Zbornik radova 49. Konferencije za ETRAN, Budva, 5-10. juna 2005, tom II
Proc. 49th ETRAN Conference, Budva, June 5-10, 2005, Vol. II

ПРЕНОСИВОСТ КОДА НА КОРИСНИЧКОМ ИНТЕРФЕЈСУ СЛОЈА УПРАВЉАЊА ЕЛЕМЕНТОМ МРЕЖЕ

Михаило Станић, Мирослав Илић, Бранка Лештар
ИРИТЕЛ, Београд

Садржај – *Приказано је једно решење преношења кода на корисничком интерфејсу у софтверу центра за управљање мрежом СУНЦЕ-М. Описано је првобитно решење на слоју управљања мрежом, на које је примењено рефакторисање што је претходило идентификовању преносивих делова кода. Потом је представљен реализовани генератор кода који омогућава добијање форми корисничког интерфејса.*

1. УВОД

Софтвер центра за управљање мрежом СУНЦЕ-М [1] омогућава управљање мешовитом транспортном мрежом, састављеном из *SDH* и *PDH* елемената. Његово коришћење обезбеђује са централног места непрекидан увид у стање мреже као и промену њеног рада.

Проблем комплексности управљања телекомуникационом мрежом решава организацијска архитектура *TMN*-а (*Telecommunications Management Network*) поделом функционалности управљања на већи број логичких слојева у облику хијерархијске структуре [2]. У таквој структури СУНЦЕ-М реализује функције слоја управљања мрежом *(Network Management Layer*) и слоја елемента мреже (*Element Management Layer*). Слој управљања елементима мреже омогућава управљање конфигурацијом елемента мреже, локализацију грешке унутар елемента и сл, док слој управљања мрежом обезбеђује управљање мрежом као целином.

TMN дозвољава да један мрежни елемент представља више уређаја. У верзији 4.4 СУНЦЕ-М омогућава рад са следећим типовима мрежних елемената:

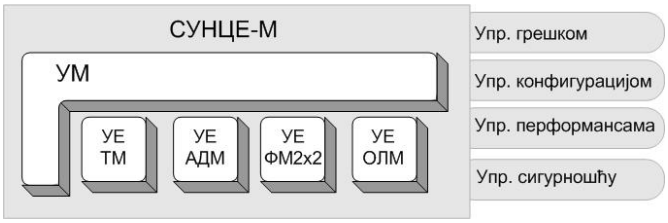
- ТМ (терминални мултиплексер *ODS155* ИРИТЕЛ)
- АДМ (*Add/Drop* мултиплексер *ODS155* ИРИТЕЛ)
- ФМ2х2 (флексибилни мултиплексер *FM2x2* ИРИТЕЛ)
- ОЛМ (произвољна комбинација следећих уређаја: оптички терминал *OTSM* ИРИТЕЛ, *HDSL* линијски терминал *LTH-E1* ИРИТЕЛ и конвертор интерфејса *G.703* у *X.21/V.35* - *KGXV* ИРИТЕЛ)

У току је развој нових фамилија уређаја чије управљање треба да се интегрише у СУНЦЕ-М: *ODS622* ИРИТЕЛ, *OTS622/155* ИРИТЕЛ, *FM8x2* ИРИТЕЛ. Да би се то омогућило треба омогућити интеграцију сваког од елемената на мрежном нивоу и реализовати слој управљања елементом мреже за сваки од њих.

Приликом додавања подршке на слоју елемента мреже у СУНЦЕ-М неопходно је извршити и реализацију графичког корисничког интерфејса. То укључује како визуелни дизајн, тако и писање кода који омогућава његову функционалност. Да би се овај временски захтеван задатак што брже извршио циљ је био издвајање у костур свих оних делова корисничког интерфејса у постојећем решењу који се могу користити приликом додавања подршке за нове фамилије уређаја. Стога је прво извршена анализа постојећег решења, начињена је измена која омогућава издвајање преносивих делова, а затим је направљен алат за прилагођавање специфичних делова кода.

2. АНАЛИЗА ПОСТОЈЕЋЕГ РЕШЕЊА

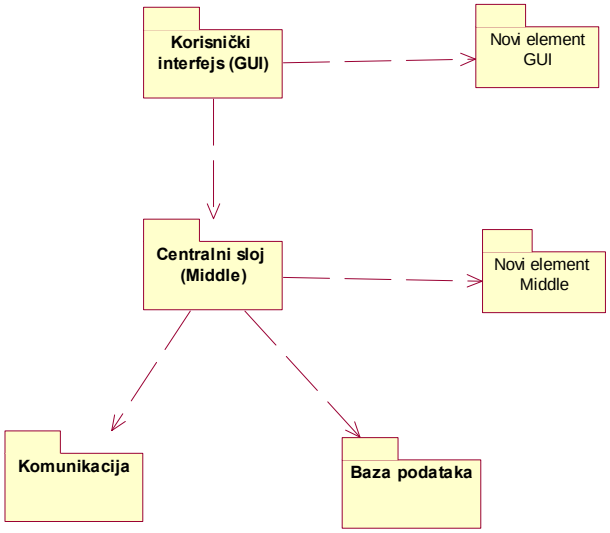
Организацијска и функционална архитектура *TMN-а* у софтверу СУНЦЕ-М дате су на слици 1. Ниво управљања мрежом (УМ) омогућава операције над параметрима који се тичу мреже, док ниво управљања елементом (УЕ) омогућава рад са сваким од елемената мреже. Рад са појединачним елементом омогућава изоловани подсистем, који се назива и елемент менаџер (*Element Manager*). Сагласно функционалној архитектури *TMN*-а [2], СУНЦЕ-М омогућава следеће функције управљања приказане на слици: грешком, конфигурацијом, перформансама и сигурношћу. СУНЦЕ-М реализовано је коришћењем алата *Visual Basic* и *Visual C++,* и извршава се под *Windows* оперативним системом.



Сл.1. *СУНЦЕ-М* - о*рганизацијска и функцијска арх.*

На слици 2 приказана је архитектура софтвера СУНЦЕ-М. Издвајају се пакети графичког корисничког интерфејса, средњег слоја, комуникационог дела и дела за рад са базом података. Графички кориснички интерфејс омогућава приказ комплетне мреже којом се управља, као и приступ појединачном мрежном елементу. Средњи слој распоређује поруке по осталим деловима система, одржава модел података о мрежи као целини и о свим појединачним уређајима у њој. Комуникациони део имплементира протокол за комуникацију [3] чиме се обезбеђује размена података са елементима у мрежи. У систему је реализован и приступ подацима преко базе.

Да би се омогућио рад на слоју управљања мрежом за било који нови елемент неопходно је додати по један пакет на корисничком интерфејсу и средњем слоју [4] – на слици пакети Нови елемент *GUI* и Нови елемент *Middle*. СУНЦЕ-М подацима о сваком елементу мреже приступа на истоветан начин, што је омогућено имплементацијом јединственог интерфејса према пакету на средњем слоју и пакету на корисничком интерфејсу.



Сл.2. *Архитектура софтвера СУНЦЕ-М*

Приликом израде корисничког интерфејса елемента мреже уведени су принципи представљања информација и у визуелном и у текстуалном облику. За овако представљене податке неопходно је написати код који омогућава њихов испис и контролу.

Визуелно приказивање требало је да омогући једноставан начин приказивања података који ће олакшати сналажење у њиховом мноштву. То је учињено тако што су уведени нивои представљања информација, па је иницијално доступан преглед исправности стања целина из којих је систем сачињен. Да би се ово реализовало систем је приказан у облику блокова који одговарају подели на логичке групе или функционалне јединице из којих се мрежни елемент састоји. Избором блока пружа се могућност увида у детаље његовог рада.

Код који омогућава представљање података треба да омогући приказивање података и њихову размену са моделом на средњем слоју. Без обзира на то што је писан у *Visual Basic*-у који је *RAD* (*Rapid Application Development*) алат, код на корисничком интерфејсу сваког елемента је средње величине (изнад 10,000 линија кода).

3. СКЕЛЕТ КОРИСНИЧКОГ ИНТЕРФЕЈСА

Приликом реализације компоненти у СУНЦЕ-М софтверском систему није било примарно омогућавање преносивости кода. Међутим, у постојећем решењу постоји могућност преношења приликом коришћења стандардних решења (визуелних, која се у коду третирају на исти начин) за приказивање особина уређаја, и требало је те делове кода издвојити и унапредити.

За софтверске системе важи да у фази одржавања расту и нарушава им се структура [5]. Узрок тога је додавање нових особина, и промена на постојећим, најчешће од стране оних који нису радили на првобитној концепцији решења. Због тога је као основа са које ће бити идентификован скелет одабран кориснички интерфејс за ОЛМ, за који је подршка последња додата у СУНЦЕ-М. Измене којима је добијен преносив код засноване су на техници рефакторисања. Рефакторисање је процес мењања софтверског система тако да се не мења спољно понашање кода, већ да се побољшава унутрашња структура. Његов циљ је унапређење већ написаног пројекта.

Циљ рефакторисања био је да се издвоји директно преносив код, код који је могуће пренети уз модификацију, и код који је неопходно поново писати. У току измене кода коришћени су шаблони из каталога рефакторисања [5]. Коришћењем скупа шаблона за организовање података извршена је промена структуре постојећег кода и уведен образац команде (*Command pattern*) [6]. Тиме је омогућено да се захтеви извршавају на униформан начин без обзира на то на ком делу корисничког интерфејса су захтевани. Класа *clsFormCommandPattern* која омогућава овај образац приказана је на слици 3 (приказани су само јавни чланови).

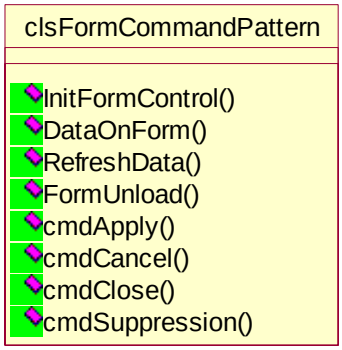
Инстанца класе *clsFormCommandPattern* постоји на свакој форми, и омогућава попуњавање контрола на њој, преузимање промена на контролама, као и реакцију на притисак на стандардну командну дугмад. Приватни чланови класе *clsFormCommandPattern* су објекти који представљају стање контрола на форми. Њихово постојање потребно је због могућности одустајања од измена на форми, и тада се стање контрола враћа у првобитно из ових објеката. Написане су класе из којих се ови објекти инстанцирају и повезују са унапред предвиђеним контролама за приказивање аларма и конфигурационих параметара. Уколико се не користе унапред предвиђена решења у приказивању података, неопходно је да се напише подршка за повезивање објеката са употребљеним контролама. Ова подршка подразумева измену контрола на основу стања у објектима, и обрнуто, измену стања објеката на основу контрола.

Класа има следеће јавне методе:

- *InitFormControl()* – омогућава креирање објеката којима се представљају контроле на форми, након чега се врши повезивање контрола са одговарајућим објектима; у позиву овог метода наводи се за сваку контролу да ли се освежава приликом измене на средњем слоју
- *DataOnForm()* – омогућава да подаци из модела уређаја на средњем слоју буду приказани на контролама и додељена стања објектима који их представљају
- *RefreshData()* – омогућава освежавање контрола и стања објеката који их представљају након њихове измене на средњем слоју
- *FormUnload()* – приликом затварања форме бришу се сви објекти који се спрежу са контролама
- *cmdApply()* – омогућава акцију на притисак дугмета за прихватање начињених измена; преузимају се стања са контрола на форми и врши се њихово форматизовање за слање на средњи слој

- *cmdCancel()* – омогућава акцију на притисак дугмета за поништавање измена на форми; контроле се враћају у стање приликом отварања форме или последње измене
- *cmdClose()* – омогућава акцију на притисак дугмета за затварање форме; дозвољено је затварање форме уколико подаци на њој нису промењени
- *cmdSuppression()* – омогућава потврђивања аларма повезано са притиском дугмета на форми

У оваквој организацији корисничког интерфејса карактеристична функционалност елемента мреже потпуно је енкапсулирана на форми на којој је имплементирана. При томе, начин интеракције остаје конандни образац. Тиме се знатно олакшава и додавање нових форми у систем, јер да би га испоштовала свака од њих треба да организује податке, визуелне елементе и начин рада на унапред одређени начин.



Сл.3. *Класа која имплеметнира командни образац*

Последица рефакторисања било је и смањење броја линија кода са 13604 на 9659, односно за 29%. Како је један од циљева рефакторисања издвајање кода који се понавља, могуће је смањење броја линија, али оволики обим није био очекиван. Пре свега је последица увођења командног обрасца, јер је свака форма подржавала описану функционалност за коју је издвојен скуп генеричких класа. Да би се нови концепт проверио, све направљене измене укључене су у кориснички интерфејс ОЛМ-а.

Из рефакторисаног кода, издвојени су:

- код који се директно може пренети у нови пројекат; представља све класе и форме које имају истоветне функције за све типове уређаја (око 40% кода за ОЛМ)
- код који се може користити уз неопходне измене; на пример, форма која приказује модул елемента мреже у коме се налазе функционалне јединице из којих је састављен (око 7% ОЛМ кода)
- део кода који се мора поново писати– све форме које дефинишу понашање карактеристично за елемент мреже (око 53% ОЛМ кода)

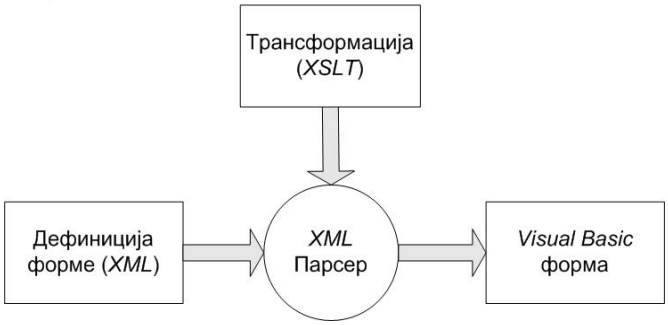
Премда су добијени добри резултати у преносивости кода, поставило се питање може ли се извршити додатно олакшање програмирања формализацијом начина додавања нових форми. Пошто су принципи за то били строго дефинисани, и обзиром на постојећа искуства са генераторима кода [7], дизајниран је генератор кода за форме карактеристичне за елемент мреже.

4. ГЕНЕРАТОР КОДА

За имплементацију генератора кода примењено је решење засновано на *XML* технологији. Процес добијања *Visual Basic* форми приказан је на слици 4, и састоји се из два корака.

Први корак је дефинисање функционалности форме. У оквиру њега се дефинишу компоненте које треба да се нађу на форми и постављају се у одговарајући хијерархијски распоред. Формални запис функционалности даје се у облику *XML* датотеке који је и крајњи производ ове фазе генерисања кода. Могуће је ово извршити у едитору текста, специјализованом *XML* едитору, или посебно написаној апликацији.

У другом кораку на дефиницију функционалности у *XML* формату примењује се *XSLT* трансформација са циљем добијања исправног *Visual Basic* кода. Овај корак своди се на учитавање дефиниције и трансформације у неки *XML* парсер и позивање функције која је извршава. Трансформација се може тумачити и као предложак (*template*) за стандардну *Visual Basic* форму из софтвера СУНЦЕ-М.



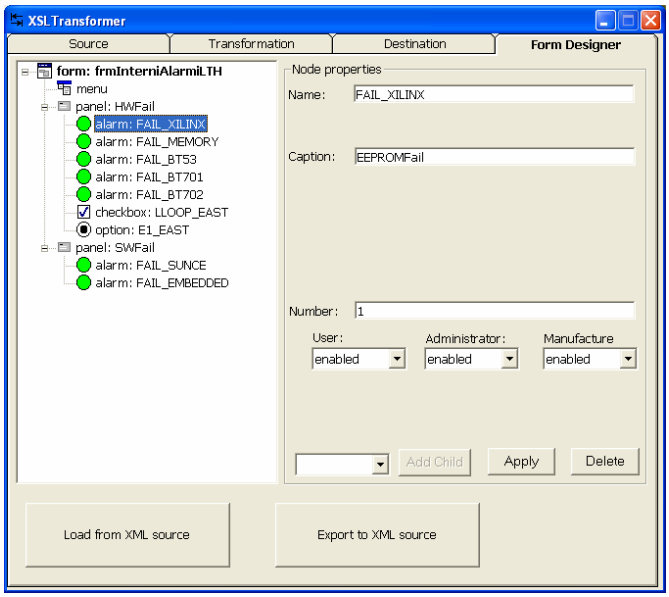
Сл.4. *Принцип генерисања кода*

За дизајнирање форми приликом развоја софтвера СУНЦЕ-М написана је посебна апликација која омогућава брзо и прегледно распоређивање потребних елемената. Корисник има контролу над комплетним процесом генерисања кода. Изглед њеног корисничког интерфејса приказан је на слици 5.

На слици могу да се примете четири картице. Прва (*Source*) омогућава да се погледа *XML* код датотеке над којом се врши трансформација. Друга картица (*Transformation*) омогућава да се види код *XSLT* трансформације, док трећа (*Destination*) омогућава генерисање кода и његов преглед. На четвртој картици (*Form Designer*) је визуелни дизајнер форми, који је детаљније приказан на слици 5.

На левој страни налази се стабло у коме је приказана хијерархијска организација у којој су чворови стандардни начини приказивања на форми (у примеру су наведени аларми, и конфигурациони подаци који су приказани преко контрола за потврду (*CheckBox*) и избор (*Option Button*)). Овде се приказују и нестандардни елементи на форми, са тим да је програмер дужан да

напише код који их подржава. На десној страни екрана су одлике које описују сваки од чворова у стаблу (у примеру са слике, за аларме је то назив, испис, број појављивања, приступ различитим нивоима корисника). У дну су дугмад која омогућавају снимање (*Export to XML source*) или учитавање (*Load from XML source*) припремљене конфигурације у *XML* датотеку.



Сл.5. *Кориснички интерфејс генератора кода*

У зависности од сложености елемента мреже, део кода са формама карактеристичним за елемент мреже може бити и већи од 53%, али се овим решењем највећи део тог кода добија из генератора. Предности тога су добијање провереног кода и да креативност може да се усредсреди на нестандардна решења уместо на поступке који се понављају.

5. ЗАКЉУЧАК

Следи дискусија резултата занимљивих са становишта рефакторисања кода, затим особина костура са преносивим кодом за нове софтверске пројекте, и дизајна генератора кода.

Премда су примењени само на једном софтверском пројекту, па недостаје поређење са другим пројектима, поједини резултати рефакторисања могу бити занимљиви за било који софтверски пројекат. Његовом применом код је потпуно реорганизован, уведен је командни образац као основа на коју се ослања рад корисничког интерфејса, и скраћен је код за 29%. Оволико скраћивање кода није било очекивано, а тумачење оваквих резултата је да су последица доследности у реализацији првобитног решења, што је и олакшало увођење командног обрасца. На основу овог искуства може се добити индикација до када је рефакторисање «исплативо» - све док промене на систему нису у значајнијој мери увеле измене у начину на који су слични проблеми третирани. Уколико у коду постоји исувише измена које их на различити начин обрађују једноставније је поновно писање кода уместо рефакторисања.

Стварањем преносивог костура издвојене су три групе кода: директно преносиви код у нови пројекат, код који се добија генератором кода и код над којим је потребна мануелна интервенција. На тај начин се од почетка рада на новом пројекту прецизно зна којим етапама у развоју софтвера треба да се посвети пажња, и за шта треба се обучи онај који ће костур користити.

Реализовани генератор кода заснован је на *XML* технологији. Он омогућава једноставно прављење нових форми, са њиховом пуном функционалношћу. Потребне интервенције на форми која се њиме добија зависе од начина представљања података на корисничком интерфејсу, односно могу бити непотребне уколико задату операцију могу да испуне предефинисани начини описа података.

Примена овог решења значајно убрзава креирање корисничког интерфејса за нови елемент мреже. Описани костур кода није само на корисничком интерфејсу елемента ОЛМ са кога је издвојен, већ се користи за развој корисничког интерфејса за уређаје *FM8x2* ИРИТЕЛ, *ODS622* ИРИТЕЛ и *OTS622/155* ИРИТЕЛ.

ЛИТЕРАТУРА

[1] М. Станић и др."Неке особине софтвера СУНЦЕ-М на нивоу слоја управљања мрежом", IX ТЕЛФОР, 20-22. Новембар 2001, Београд, стране 91-93

[2] М. Станић "Софтверски систем за надзор и управљање елементом транспортне телекомуникационе мреже", магистарски рад, ЕТФ Београд, 2002

[3] М. Станић "Интеграција на вишим нивоима управљања мрежом заснована на мобилним агентима", XLVI ЕТРАН, 5.-8. Јун 2003, Херцег Нови, свеска II, стране 135-138

[4] М. Станић и др. "Архитектура софтвера центра за управљање мрежом SDH и PDH уређаја - СУНЦЕ-М", XLVI ЕТРАН, 4.-7. Јун 2002, Бања Врућица - Теслић, свеска II, стране 99-102

[5] Мартин Фаулер "Рефакторисање – побољшање дизајна постојећег кода", ЦЕТ, 2003

[6] Erich Gamma et al. "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1995, pp. 233-242

[7] М. Станић "Генератор кода за моделовање аларма телекомуникационих уређаја", XLIV ЕТРАН, 26.-29. Јун 2000 Сокобања, свеска II, стране 67-70

Abstract – Presented is a solution for code reuse in the SUNCE-M's user interface for network operation and management centres. The existing solution for the element management layer is described, followed by an outline of how this code was refactored before reusable code was identified. Following this is a discussion of the development of a generic code generator that produces user interface forms.

CODE REUSE IN USER INTERFACE OF ELEMENT MANAGEMENT LAYER

Mihailo Stanić, Miroslav Ilić, Branka Leštar