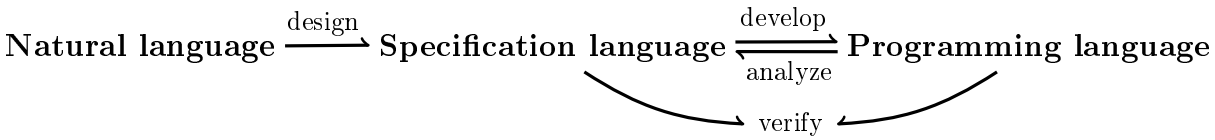
Автоматизации при разработка на софтуер

Разработката на комплексни софтуерни приложения може да се раздели на няколко основни под-процеса:

- **дизайн** (**design**): този процес има за цел да събере изискванията към бъдещата софтуерна система, да формулира нейното задание по тези изисквания и да определи основните терминологични поянтия и задачи от областта, която ще бъде обслужвана от системата; заданието и терминологичния анализ спомагат за формирането на план за реализация на софтуерното решение на поставените задачи;
- **разработка** (**develop**): това е процеса по създаване на софтуерното приложение, което изпълнява решенията на формулираните задачи; разработката следва заданието и начертания план от процеса по дизайн;
- **анализ** (**analyze**): анализ на вече разработен софтуер е определянето, дали софтуерът притежава определени свойва; този процес може да се наложи за да се провери, дали вече разработен софтуер може да изпълни специфични изисквания от заданието (и по този начин да се спести раработката на ново приложение); също така анализа позволява да се открият свойства на софтуера, които не са били включени в първоначалното задание; тази информация спомага за последващи оптимизации, разширения и/или интеграция в по-големи системи;
- **верификация** (**verify**): този процес установява дали вече разработения софтуер реализира точно и коректно поставените му задачи; той може да взаимства информация от процеса по анализ, която да помогне за определяне на свойствата и качествата на софтуера, или пък да използва независими методи, които да проверяват коректността на софтуера спрямо началното му задание.

Тези под-процеси на софтуерната разработка използват огромни количества от данни на всеки един етап от тяхната работа. Обработката на тези данни изисква най-ралични формулировки и нотации на запис, което на свой ред въвлича използването на множество от различни типове езици, които се използват за запис и трансфер на данните.



- **Естествен език** (**Natural language**): Това е неформален или полуформален (структуриран) език, като български език, или английски език, или др. Той се използва за комуникация между различните групи от специалисти и неспециалисти, които договарят и описват изискванията за софтуерната система.
- **Език за спецификация** (**Specification language**): Това е формален (често математически) език, на който се записва изготвеното по време на дезайн задание. Този език може да е език за моделиране (като UML), или пък специализиран език за запис на спецификации (например, CASL, VDM, или Z), или подходяща математическа система за формуилране на свойства на софтуера (като LTL, PDL или Hoare logic).

- **Програмен език** (**Programming language**): Програмния език е избраната софтуерна технология, чрез която реализираме новия програмен продукт. Този език може да е някой от популярните езици за програмиране (като C#, Java, C++, JavaScript, Python, и др.), език за заявки към бази от данни (SQL), или пък структурен език за описание и съхранение на данни (като HTML или XML).

Горната схема описва най-основния случай на взаимоотношения между различните типове езици (различните взаимоотношения се обслужват от различни основни под-процеси на софтуерната разработка). В някои конкретни разработки може и да не участват всички видове езици. Например, при по-прости системи (или при по-ограничени средва) може да не се използват специализирани езици за спецификации, ами заданието се записва директно на (често структуриран или ограничен по друг начин) естествен език. Но с покачване на сложността на поставените задачи (и като следсвтие и сложността на системата) основната схема въвлича все повече етапи на работа и различни езици. Например, началните етапи на разработка могат да използват няколко естествени езика за комуникация между групи от различни страни (български език и английски език, например) или между групи с различна техническа специализация (разговорен език и терминологичен език). Също така при участието на много на брой отдели, експертни групи и/или консултански фирми, може да се използват повече от един език за спецификации. А при многослойните софтуерни системи, всеки слой от системата ще наложи използванто на специализиран програмен език за реализация на задачите от този слой (например, SQL за слоя за бази от данни, C++ за сървърната част от системата и JavaScript за клиентските приложения).

Така, разработката на една комплексна софтуерна система използва (потенциално) много на брой езици и по този начин основните под-процеси могат да се разглеждат като задачи за трансформация/превод между различните езици или пък за лингвистичен анализ на формулираните условия. Т.к. на всяка стъпка се обработват големи обеми от данни, то възможността за компютърна автоматизация на тези задачи е от голямо значение за софтуерната индустрия и може да доведе до значително намаляване на себестойността на софтуерната разработка.

Последващите описания представят шест отделни теми (групи от задачи), в които съвременни технологии от копютърните науки могат да се прилагат за постигане на пълна или частична автоматизация на под-процесите на разработка.

Автоматизации при дизайн и разработка I

Трансформация и превод между естествени езици и езици за спефицикации може да се постигне чрез прилагане на следните технологии от компюртърната лингвистика и/или обработката на естествени езици: *машинен превод* (*статистически* и/или *граматически*), *анализ на части от речта* (чрез *Станфордския* анализатор, например), *семантичен превод* на формални дефиниции, *извличане на понятия*, *анализ на многозначия*, *формални граматики* и др.

$$\textbf{Natural language} \underset{\text{semantical translation}}{\overset{\text{machine translation, disambiguation, entity extraction}}{\rightleftharpoons}} \textbf{Specification language}$$

Приложението на тези технологии може да постигне следните подобрения и оптимизации в под-процесите на софтуерна разработка:

- автоматизация на съставянето и запис на софтуерни задания и спецификации;
- съставяне на софтуерни задания и спецификации от неспециалисти;
- читаемост на софтуерни задания и спецификации от неспециалисти;
- автоматична верификация на коректността на формални задания и спецификации спрямо техни описания на естествен език.

Автоматизации при дизайн и разработка II

В случаите, когато при дизайна и разработката на софтуерната система не участват езици за спецификации, задачите от предната тема могат да се адаптират към директен превод към/от програмните езици. Т.к. отново става въпрос за трансформация/превод между неформален и формален език, голяма част от технологиите, приложими в предишната тема, са използваеми и тук - *машинен превод*, *лексикален анализ*, *семантичен превод* (чрез денотационна и операционна семантика на езиците, формални интерпретации, типови системи), *извличане на понятия*, *анализ на многозначия*, и др. По този начин могат да се получат методологии на разработка, подобни на English Query или Full Text Search от Microsoft SQL Server.

$$\textbf{Natural language} \underset{\text{semantical translation}}{\overset{\text{machine translation, disambiguation, entity extraction}}{\rightleftharpoons}} \textbf{Programming language}$$

Потенциалните ползи от използването на тези технологии за конкретната задача са:

- автоматизация на съставянето на софтуерни програми;
- съставяне на програми от неспециалисти;
- читаемост на програми от неспециалисти;
- автоматична верификация на коректността на софтуерен код спрямо описания на естествен език.

Автоматизации при дизайн и разработка III

Както и в предните две теми, можем да имаме аналогична трансформация/превод и между езиците за спецификации и програмиране. Към тази задача може да се подходи със следните технологии: *машинен превод*, *семантичен превод* между формални езици и дефиниции, *синтактични преобразуватели* ("parse"-ери и "port"-ери), *компилиране* и *транслиране* между спецификации и програмни езици.

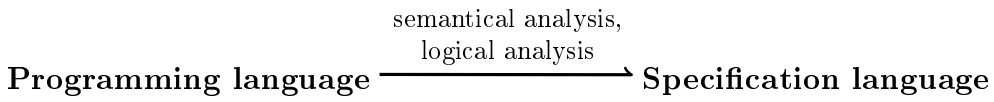
$$\textbf{Specification language} \underset{\text{machine translation, semantical translation}}{\overset{\text{machine translation, semantical translation}}{\rightleftharpoons}} \textbf{Programming language}$$

Възможните подобрения на софтуерната разработка чрез решения на тази задача включват:

- автоматично генериране на софтуерни програми по спецификации;
- читаемост на програми от аналитични специалисти, които не познават съответния език на програмиране;
- автоматична верификация на коректността на софтуерен код спрямо неговата спецификация.

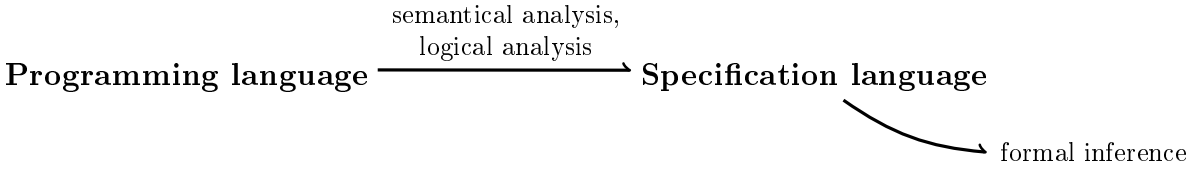
Автоматизации при анализ на софтуер

Автоматизациите при анализа на софтуерен код могат значително да помогнат за поддържката на частична коректност на кода, както и за неговата оптимизация. Анализа, обикновено, се състои от определяне на формални свойства на програмите (т.е. един вид трансформаация/превод от програмен език към език за спецификации или друг формален и/или математически език) и след това прилагане на алгоритъм за атвоматичен анализ на извлечените свойства. Приложими са технологии и методологии от компюртърната лингвистика, математическата логика и/или статистиката като: *машинен превод*, *семантичен превод* между формални езици и дефиниции, *формален анализ*, *логически анализ*, *Symbolic execution*, *Control flow analysis*, *Data-flow analysis*, *симулации* (динамичен и статичен анализ), *статистическо класифициране* и *статистическа подредба*.



Автоматизации при верификаия на софтуер I

Някои от предните задачи за трансформация/превод от програмни езици към описания и спефицикации на естествени и/или формални езици могат много успешно да служат като инструменти за автоматична верификация на програмния код. При тези задачи успешния превод на програмата до оригиналното ѝ описание и/или спецификация потвърждава нейната коректност. Често, обаче превода на програмата до друг формален език не дава директно спецификацията, ами дава нейна еквивалентна формулировка или пък по-общо твърдение, на което спецификацията е само частен случай или следствие. Така процесът по верификация на програмата може да се разшири до комбинация от превод/анализ на кода до спецификации/свойства заедно с похвати за анализ от математическата логика, изкуствения интелект и/или статистиката като: *формален извод*, *анализ за изпълнимост*, *анализ за непротиворечивост*, *семантични мрежи*, *невронни мрежи*, *Марковски модели*, *приемане/отхвърляне на хипотези*, *статистически извод*.



Автоматизации при верификаия на софтуер II

Друг похват за автоматична верификация на софтуер е не да се извличат свойствата му и по тях да се потвърждава неговата коректност, а вместо това написаната спефицикация да се използва като своеобразен шаблон, който да се приложи върху програмния код и така да се извърши проверката. Този метод е познат като *проверка на модела* (*model checking*). При този подход програмата се представя като формална структура, а спецификацията/свойството като формула, и се проверява за изпълнимост на тази формула в съответната структура. Тази дейност може допълнително да се подпомогне от действия като превод до съждителни формули или control flow graph и прилагане на SAT разрешители. Популярни приложения за проверка на модели са JavaPathFinder, Cmc, MaceMC, F-Soft, Saturn, Calysto, SPIN, CHESS.