Метод за автоматизирано програмиране на роботи Описание на метода

Иван Станев

Method for Automated Programming of Robots – Method description: *The paper describes the main design principles of the method, as well as its specification techniques, phases, user roles, and grouped in phases actions. A description of the Module for Automated Programming components Is done.*

Key words: *Software Engineering, Automated Programming, CASE, Service Robots, Robot Programming.*

ВЪВЕДЕНИЕ

Анализът направен в [1] показва, че има много на брой, разнообразни и с добри характеристики инструментални среди за програмиране на роботи. Това, което обаче също се вижда е, че в повечето от разгледаните ИСПР липсва добре развит инструмент за автоматизация на програмирането, който да отговаря на изискванията (И.х), посочени в колона Изисквания на Фиг. 1.

Тук ще бъдат разгледани принципите на проектиране и същността на **Метода за автоматизирано програмиране на роботи** (**МАПР**), както и принципа на работа на **Модул за автоматизирано програмиране (МАР[1])**. МАР е разработен като инструмент за прилагане на МАПР (И.4, И.5), чиято архитектура позволява използването му като компонент към различни, вече съществуващи ИСПР (И.6) и осигурява генерирането на програмен код, чието бързодействие при работа в реално време се влошава незначително вследствие на автоматизацията (И.7). МАР трябва да позволява също използването на компоненти и услуги за многократна употреба (И.3), които са задоволително стандартизирани (И.2) и насочени към тясна предметна област (И.1).

1. Принципи на проектиране на МАПР[1]

Основните принципи за проектиране на МАР (колона Принципи на проектирането – ПП.х от Фиг. 1) са както следва: **(1)** Ще бъде избрана **неформална естественоезикова спецификация** (ПП.5) поради факта, че тя позволява да се постигне без специално допълнително обучение на оператора платформена и езикова независимост при непълно и неточно описание на потребителската задача и при задоволителна степен на автоматизация на продукта. **(2)** Чрез избраните **три нива на спецификация** (ПП.7) - неформално, формално–платформено независимо и формално–платформено зависимо се цели постигането на по-добра платформена и езикова независимост, както и по-добро бързодействие на продукта в реално време. Платформената независимост се постига на второто ниво, където неформалните естественоезикови текстове се преобразуват с платформено и езиково независими конструкции, които имат в различните системни библиотеки образи на различни езици (обикновено най-масово разпространените езици за програмиране и по правило компоненти и услуги на конкретния или външни производители). На това ниво се извършва симулация на генерираните алгоритми, при което те се изчистват от грешки неточности или многозначност на спецификацията. След това получените конструкции се превеждат на платформено зависим машинен език и се изпълняват вече при добра скорост и добро ниво на оптимизация на кода в реално време. **(3)** Стандартизация на разработките се постига с два основни инструмента – чрез **използване на стандартни речници** (ПП.1 - обикновено комбинаторните речници на официалния за производителя език) и чрез разработване на **стандартни модели на предметната област** (ПП.2).

[1] MAP – Module for Automated Programming.

Именно за да се съкрати времето за разработка и ефективността при експлоатация на крайния продукт се прибягва до използването на тясна предметна област. Чрез стандартизираните модели се получава стандартизиране на използвания код, независимо от спецификацията. Освен това при запазване на предметната област се улеснява многократната употреба на вече разработени компоненти и услуги.

Фиг. 1 Матрица на изискванията към МАР

(4) Чрез въвеждането на **алгоритми за евристично търсене на решението** (ПП.6) се постига задоволителна автоматизация на разработката на софтуерни елементи и се подобрява бързодействието при работа в реално време, без това да води до преструктуриране на предварително генерирания код и същевременно без да нарушава свободата и гъвкавостта на обработващия алгоритъм. Рискът да бъде пропуснато при евристичното търсене съществуващо решение се избягва, чрез допълнително забавяне, на работата в реално време като ако евристичния алгоритъм не намери решение има възможност да бъде стартирано и пълно претърсване, което обикновено или намира решението или констатира неговото отсъствие (без значение дали защото решението липсва или защото попадаме в безкраен клон). **(5)** Въвеждането на **работата с размити понятия** (ПП.3) и използването на **системи за обработка на знания** (ПП.4) позволява също да е подобри ефективността при проектирането на софтуера като се откриват допълнителни възможност да се намери точно решение на задачата при непълна и неточна спецификация. Това обикновено става като чрез използването на лингвистичен процесор знанията се извличат от естественоезиковия текст, след което се позиционират на правилите места в модела на света и в случаи на непълноти и неточности или автоматично или с допълнителна намеса на оператор (значително по-рядко) се достига на задоволително крайно решение. **(6)** Чрез въвеждането на **софтуерна архитектура, която представлява комбинация от СОЗ и АОУ** (ПП.4) се постига запазване на отличните качества на СОЗ – като удобен инструмент за автоматизирано програмиране и на АОУ – като инструмент, осигуряващ гъвкавост и възможности за реконфигуриране на генерирания софтуер и едновременно с това подобрява интегрирането на собствени и външни разработки.

2. Същност на МАПР

Методът за автоматизирано програмиране на роботи е създаден за работа в предметна област с предварително дефиниран модел (примерно формален модел, семантична мрежа, онтология, формална граматика, речници и други).

При изпълнението на метода софтуерът, който трябва да бъде генериран се специфицира чрез естественоезиков текст. От тази спецификация МАР, като използва предварително разработените формални модели, правила и знания за предметната област, генерира размита програма, която на втора стъпка се

преобразува в детерминирана, а на трета стъпка се получава машинно зависима програма, съответстваща на началната спецификация.

Дефинирането на модела и знанията в предметната област (ПО), както и описанието на решаваната задача се извършват като се използват два типа техники за специфициране: **(1) формални** – с които обикновено се конструират базата знания, моделите и при необходимост бизнес процесите на предметната област; **(2) неформални** – най-често естествен език, с който (понякога непълно и/или неточно) се описва решаваната задача, алгоритъма за решаване на задачата (ако е необходимо) и правилата за интегриране на компоненти, услуги и модули в нови софтуерни продукти.

Дейностите по подготовката за генериране на софтуера са организирани в три фази на метода: **Ф1 предметно-независима фаза** – за създаване на предметно независимите операции и обекти, изграждане на изчислителните стратегии, за въвеждане на общите правила и ограничения за работа с инструмента, както и за създаване на други специфични предметно-независими знания; **Ф2 предметно зависима фаза** – където се създават операции, обекти, правила и ограничения на предметната област и допълнително се въвежда друга предметно зависима информация; **Ф3 фаза - задача** – в която се описват спецификацията на решаваната задача, свързаните с решаваната задача обекти, както и друга специфична задачно ориентирана информация.

Отговорни за реализацията на отделните фази при подготовката за генериране на софтуер са следните четири роли в метода: Р1 ИТ експерт, който отговаря за реализация на предметно независимата фаза, създава и поддържа инструментите (примитивни операции, обекти, стратегии за изчисление и други) на предметно независимо ниво, както и съдейства на Лингвиста и Бизнес експерта да изградят тяхната част от формалните модели и спецификации; **Р2 Лингвист**, който създава речниците, знанията за обработката на използвания естествен език, както и операциите за проверка на коректността, пълнотата и непротиворечивостта на извличаната от текста информация; **Р3 Бизнес експерт**, чиято задача е да създаде модела на предметната област, необходимите за работа на системата знания, както и да дефинира набора стандартни за предметната област обекти и операции; **Р4 Краен потребител**, който описва решаваната задача, описва решението (ако то не се получава автоматично), тества генерирания софтуер чрез симулация и в реално време и осигурява пускането на готовия софтуер в експлоатация.

Дейностите на метода се извършват в посочената тук последователност: **Д01 Изграждане на универсална ИТ компонента** – ИТ експертът избира компоненти, услуги и обекти, които представляват универсални за ИТ областта структури и действия, избира също стратегии за водене на изчислителния процес, правила за превключване от една изчислителна стратегия към друга, както и правила за интегриране на отделните елементи на търсеното решение. **Д02 Изграждане на универсална лингвистична компонента** – Лингвистът описва предметно независимите естественоезикови обекти и техните характеристики, както и допустимите синтактичните конструкции на изреченията от естественоезиковите текстове. **Д03 Изграждане на бизнес компонента на ПО** – Бизнес експертът изгражда модела на предметната област, дефинира основните бизнес обекти и бизнес операции, определя ограниченията в предметната област. **Д04 Изграждане на лингвистична компонента на предметната област** – Лингвистът избира речниковите статии и синтактичните конструкции подходящи за предметната област, извършва подходящите прагматични свивания (тоест съкращава броя на описваните характеристики) на речниковите статии до контекста на предметната област и подготвя езиковите правила, които ще се използват при провеждане на анализа на естественоезиковите текстове. **Д05 Изграждане на ИТ компонентата на ПО** – ИТ експертът анализира разработените вече бизнес и лингвистична компонента на ПО,

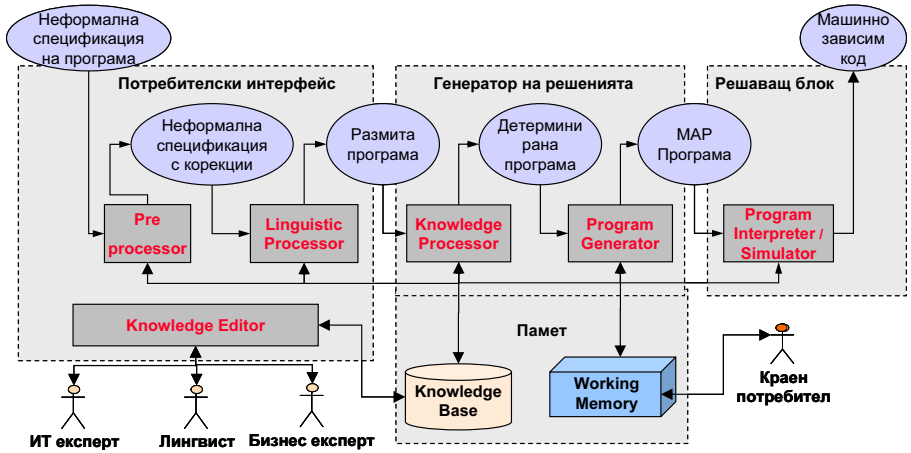
създава ИТ образите тези компоненти, създава необходимите връзки между тези компоненти, добавя предметно зависими ИТ компоненти и проверява за непротиворечивост формално специфицираните компоненти. **Д06 Задаване на решаваната задача** – Крайният потребител специфицира на естествен език изискванията към генерирания софтуер и задачите, които той трябва да решава. **Д07 Генериране на първа версия на заявения софтуер** – МАР обработва естественоезиковата спецификация, извършва анализ на спецификацията за непълноти и неточности (ако има такива), предприема действия по отстраняването им (ако разполага с необходимите за това знания) и минавайки през различни междини фази генерира машинно независим код за всички възможни решения, отговарящи на направената спецификация. В случай, че МАР не е успял да генерира кода издава съобщение с описание на причините, поради които не може да извърши тази дейност. **Д08** (**незадължителна**) **Дообучение на МАР** – ако МАР е извел съобщение, че има проблеми при генерирането на кода, тогава крайният потребител, бизнес експертът, лингвистът или ИТ експертът анализират причините за това и при необходимост някой от тях дообучава система, като добавя нови знания, внася корекции в модела на предметната област или извършва друга подходяща корекция. **Д09 Проверка на коректността на решението** – Крайният потребител преглежда всички генерирани от системата решения, избира подходящо за работата решение и отстранява от него наличните грешки и неточности, като за целта е възможно да повтори необходимия брой пъти стъпки от Д06 до Д08. **Д10 Генериране на машинно зависим софтуер** – МАР, на базата на избора на крайния потребител генерира машинно зависим код и стартира изпълнението на генерирания софтуер в реално време. Дейностите са подредени в групирани във фази стъпки както следва:

Фаза	Стъпка	Дейнсот по разработването на:
Ф1	С01	Набор системни примитиви за реализацията на УПС
Ф1	С02	Гама интерпретатори за реализацията на различни изчислителни стратегии.
Ф1	С03	Системни примитиви за интерпретация на размити програми
Ф1	С04	Гама интерпретатори за интерпретация на размити програми
Ф1	С05	Набор примитиви за морфологичен анализ
Ф1	С06	Набор примитиви за семантико-синтактичен анализ
Ф1	С07	Набор морфологични филтри
Ф1	С08	Набор Словоредни модели
Ф1	С09	Набор РМП за управление на морфологичен анализ
Ф2	С01	Честотен речник
Ф2	С02	Комбинаторен речник - статии
Ф2	С03	Комбинаторен речник - синтактични функции
Ф2	С04	Комбинаторен речник - синтактични отношения
Ф2	С05	Комбинаторен речник - семантико-синтактични функции
Ф2	С06	Комбинаторен речник - семнатични признаци
Ф2	С07	Комбинаторен речник - словоредни модели
Ф2	С08	Машинен модел на приложната област
Ф2	С09	Набор Матрици на отношенията
Ф2	С10	Набор Семантични филтри
Ф2	С11	Набор параметри за интерпретация на размити програми
Ф2	С12	Набор РМП за управление на семантико-синтактичния анализ
Ф3	С01	Спецификация на решаваната задача
Ф3	С02	Изграждане в диалогов режим на модел на предметната област на решаваната задача
Ф3	С03	Дообучение на Базата знания при необходимост
Ф3	С04	Генериране на програмата
Ф3	С05	Интерпретиране на програмата в диалогов режим
Ф3	С06	Избор на решение

3. Принцип на работа на МАР

Модулът за автоматизирано програмиране представлява инструмент за реализиране на МАПР, който притежава показаните на Фиг.2 елементи. **Актьори**,

работещи с МАР са актьорите на МАПР – ИТ експерт, Лингвист, Бизнес експерт и Краен потребител.



Фиг. 2 Елементи на МАР

На входа на МАР се подава неформална естественоезикова спецификация на програмата, която МАР трябва да генерира. **На изхода на МАР** се получава машинно зависим код, който се инсталира в робота, който програмираме. Този код позволява на робота да изпълни поставените му в текста на входната за МАР неформална спецификация, задачи в реално време. **Компонентите на МАР** са групирани в четири модула както следва: **Потребителски интерфейс**, чиято основна задача, постигната чрез работата на предпроцесора и лингвистичния процесор е да преобразува неформалната естественоезикова входна спецификация във формално дефинирана размита програма. Потребителският интерфейс осигурява още достъпа на експертите до базата знания, включена в компонента **Памет** на МАР. В този компонент е включена и работната памет, необходима за реализацията на МАР чрез парадигмата Управление по данни. Компонента Генератор на решенията, чрез процесора си за обработка на знанията и генератора си на програми осигурява преобразуването на размитата програма в детерминирана, машинно независима програма за управление на робота. **Решаващия блок** на МАР осигурява преобразуването на машиннонезависимата програма в машинно зависима, като освен това осигурява нейното тестване и донастройване, чрез вградения в този компонент симулатор.

Реализирания програмен прототип на МАР показва, че разгледания тук метод може да се приложи в различни предметни области, чрез разширяване най-вече на механизмите за представяне на знанията в този инструмент.

ЛИТЕРАТУРА

[1] Станев И. Метод за автоматизирано програмиране на роботи - изисквания към метода. Научни трудове на Русенски Университет. Том 48, Серия 5.1 2009.

За контакти:

Гл.ас инж. Иван Станев, Катедра “Информатика и информационни технологии”, Русенски университет “Ангел Кънчев”, тел.: 082-888-326, e-mail: IStanev@ami.ru.acad.bg

Докладът е рецензиран.