Декларативен език за програмиране Net Софтуерна архитектура

Иван Станев

Declarative Programming Language Net – Software Architecture: *The paper describes the prerequisites and requirements for the realization of declarative programming language Net, concerning its implementation as specification technique for Method for Automated Programming of Robots. Comparative analysis of imperative (Java), declarative (ML and Prolog), and modeling (UML, BPMN) languages is done. The main characteristics of the language Net are selected based on this analysis. The software architecture of Net is designed, including three components – Net-M Meta-Language, Net-I Interpreter, and Net-V Validator. Two Net variants of usage are described – Metalanguage Net used as superstructure of other languages, and language Net-L used as high level declarative programming language. Two architectural layers of Net are described including Layer 1 Solution Generator, and Layer 2 Real Time Manager.*

Key words: *automated programming, software architecture, SOA, UML, BPMN, declarative languages, imperative languages, CASE, technological framework.*

Езикът Net е разработен като инструмент на Модула за автоматизирано програмиране на роботи (МАР – виж [6]), с чиято помощ в декларативен стил се описват решаваните в МАР задачи, технологията за тяхното решаване, както и знанията за предметната област, в контекста, на която се решават тези задачи.

1. Изисквания към декларативния език за програмиране Net

Като се вземат под внимание избраните в [6] принципи за разработване на МАР и особеностите на разгледаните в [7] изчислителни модели могат да бъдат определени следните предусловия, на които трябва да отговаря езика за програмиране Net: (**1**) да се поражда от изчислителния модел Управляващи мрежи; (**2**) да осигурява подходящи инструменти за реализация на системи за обработка на знания и Архитектури ориентирани към услуги; (**3**) да има вградени инструменти за автоматично решаване на задачата, които могат да бъдат сменяни или пренастройвани според особеностите на разглежданата предметна област; (**4**) да позволява формални графично-текстови спецификации за описание на модела на предметната област; (**5**) да е удобен за работа с неформална естественоезикова спецификация на задачата; (**6**) да реализира или да позволява реализацията на три независими архитектурни слоя; (**7**) да е удобен за интерпретация на размити понятия; (**8**) да дава възможности за евристично, автоматизирано търсене на решения.

2. Сравнителен анализ на езици за програмиране

За да бъдат определени по-важните по отношение на автоматизацията на програмирането характеристики на разработвания език ще бъде извършен сравнителен анализ (Таблица 1) между императивни (Java – [3]), декларативни (ML – [4] и Prolog – [2]) и езици за моделиране (UML – [5], BPMN – [1]). Сравнението ще бъде извършено по характеристики (редовете на Таблица 1), които указват влияние върху процеса на автоматизация. Нека анализираме всеки от включените в таблицата езици:

Java е език за програмиране, който позволява работа с всички типове данни. При необходимост може да поддържа референциално прозрачни променливи. Позволява използването на всички видове основни оператори за програмиране и организира управлението на изчислителния процес, като използва механизма на обектите за предаване на съобщения. Поддържа машина на състоянията като инструмент за управление на данните. Този език е реализиран в два архитектурни слоя и работи с константен интерпретатор на виртуалните Java команди. Предимства на езика по отношение на автоматизация на програмирането са индустриалната му значимост, двуслойната архитектура, работата с обекти,

поддръжката на паралелна работа и вградената машина на състоянията. Недостатъци на езика, свързани с автоматизацията са слабите и неудобно за работа инструменти за спецификация на задачата и решението, липсата на вградени инструменти за автоматизиран извод на решения и нови знания, както и невъзможността стратегията на изчисление да бъде променяна.

Таблица 1 Програмни езици - характеристики

Характеристика	Java	ML	Prolog	UML	BPMN	Net
Типове данни	всички	текст	текст	всички	всички	всички
Тип на променливите	опция за референциална прозрачност	референциално прозрачни	обикновени	опция за референциална прозрачност	опция за референциалн а прозрачност	референциално прозрачни
Типове оператори	всички	функции	неподредени правила	всички	задачи	подредени правила
Инструменти за контрол на управлението	обекти	функции	продукции	диаграми	бизнес процеси	управляващи мрежи
Инструменти за контрол на данните	машина на състоянията	няма	няма	машина на състоянията	няма	управляващи мрежи
Инструменти за поддръжка на паралелна работа	интерпретация на нишки	няма	няма	описание на разклонители и купони	интерпретация на разклонители и купони	опция за интерпретатор на УМ със стртегия за паралелни изчисления
Основен изчислителен механизъм	интерпретатор на Java виртуална машина	интерпретатор на системи уравнения	интерпретатор на продукции	няма	интерпретатор на бизнес процеси	интрепретатор на управляващи мрежи
Стратегия за интерпретация	константна	константна	константна	няма	константна	променлива
Архитектурни слоеве	два	един	един	един	два	два

ML е език за програмиране, който работи единствено с текстови данни. Той използва само референциално прозрачни променливи. При него изчислението е организирано чрез функции. Езикът не поддържа машина на състоянията. Той е реализиран в един архитектурен слой, а интерпретатора на функциите не може да променя стратегията си на изчисление. Предимства на езика по отношение на автоматизация на програмирането са добрия механизъм за специфициране на задачата, без да е необходимо специфициране на решението, наличието на референциално прозрачни променливи, както и вградения механизъм за автоматично търсене на решение. Недостатъци на езика, свързани с автоматизацията са незадоволителната скорост на изчисление при работа в реално време, бедността на спецификационния инструмент, произтичаща от ограничението за работа само с текстови данни и функции и липсата на инструменти за паралелни изчисления.

Prolog е език за програмиране, който работи само с текстови данни, не позволява работа с референциални променливи, организира изчислителния процес като използва продукции, не поддържа машина на състоянията, реализиран е в един архитектурен слой и има интерпретатор за изчисляване на продукциите с константна стратегия на изчисление. Предимства на езика по отношение на автоматизация на програмирането са отличния механизъм за извеждане на нови решения на задачата или нови факти от предметната област, както и възможността да се специфицира само задачата, а решението да се търси автоматично. Недостатъци на езика, свързани с процеса на автоматизация са забавеното изчисление на решението, породено от изпълняваните в случаен ред продукции, бавната работа в реално време поради използването само на текстови променливи и невъзможността да се използва друг освен вградения изчислителен механизъм.

UML е графичен език за моделиране, който работи с всички типове данни, има опция за използване на референциално прозрачни променливи, описва решаваната задача и нейното решение чрез диаграми, поддържа машина на състоянията и позволява описването на паралелни изчислителни конструкции. Основно предимство на езика по отношение на автоматизация на програмирането е, че разполага с отлична спецификационна техника, която позволява формално пълно и точно да се опише с един и същ инструмент както решаваната задача, така и нейното решение от всички възможни гледни точки. Тази техника, освен че е много удобна за проектиране, осигурява отлични възможности за автоматичен формален и логически контрол на коректността на разработваните спецификации. Сериозен недостатък на езика е, че липсват интерпретатори на създадените по време на проектирането модели. Съществуващите техники за генериране от тези модели на кух код, който в последствие може да се доработи на някой от императивните езици за програмиране, не са особено удобни и нямат практическа значимост. Освен това езикът не е пригоден за автоматично генериране на решения и нови знания за предметната област.

BPMN е графичен език за моделиране, който работи с всички типове данни, има опция за използване на референциално прозрачни променливи, описва решаваната задача и нейното решение чрез бизнес процеси и позволява описването на паралелни изчислителни конструкции. С двете си архитектурни нива езикът представлява завършен инструмент за автоматизация на програмирането. Задачата и нейното решение се описват чрез бизнес процеси на горното машинно независимо архитектурно ниво. Всеки възел на бизнес процеса представлява услуга (обикновено машинно зависима и написана на императивен език) от долното архитектурно ниво. Интерпретаторът на езика управлява изчислителния процес, следвайки инструкциите на бизнес процесите, предава управлението на текущо извиканата услуга и приема обратно управлението на процеса след края на нейното изпълнение. При добро проектиране голяма част от услугите могат да бъдат разработени за многократна употреба и съхранени в библиотека на услугите за използване в следващи решавани задачи. Недостатъци на този език са не особено добрата ефективност при работа в реално време, необходимостта да се описват както решаваната задача, така и нейното решение и липсата на инструменти за автоматично генериране на решение и нови знания в предметната област.

В резултат от анализа на тук разгледаните езици можем да заключим, че: (**1**) Няма нито един език, който да осигурява сменяем или пренастройващ се в зависимост от особеностите на предметната област инструмент за автоматизирано решаване на задачата. (**2**) Няма нито един език, който позволява неформална спецификация на решаваната задача и на начините за нейното решение. (**3**) За да бъдат изпълнени предусловията за реализацията на MAP е необходимо да се разработи нов език, който притежава комбинация от характеристики на различните тук разгледани езици.

На базата на тези заключения можем да определим изискванията към разработвания език Net както следва:

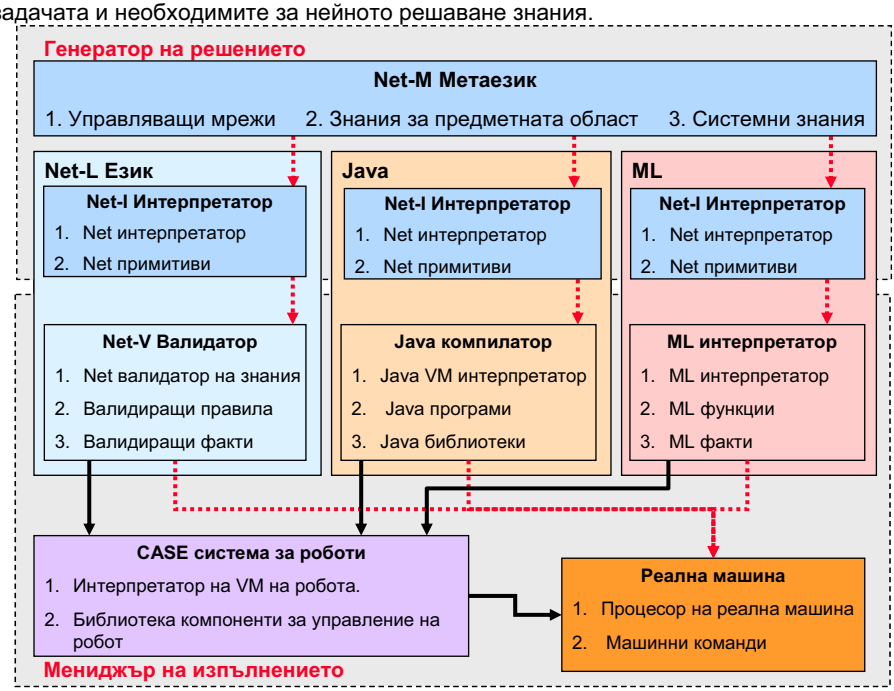
Net трябва да бъде език за моделиране и програмиране от декларативен тип. Той трябва да позволява работа с всички типове данни и да ползва (аналогично на ML) референциално прозрачни променливи. Аналогично на Prolog езикът трябва да работи с правила, но за разлика от Prolog, правилата трябва да се изпълняват в подредба, зависеща от контекста на решаваната задача. Необходимо е правилата да са подредени в управляващи мрежи (аналогично на бизнес процесите на BPMN) и тези правила да са изградени от написани на императивен език услуги (аналогично на BPMN). Net трябва да работи с гама интерпретатори, чиято промяна да води до смяна на стратегията на изчисление. Между тези интерпретатори трябва да има такива, които позволяват: използване на паралелна стратегия на изчисление

(аналогично на Java, UML и BPMN); симулация чрез управляващи мрежи на машина на състоянията; както и такива, които реализират евристични стратегии за генериране на решението. Езикът трябва да се реализира в два архитектурни слоя и горния от тези слоеве да позволява генериране на разработвания продукт или търсеното решение, а долният слой да осигурява управление на изпълнението на генерирания продукт в реално време.

3. Компоненти в езика Net

При дефинирането на езика Net са спазени определените в предходната точка изисквания, а разработката следва контекста и терминологията за разработка на програмни езици, въведени в Aho.78.

Езикът Net е изграден от следните три компоненти: (1) **Net-M** (**компонент метаезик**), който е предназначен да описва в термините на езика Net решаваната задача, знанията от предметната и ИТ областта, необходими за решаване на задачата, както и да управлява процеса по автоматизираното решаване на тази задача или извличането на нови знания за разглежданата предметна област. (2) **Net-I** (**компонент интерпретатор**), който включва програмния код на интерпретатора и на използваните от управляващите мрежи примитиви. Тези инструменти се използват от надстроявания език за интерпретиране на описаните в термините на език Net действия и знания. (3) **Net-V** (**компонент валидатор**), който включва програмния код на валидатора, валидиращите правила и факти, проверяващи функционалната пълнота и логическата непротиворечивост на спецификацията на задачата и необходимите за нейното решаване знания.



Фиг. 1 Архитектурни нива и компоненти на език Net

Езикът Net е разработен в два варианта както следва: (1) **Метаезик** (**Net**), когато с компонентите Net-M и Net-I се надстроява друг език от високо ниво (на Фиг. 1 примерите с надстрояване на Java и ML). В този случай Net.M се разработва еднократно за дадена предметна област, а интерпретаторът и примитивите от Net.I

се разработват като програмен код за всеки език, който ще използва Net.M за конкретната предметна област. (2) **Декларативен език от високо ниво** (**Net.L**), когато се използват и трите компонента Net-M, Net-I и Net.V за самостоятелно програмиране в декларативен стил чрез управляващи мрежи (на Фиг. 1 примера Net.L). В този случай отново Net.M се разработва еднократно и зависи само от предметната област. Примитивите и интерпретатора на Net.I се пишат директно на машинно зависим език от ниско ниво. В този вариант се добавя и компонент Net.V, който поема функциите на семантичен процесор, изпълнявани в предходния вариант от надстрояваните езици.

И двата варианта са реализирани като са използвани два **архитектурни слоя**: (**1**) **Генератор на решение** (Архитектурен слой 1), който има задачата да управлява (червените пунктирани стрелки от Фиг. 1) работата по автоматизирано генериране на нов софтуер и нови знания от предметната област. (**2**) **Мениджър на изпълнение** (Архитектурен слой 2), който има задачата да преобразува (червените пунктирани стрелки от Фиг. 1) решението на задачата, интерпретатора и знанията за предметната област, от надстроявания език от високо ниво в езика на реалната машина, както и в последствие да управлява процеса на изпълнение в реално време на генерираното решение.

В случаите на програмиране на робот е възможно във втория архитектурен слой да има един допълнителен компонент - CASE система за програмиране на робот. В този случай програмите не се преобразуват до машинен език, а до езика на използваната CASE система (черните непрекъснати стрелки от Фиг. 1). След това преобразуване CASE системата поема грижата за бъдещото доработване и изпълнение в реално време на разработените на език Net приложения. Този механизъм осигурява възможността езика Net да се използва като модул за автоматизирано програмиране на роботи (МАР) във вече съществуващи CASE системи за програмиране на роботи.

ЛИТЕРАТУРА

[1] BPMN. Business process model and notation v1.2. OMG Document Number: formal/2009-01-03 Standard document URL: http://www.omg.org/spec/BPMN/1.2. OMG. 2009. Pp. 294.

[2] Clocksin W.F., Mellish, C.S. Programming in Prolog: Using the ISO Standard. Springer. 5th ed.. 2003. Pp. 307

[3] Gosling J., et. al. The Java Language Specification. Addison Wesley.2005. Pp. 684

[4] Milner R. et al., The Definition of Standard ML. The MIT Press. 1990

[5] OMG Unified Modeling Language v2.3. OMG Document Number: formal/2010-05-03. Standard Specification. URL http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/. OMG. 2010. Pp. 226.

[6] Станев И. Метод за автоматизирано програмиране на роботи - Изисквания към метода. Научни трудове Том 48, Серия 5.1. Математика, Информатика, Физика. Русенски Университет “А. Кънчев”. Русе. 2009 г. Стр. 81-85.

[7] Станев И., К. Крачанов, В. Вълева. Теория на алгоритмите. Русенски Университет. 1997. Стр. 210.

За контакти:

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

Докладът е рецензиран.