доводи до непотребног трошења времена и ресурса.

Кодски дуг – Односи се на проблеме у изворном коду који негативно утичу на његову читљивост и одржавање. Такви проблеми често потичу од лоших програмерских пракси. Овај дуг се обично може идентификовати прегледом изворног кода или помоћу алата за статичку анализу.

Документациони дуг – Односи се на проблеме у документацији софтверског пројекта попут недостатка, неадекватности или непотпуности документације. Документација може формално постојати и функционисати, али да не задовољава квалитетне стандарде (нпр. неажурност, непрецизност, изостанак техничких детаља). Ово отежава разумевање система и продужава време увођења нових чланова тима.

Инфраструктурни дуг – Односи се на проблеме у инфраструктури организације који могу одлагати или отежавати развојне активности. Примери укључују одлагање надоградњи сервера, база података или алата, што касније узрокује проблеме са перформансама, безбедношћу и компатибилношћу.

Процесни дуг – Односи се на неефикасне или застареле процесе. На пример, процес који је раније био прикладан више не одговара тренутним потребама пројекта, што доводи до застоја, дужег циклуса испоруке и неспоразума у тиму.

Дуг у захтевима – Настаје када се праве компромиси у вези са тиме који захтеви ће бити имплементирани или на који начин. Примери укључују делимично имплементиране захтеве, имплементације које не покривају све случајеве, или оне које не задовољавају све нефункционалне захтеве (нпр. сигурност, перформансе).

Тестерски дуг – Означава проблеме у тестирању који утичу на квалитет. Примери укључују планиране тестове који нису извршени или познате недостатке у сету тестова, као што је ниска покривеност кода тестовима. Овај дуг повећава ризик да дефекти прођу неоткривени у продукцију.

3. МЕТОДОЛОГИЈА

Методолошки оквир рада заснива се на комбинацији прегледа литературе у области и анализе техничког дуга на примеру реалног софтверског система. Овакав приступ омогућава повезивање теоријских концепата и класификација техничког дуга са практичним примерима из стварног развојног окружења.

3.1. Преглед литературе

Систематичан преглед литературе заснива се на анализи досадашњих истраживања која обухватају методе идентификације и мерења техничког дуга, приступе засноване на статичкој анализи кода, као и новије правце усмерене ка техничком дугу у фази захтева.

Идентификација и мерење техничког дуга

Идентификација и мерење техничког дуга представљају кључне кораке за његово разумевање и управљање. Мерење техничког дуга омогућава организацијама да јасно разумеју који су проблеми присутни у систему и да на основу тога дефинишу приоритете за њихово решавање и ојачају стабилност и одрживост пројекта [7].
Најчешће истраживани приступи мерењу техничког дуга ослањају се на алате за статичку анализу кода, који пружају објективне метрике о квалитету софтвера, попут комплексности кода, присуства дуплираних делова и лоших програмерских пракси [8].

Технички дуг у фази захтева

Последњих година све више пажње посвећује се детекцији техничког дуга заснованој на захтевима (енг. *Requirements Engineering Debt, RED*) која се фокусира на анализу проблема насталих у фази дефинисања захтева, како би се идентификовали индикатори потенцијалног техничког дуга.
RED представља проширење традиционалног концепта техничког дуга са нивоа имплементације на ниво инжењеринга захтева, при чему се дуг не односи само на код или архитектуру, већ и на недоследне, непотпуне или нејасне захтеве који могу довести до каснијих трошкова у развоју и одржавању система [9]. Последице се испољавају у виду пада квалитета софтвера, повећања трошкова имплементације и одржавања, као и смањења поузданости и задовољства корисника.

3.2. Опис реалног система

У оквиру рада биће анализирана појава техничког дуга на систему за резервацију термина Petify, апликације која повезује кориснике и пружаоце услуга (салоне и хотеле) специјализованих за кућне љубимце и олакшава им процес заказивања термина. Апликација покрива претрагу услуга, резервације, нотификације и специјалне погодности чланства. Администраторски део система Petify је веб апликација која представља календарски систем вођења резервација уз базу клијената и евиденцију ценовника и трошкова.
Софтвер је имплементиран у *.NET* технологији за бекенд део пројекта, док су на фронтенду коришћени *React* и *React Native* технологије, а подаци се чувају у релационој бази података *PostgreSQL*. Систем је грађен итеративно и у више развојних фаза, са честим изменама и проширењима захтева од стране клијената. *Petify* је софтвер који је вођен у малом тиму, што је резултовало брзим доношењем одлука и променама у току имплементације. Управо овакав контекст чини *Petify* апликацију погодним примером за анализу и праћење појаве различитих врста техничког дуга, нарочито у ситуацијама када се систем брзо прилагођава захтевима тржишта и корисника.
Анализа на реалном систему *Petify* обухвата различите врсте техничког дуга (архитектонски, дизајнерски, кодски, дефектни и дуг у захтевима), као и њихове међусобне односе. Посебан фокус стављен је на захтеве као иницијални извор техничког дуга (*RED*), будући да промене и непрецизности у захтевима често узрокују каснију појаву других врста дуга.
Детекција техничког дуга у систему *Petify* заснива се на анализи одлука развојног тима и промена кроз више итерација система. Праћени су примери у којима су краткорочна решења и непрецизно дефинисани захтеви доводили до проблема у архитектури, дизајну и одржавању кода.

4. ПРИМЕРИ ТЕХНИЧКОГ ДУГА У СОФТВЕРУ

Идентификација и мерење су прва два корака у процесу менаџмента која одређују тип техничког дуга и њихов утицај на софтвер [2]. У овом поглављу биће анализирани конкретни примери појава техничког дуга у реалном систему *Petify* који су оставили највећи утицај и последице на даљи развој.

4.1. Архитектонски технички дуг

Приликом реализовања нових пројеката честа је појава овог дуга, јер је акценат на брзој испоруци решења, како би се софтвер тестирао и валидирао што пре од стране корисника, клијената и инвеститора. Овај дуг је уочен и на примеру *Petify* апликације, која је у фази пројектовања и идеје била само за резервацију термина у домену салона. Архитектура је била једноставна, корисник је бирао салон, услугу и слободан термин. Модел података и интерфејс за програмирање апликација (енг. *API – Application Programming Interface*) су се састојали од једноставног ентитета попут *Appointment* који садржи податке *salonId*, *serviceType*, *dateTime* и *petId*. Проблем је настао када је одлучено да апликација обухвата и систем за резервацију хотела, који представља систем заказивања, са напредним захтевима попут различитих термина пријављивања/одјављивања, различитих врста соба и додатних услуга (храна, нега) као и капацитет термина по соби. Како је постојећи систем био креиран за једнодневне термине и ентитет салона, додавање хотелских резервација није било могуће једноставним проширењем модела, већ је било неопходно рефакторисати постојећу базу података да укључује вишедневне резервације са додатним услугама и измене у контролерима и сервисима који разликују ентитет салона и хотела. До овог случаја је дошло јер је архитектура дизајнирана за строго један систем заказивања без модуларности и флексибилности за будућа проширења. Имплементација система за хотеле је захтевала значајне развојне активности, миграције података и измене у више слојева система (фронтенд, бекенд и база података), уместо једноставних локалних интервенција у коду.

4.2. Дизајнерски технички дуг

Један од примера појаве дизајнерског дуга су лоше програмерске праксе, попут превеликих класа. У случају *Petify* апликације, један од примера представља модел ентитета *Salon*. У почетку је овај модел садржао само основне информације. Међутим, како је број клијената растао, тако су расли и њихови захтеви о персонализацији и конфигурацији, попут дужина времена између термина, да ли дозвољавају резервације данас за данас, или отказивање један или више дана унапред. Ови захтеви су решени додавањем већег броја атрибута попут: *canReserveTomorrow*, *canCancelTomorrow* и *timeSlotRange,* што је резултовало значајним проширењем модела. До овог дуга не би дошло да смо креирали табелу за конфигурацију (*CustomSettings* са пољима *timeSlotRange, canOperateAfterWorkingHours, reservationPolicy*) чиме се правила изолују од основних атрибута салона.

4.3. Дефектни технички дуг

Дефектни технички дуг указује на познате грешке (дефекте) у софтверском систему које су уочене током тестирања или пријављене од стране корисника и чије се решавање често одлаже због ограничених ресурса и конкурентских приоритета унутар пројекта.

Petify апликација је имала више различитих дефеката у свом систему. Један од њих је неуспешна промена радног времена салона, чије се решавање проблема одлагало, јер не представља функционалност која се често користи, док би њено решавање захтевало значајно улагање времена у програмирању. Проблем је кулминирао и порастао када смо додали опцију нових запослених у салону и креирање и измене њиховог радног времена које су користиле логику из салона. Овај случај је могао бити избегнут, да је након прве пријаве грешке, пронађено и имплементирано решење у коду.

5. РЕЗУЛТАТИ И ДИСКУСИЈА

Анализа техничког дуга на примеру система за заказивање термина показала је да се понављају одређени обрасци појављивања техничког дуга. Ово је честа појава код брзо растућих пројеката који имају постављене рокове, налазе се у раним фазама позиционирања на тржишту и интензивно комуницирају са клијентима. У случају *Petify* апликације, најизраженији облици техничког дуга били су архитектонски, дизајнерски и дефектни. Свака од ових појава била је директно повезана са променама захтева, било да је реч о повећању броја клијената, додатним захтевима од стране корисника и менаџмента или проширењу домена апликације на нове типове услуга.

Овакав образац имплицитно указује на технички дуг у захтевима (*RED*) као иницијални фактор који подстиче каснији развој осталих категорија техничког дуга. Непрецизно дефинисани захтеви често доводе до импровизованих архитектонских решења, повећања сложености модела и појаве дефеката у каснијим фазама развоја. Претходне студије потврђују да непотпуни захтеви представљају водећи симптом *RED*-а, док је незадовољство корисника најчешћа последица [8,9]. Ниски спољашњи квалитет софтвера, који се често манифестује кроз корисничко незадовољство, једна је од кључних последица документационог и захтевски условљеног техничког дуга [9]. Ови налази су у складу са анализом система *Petify*, где су проблеми настали у фази дефинисања захтева касније резултирали значајним архитектонским изменама, сложенијим дизајном и повећањем броја дефеката.

Додатно, значајан фактор који је утицао на појаву техничког дуга био је временски притисак условљен појавом конкурената и притиском тржишта за бржим завршетком пројекта. Под утицајем кратких рокова и брзим пласирањем производа, развојни тим је правио намерно краткорочне компромисе у дефинисању и разради захтева, што је касније резултовало дугорочним последицама на архитектуру и код. Овај сценарио је у складу са налазима претходних истраживања, која указују да временски притисак и тржишна конкуренција представљају једне од кључних фактора у генерисању *RED*-а и његовом преношењу на остале техничке слојеве система [6].

6. ЗАКЉУЧАК

Тема техничког дуга је од суштинског значаја за разумевање квалитета софтверских пројеката, нарочито у раним фазама развоја када се системи брзо мењају и прилагођавају тржишту. У овом раду анализиран је технички дуг на примеру система за заказивање термина, са фокусом на *Petify* апликацију. Циљ је био да се идентификују и разумеју обрасци појављивања различитих типова техничког дуга и да се они повежу са теоријским концептима и факторима који на њих утичу.

Резултати анализе показали су да су најизраженији облици техничког дуга у *Petify* систему архитектонски, дизајнерски и дефектни дуг, а да су промене и недовољна разрада захтева биле кључни покретачи њиховог настанка. Тиме је потврђено да технички дуг у захтевима често представља иницијалну тачку која води ка проширењу других категорија техничког дуга. Додатно, временски притисак и тржишна конкуренција значајно су утицали на одлуке које су генерисале дуг, што је у складу са налазима претходних студија.

Ови резултати наглашавају потребу за систематичним праћењем и управљањем техничким дугом кроз цео развој софтвера. Посебан акценат треба да буде на квалитетно дефинисаним захтевима, чиме се може значајно смањити појава нових облика техничког дуга током развоја. Поред традиционалног фокуса на дуг кода, резултати указују на значај препознавања и анализирања осталих врста техничког дуга, посебно оних који често имају дугорочне последице по одржавање и развоју система. У будућности, од великог значаја би био развој специјализованих алата који би пратили све појаве техничког дуга и одређивали приоритете за ефективно решавање и унапређење софтвера и система.

7. ЛИТЕРАТУРА

[1] W. Cunningham, "The WyCash Portfolio Management System," ACM SIGPLAN OOPS Messenger, vol. 4, no. 2, pp. 29–30, 1992.
[2] N. S. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, "Identification and Management of Technical Debt: A Systematic Mapping Study," Information and Software Technology, vol. 70, pp. 100–121, 2016.
[3] C. Seaman, Y. Guo, N. Zazworka, F. Shull, C. Izurieta, and A. Vetrò, "Towards an Ontology of Terms on Technical Debt," in Proceedings of the IEEE International Workshop on Managing Technical Debt (MTD), 2014.
[4] M. Fowler, "Technical Debt Quadrant," *Bliki* [Blog], 2009. [Online]. Доступно: http://www.martinfowler.com/bliki/TechnicalDebtQuadrant.html (приступљено 10.10.2025.)
[5] S. McConnell, *Managing Technical Debt*, Construx Software Builders, Inc., White Paper, 2008. [Online]. Доступно: https://www.construx.com/uploadedfiles/resources/whitepapers/Managing%20Technical%20Debt.pdf (приступљено 10.10.2025.)
[6] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, and F. Arcelli Fontana, "A Systematic Literature Review on Technical Debt Prioritization: Strategies, Processes, Factors, and Tools," Journal of Systems and Software, vol. 171, p. 110827, 2021.
[7] D. Koulla Moulla, E. Mnkandla, H. Oumarou, and T. Fehlmann, "Technical Debt Measurement: An Exploratory Literature Review," CEUR Workshop Proceedings, vol. 3852, pp. 1–14, 2024.
[8] A. Melo, R. Fagundes, V. Lenarduzzi, and W. Santos, "Identification and Measurement of Technical Debt Requirements in Software Development: A Systematic Literature Review," arXiv preprint arXiv:2105.14232, 2021.
[9] J. Frattini, D. Fucci, D. Mendez, R. Spinola, V. Mandic, N. Tausan, M. O. Ahmad, and J. Gonzalez-Huerta, "An Initial Theory to Understand and Manage Requirements Engineering Debt in Practice," arXiv preprint arXiv:2211.06189, 2022.

Кратка биографија:

Милица Окиљевић рођена је у Урошевцу 1999. год. Дипломски рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењене рачунарске науке и информатика одбранила је 2021.год.
Контакт: milica99okiljevic@gmail.com