Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.85
ДОИ: https://doi.org/10.24867/35BE05Mirkovic

Развој подршке за парсирање за SeamlessMDD развојни оквир уз ослонац на Web API

Development of parsing support for SeamlessMDD framework based on Web API

Драган Мирковић, *Факултет техничких наука, Нови Сад*

Студијски програм - СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *Овај рад представља проширење SeamlessMDD оквира за развој подршком за парсирање путем удаљених Web API сервиса. Циљ је био избећи јаку зависност од библиотека за парсирање, уколико би се оне укључиле у језгро алата. Предложени приступ дефинише апстрактни интерфејс IApiParser и двије имплементације, тако да клијент изражава намјеру у доменским појмовима, а конкретно извршење обавља удаљени сервис за парсирање. Рјешење смањује зависност, јасно раздваја одговорности и омогућава лаке замјене технологија, што је показано кроз двије имплементације, AdvancedHTMLParser и lxml, које се интегришу без измјена у клијентском коду. На тај начин SeamlessMDD добија чистију архитектуру и подршку за проширивост новим парсерима.*

Кључне речи (три до пет): *развој базиран на моделима (MDD), SeamlessMDD, парсирање, Web API*

Abstract – *This paper presents an extension to the SeamlessMDD framework that introduces support for parsing via remote Web API services. Our goal was to minimize dependence on parsing libraries that would otherwise be included in the tool's core. The proposed approach defines an abstract interface (IApiParser) along with two implementations, allowing the client to express intent in domain terms while a remote parsing service carries out the execution. This solution reduces coupling, clearly separates responsibilities, and facilitates easy technology swaps, as demonstrated through two implementations, with AdvancedHTMLParser and lxml, that integrate without requiring changes to the client code. In this way, SeamlessMDD achieves a cleaner architecture and supports extensibility with new parsers.*

Keywords: (three to five): *Model-Driven Development (MDD), SeamlessMDD, parsing, Web API*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић, ред. проф.

1. УВОД

Развој вођен моделима (Model Driven Development – MDD) има за циљ да тежиште рада пребаци са имплементационих детаља на доменске концепте, тако што модел постаје извориште из којег се аутоматски изводе код, конфигурације и други артефакти. Тај приступ подразумијева јасно дефинисане трансформације улазних на излазне елементе, како би се промјене у моделу имплементирале као минимални, контролисани захвати у излазном коду, без губитка ручно писаног кода и уз предвидљиво понашање током еволуције система **Error! Reference source not found.**. SeamlessMDD је алат за MDD развој који проблем интеграције генерисаног и ручно писаног кода адресира инкременталним, селективним трансформацијама модела на програмски код и очувањем ручно писаних измјена. Његов циљ је неометан рад програмера кроз подршку за директно спајање ручно писаног и генерисаног кода у оквиру исте датотеке и мијењање од стране алата само оних дијелова кода захваћених промјеном модела **Error! Reference source not found.**5,6]. То се постиже API базираним генераторима кода који раде над синтаксним стаблима, коришћењем парсера за језике на којима ће се вршити развој циљног решења.

Циљ овог рада је да се у SeamlessMDD алат уведе подршка за парсирање за различите програмске језике, уз избјегавање јаке зависности између језгра алата и библиотека/сервиса за парсирање, што би отежало замјену технологије и тестирање. Из тог разлога уводимо механизам за парсирање кода употребом удаљених API-ја, тако да алат говори једним доменским „језиком намјере“, а парсирање и операције изводе одвојени сервиси **Error! Reference source not found.**. Конкретно, у овом раду представљамо проширење које уводи апстрактни интерфејс за API парсирање и измијешта манипулацију стаблом у засебне удаљене сервисе, чиме се поједностављује архитектура и омогућава лакша употреба различитих парсерских библиотека **Error! Reference source not found.**.

Структура рада је сљедећа. У другој секцији представљамо SeamlessMDD развојни оквир и његов начин рада. У секцији 3 представљамо дизајн нашег рјешења и дискутујемо пројектантске одлуке. Секција

4 даје детаље имплементације а секција 5 показује како се додаје нови парсер без промјена у клијентском коду. Секција 6 је закључак рада, гдје резимирамо постигнуто и дефинишемо правце будућег развоја.

2. SEAMLESSMDD ALAT

SeamlessMDD у средиште ставља проблем који се у пракси MDD-а упорно враћа: како синхронизовати генерисани код са ручним измјенама, тако да крајње рјешење ради и да се те измјене очувају кроз будуће циклусе генерисања [6]. Умјесто трансформација базираних на шаблонима које приликом промене модела поновно генеришу све артефакте, SeamlessMDD инсистира на инкременталним, селективним трансформацијама. Мијења се само релевантни дио циљног кода на основу промењених елемената модела, чиме се смањује вријеме извршавања и ризик од конфликата [6]. Ова идеја „source-preserving incrementality“ поткријепљена је и у литератури о инкременталном model-to-text превођењу, с јасном препоруком да обим пропагиране промјене буде пропорционалан само утицају промјене, а не величини улазног модела [6].

Да би такво понашање било могуће, архитектура се ослања на два основна појма: „делту“ и интерну репрезентацију артефакта (Internal Artifact Representation, IAR). Делта је минималан и прецизан опис промјене у моделу (шта је додато, уклоњено или измијењено), док је IAR структуриран приказ постојећег излазног артефакта (нпр. HTML/код) у облику стабла са типовима, идентификаторима и контекстом. На основу делти се граде циљане операције над IAR-ом (додај, замијени, обриши конкретан чвор или атрибут), па се тек затим IAR серијализује у текст излазне датотеке. Тако се не преписује цијела датотека него се локално захвата само њен дио захваћен променом елемента модела. Када артефакт не постоји, генератор га ствара из шаблона. Ако постоји, парсира се постојећи садржај и претвара у синтаксно стабло са контекстом (IAR) над којим API-базирани генератор додаје, мијења или брише чворове [6]. Овим се циљна датотека третира као структура, не као обичан текст, па су и интервенције прецизније.

На основу овог приступа, SeamlessMDD у средиште поставља процес који је усмјерен на мале, провјерљиве кораке. Прво се детектују промјене на моделу путем поређења тренутне и претходне верзије, за генератору релевантне дијелове. Резултати тих промјена су делте (додато/обрисано/мијењано) на нивоу елемента модела. Затим слиједи валидација која користи API базиране генераторе да провјере синтаксну исправност намјераваних захвата над IAR-ом и да семантички испитају усклађеност са моделом. Тек послије тога долазимо до корака извршавања трансформације. Трансформација обухвата функције: убаци, модификуј и обриши. Извршавање се врши над IAR-ом, а корисник може да бира између два режима рада: „Preview“ (увид у планиране измене у коду без серијализације) и „Generate“ (трајна серијализација промјена), при чему се ослања на VCS - систем за контролу верзија који омогућава евиденцију измјена, поређење стања и повратак на претходне ревизије [6].

Ова подјела на преглед и упис значајно смањује проблеме при усвајању MDD-а, јер се ефекти могу испробати и кориговати прије него што постану трајни. Поред овога, за SeamlessMDD је кључно да се омогући ко-еволуција модела и кода, односно њихов паралелан и усклађен развој: промјене у моделу доводе до минималних, циљаних измјена у коду, док се ручно унесене измјене у коду очувају и узимају као приоритет при наредним циклусима. Ово се постиже мапирањем елемената модела на кодне фрагменте, провјерама интегритета фрагмената и интерактивним рјешавањем конфликата на нивоу елемената модела, а не линија текста [6]. Ово има за циљ да подстакне употребу овог алата и у системима гдје се рад врши „по дијеловима“ и гдје је само дио система моделован (остало се наставља развијати ручно), што омогућава постепено усвајање MDD-а. Веома је важно и да рјешавање конфликата не буде на нивоу линије како VCS-ови функционишу (поређењем редова обичног текста који занемарује контекст), већ да алат прикаже конфликт у контексту модела и понуди смислене опције за његово разрешавање. Такође, обим трансформација треба да се смањи колико је могуће, да би могућност настајања конфликта била мања.

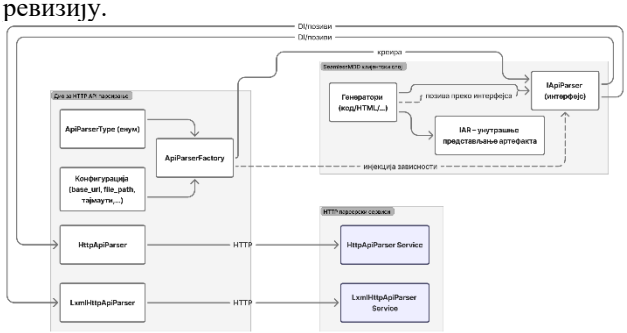
Разлог зашто озбиљни напори у SeamlessMDD-у иду баш у овом правцу јесте дуго познат „round-trip“ проблем [12]. MDD природно уводи више нивоа и више међусобно повезаних артефаката; кад промјена крене „одоздо“ (из кода), веома је тешко поуздано подићи апстракцију нагоре и пренијети значење у модел, а поновно генерисање има тенденцију да често изгуби ручно писан код. Отуда нагласак на IAR-у, на селективности, на приоритету ручних измјена и могућности увида у измјене, у Preview режиму.

3. ДИЗАЈН РЈЕШЕЊА

Анализирајући SeamlessMDD оквир, може се уочити да парсирање представља уско грло самог алата и да је чување њега у језгру алата потенцијалан проблем. Код за парсирање као дио језгра алата доводи до јаке зависности између генератора и конкретне библиотеке за парсирање. Свака промјена библиотеке, верзије или начина адресирања чворова у стаблу прелива се на клијентски код и на логичке токове генератора. Тестирање је скупо и подложно променама јер зависи од исте те библиотеке: тешко је изоловати неуспјехе, симулирати статусне кодове или провјерити понашање у граничним случајевима. Подршка више парсерских технологија значи одржавање паралелних грана кода или разгранавања по условима, што увећава технички дуг. Операције над стаблом (читања, уметања, замјене, брисања) постају уско везане за специфични DOM/AST модел, што отежава увођење другог алата са другачијим семантичким нијансама. Перформансе и стабилност су тешко мјерљиве јер се све дешава унутар истог процеса, без јасних мјеста за кеширање, временска ограничења или поновне покушаје. Оперативно, тимови не могу лако да изаберу најбољи парсер за свој домен, а да то не утиче на остатак система. Прелазак на удаљени сервис касније би свакако био нужан ради скалабилности, али тренутни спој парсирања и генератора то отежава. На крају,

додавање трећег или четвртог парсера тражи понављање шаблона и ризикује неконзистентност у обради грешака, формату одговора и уговора који клијент очекује. Све ово кочи еволуцију, повећава трошак одржавања и смањује повјерење у минималне, селективне трансформације које SeamlessMDD заговара.

Рјешење проблема је дизајнирано тако да централни дио алата не зависи од конкретне технологије парсирања, већ од стабилног и јасно дефинисаног интерфејса за API парсирање. Умјесто да клијентска апликација у себи садржи парсере за све језике и формате, парсирање и манипулација стаблима се изводе у посебном сервису изложеном преко програмског интерфејса IAPIParser (слика 1). Клијент и сервис комуницирају јасним уговором: „нађи елемент по идентификатору“, „провјери да ли чвор постоји у овом контексту“, „убаци подструктуру на дату путању“, „замјени елемент датим HTML-ом“. Клијент не мора знати ни којом је библиотеком стабло изграђено, ни како се интерно представља чвор. Тако се добија неколико директних користи. Прво и основно, могућност пружања подршке парсирања за много различитих структура података, те лакше мијењање употребљених библиотека за парсирање. Друго, скалабилност: сервиси се могу хоризонтално умножавати, распоређивати на локалну машину или у мрежу, а клијент се не мијења. Треће, бољи квалитет парсирања. Пошто сервис ради над стаблима, лакше је обезбиједити да су измјене минималне, да чувају ручне измене и да остављају чист траг за тестирање и ревизију.



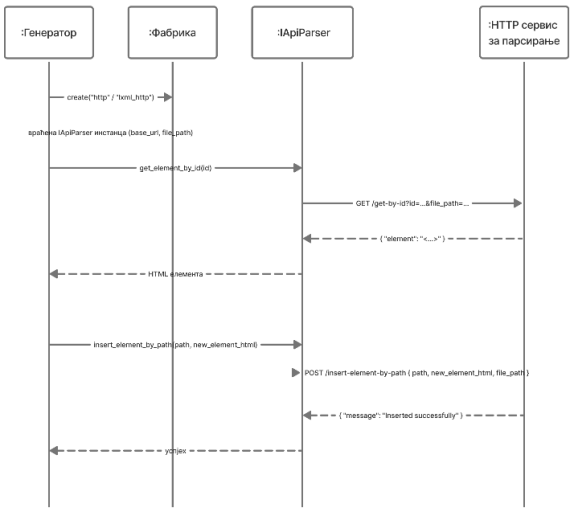
Слика 1. Комуникација компоненти за парсирање

Да би се остварила максимална добит, било потребно употријебити још додатних помоћних образаца, као што су уметање зависности (енг. *Dependency Injection*) и фабрика (енг. *Factory*). Клијент зависи само од апстрактног интерфејса парсера; конкретна имплементација се умеће приликом покретања или конфигурације. Фабрика, са друге стране, се брине и чува логику избора имплементације. То се остварује на основу типа, адресе сервиса или окружења, враћа се парсер који говори истим интерфејсом. Ова комбинација креира ниску зависност између клијента и имплементације и пружа „*hot-swap*“ могућност – лако је замијенити технологију парсирања без интервенција у клијентском коду или у остатаку алата. Уз то, тестови се пишу према структури интерфејса: једном дефинисан скуп понашања може се провјеравати над симулираним сервисима, локалним серверима или правим инстанцама у интеграционом окружењу.

Ово даје нову димензију SeamlessMDD оквиру, јер је много лакше пронаћи постојећи сервис парсирања и само се прилагодити његовом API-у, него њега интегрисати у наш код.

4. ИМПЛЕМЕНТАЦИЈА РЈЕШЕЊА

Имплементација прати дизајн кроз двије конкретне класе IApiParser интерфејса и два помоћна сервиса. Прва имплементација HttpApiParser комуницира са сервисом заснованим на AdvancedHTMLParser-у [9], а друга имплементација, LxmlHttpApiParser са сервисом на lxml-у [10]. Методе интерфејса мапирају се на HTTP руте сервиса за читање (нпр. GET /get-by-id, /get-by-name, /check-exists, /get-elements-by-path) и мутације (нпр. POST /insert-element-by-path, PUT /update-by-path, DELETE /delete-by-path, као и операције по ID-у). Одговори се нормализују у уједначене JSON облике (нпр. { "element": "<…>" } за читања, { "exists": true|false } за провјере, { "message": "…" } за успјешне мутације), тако да клијент не зависи од специфичности библиотека за парсирање. Обрада статуса је централизована: 200 означава успјех, 404 се тумачи као „не постоји“, 400 сигнализује лош улаз, 500 серверску грешку, 501 „није подржано“ (на примјер, AdvancedHTMLParser сервис не нуди све операције које пружа lxml варијанта). Конфигурација имплементација кроз фабрику прима параметре за подешавање комуникације, чиме се извршење лако прилагођава локалним или удаљеним окружењима. На овај начин, SeamlessMDD задржава исти начин рада (Preview/Generate над IAR-ом), док конкретно извршење обавља изабрани сервис са стандардизованим интерфејсом и понашањем (слика 2).



Слика 2. Добављање и упис елемената стабла

5. ДОДАВАЊЕ НОВОГ ПАРСЕРА

Архитектура је осмишљена тако да увођење новог парсерског сервиса не захтијева измјене у клијентском коду већ само локализоване допуне у слоју имплементација. Полазно, проширује се енум тип којим се бира имплементација и додаје се нова класа која реализује интерфејс IApiParser, прилагођена API-ју конкретног сервиса (путање, параметри, повратни формати). Конфигурација нове имплементације

прихвата адресу сервиса, путању до обрађиване датотеке и подешавања HTTP клијента, уз могућност додавања заглавља за аутентификацију/ауторизацију када је потребно. Фабрика се допуњава једном граном која нову вриједност енум типа мапира на нову класу, чиме избор парсера остаје централизован и транспарентан. При имплементацији метода интерфејса, сваку операцију треба досљедно пресликати на руте сервиса, те нормализовати одговоре у уједначене JSON облике које клијент већ очекује. Нови парсер може радити локално или удаљено, за SeamlessMDD је то прозирно, јер се клијент и даље обраћа само IApiParser-у и наставља свој Preview/Generate ток над IAR-ом. Практично, постојећи интеграциони сценарији се могу поново искористити над новом имплементацијом, што омогућава брзу провјеру усклађености са интерфејсом. На овај начин, увођење трећег или четвртог парсера своди се на предвидљив шаблон корака, без додира у генераторима и остатку система. Резултат је линија проширивости која омогућава да се технологија парсирања бира према доменским критеријумима, а да архитектура остане чиста и стабилна.

6. ЗАКЉУЧАК

У овом раду смо адресирали уочено уско грло у SeamlessMDD-у: чврсту спрегу између језгра алата и конкретних парсерских библиотека. Увели смо стабилну, замјењиву апстракцију за парсирање (IApiParser) и централизовали избор имплементације кроз Factory шаблон, тако да клијентски дио изражава намјеру у доменским појмовима, док конкретно извршење обавља сервис. На тај начин задржана је филозофија SeamlessMDD-а о минималним, селективним захватима над IAR-ом и приоритету ручних измјена, а истовремено је смањена зависност од технологија и поједностављено увођење различитих парсера. Двије изведене имплементације, на основу AdvancedHTMLParser-а и lxml-а, показују да исти сценарији могу да се изведу без измјена у клијентском коду, што директно потврђује циљеве о смањењу зависности ка библиотекама за парсирање, повећању лакоће тестирања и природној проширивости [6,7].

Ограничења тренутне верзије односе се прије свега на број подржаних сервиса и на оперативне аспекте мрежне комуникације. Нису уведени механизми временских ограничења и поновних покушаја, нити подршка за аутентификацију/ауторизација за заштићене сервисе. Такође, нису спроведена систематска мјерења перформансии. Будући рад усмјерићемо на додавање нових парсера (и за друге језике/формате), оснаживање комуникационог слоја, подршку безбједносним захтјевима, као и увођење метрика и трајног логовања за оперативну употребу у већим тимовима. Планирано је и проширење уговора за богатије семантике ажурирања елемената и експлицитну идемпотентност, као и формалнија валидација конзистентности између различитих имплементација. На тај начин, предложена архитектура остаје стабилна основа за даљу еволуцију SeamlessMDD-а ка ширем, индустријски одрживом еко-систему парсерских сервиса.

7. ЛИТЕРАТУРА

[1] B. Selic, “The Pragmatics of Model-Driven Development,” IEEE Software, 20(5), 2003.

[2] D. C. Schmidt, “Guest Editor’s Introduction: Model-Driven Engineering,” Computer, 39(2), 2006.

[3] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, 2nd ed., Morgan & Claypool, 2017.

[4] B. Hailpern, P. Tarr, “Model-driven development: The Good, the Bad, and the Ugly,” IBM Systems Journal, 45(3), 2006.

[5] B. Dragaš et al., “SeamlessMDD: Framework for Seamless Integration,” технички извјештај, GitHub: github.com/BojanaZ/SeamlessMDD, приступљено: окт. 2025.

[6] B. Dragaš, N. Todorović, T. Rajačić, G. Milosavljević, Ž. Vuković. "A Novel Approach to Integration of Manual Changes in Generated Code: SeamlessMDD." *Journal of Web Engineering* 24, no. 4 (2025): 499-528.

[7] N. Todorović, B. Dragaš, G. Milosavljević, “Supporting Integrative Code Generation with Traceability Links and Code Fragment Integrity Checks,” in Disruptive Information Technologies for a Smart Society, Springer, Cham, 2024.

[8] W3C, “XML Path Language (XPath) 3.1 (Recommendation),” w3.org/TR/xpath-31/, приступљено: окт. 2025.

[9] lxml, “Parsing XML and HTML with lxml,” lxml.de/parsing.html, приступљено: окт. 2025.

[10] AdvancedHTMLParser, “Project page (PyPI),” pypi.org/project/AdvancedHTMLParser/, приступљено: окт. 2025.

[11] B. J. Ogunyomi, L. Rose, D. Kolovos, “Incremental execution of model-to-text transformations using property access traces,” Software & Systems Modeling, 18:1–17, 2019.

[12] B. Hailpern, P. Tarr, „Model-driven development: The Good, the Bad, and the Ugly“, IBM Systems Journal, 45(3), 2006.

Кратка биографија:

Драган Мирковић основну и Гимназију „Васо Пелагић“ у Брчком завршио је као вуковац. Од 2019. студира на ФТН-у у Новом Саду (Софтверско инжењерство и информационе технологије). Основне студије окончао је 2023., а мастер студије уписује исте те године, смијер Софтверско инжењерство и информационе технологије, усмјерење Софтверско инжењерство.
Контакт: dmirkovic.se@gmail.com