Поновна употребљивост софтвера

“Before software can be reusable it first has to be usable.”
Ralph Johnson

Поновна употребљивост софтвера

- У већини области које се баве неком врстом пројектовања, системи се дизајнирају састављањем постојећих компоненти које се користе у другим системима
- Софтверско инжињерство је више било фокусирано на развој од почетка до краја, али данас је препознато да уколико тежимо бољем софтверу, брже развијеном и по нижој цени, морамо усвојити процес развоја који се заснива на систематском поновном коришћењу софтвера

Развој заснован на поновној употребљивости

- Поновна употреба апликационог система
- Поновна употреба компоненти
- Поновна употреба објеката и функција

Предности

- Поузданост
- Олакшано управљање ризицима
- Коришћење знања експерата из дате области
- Усаглашеност са стандардима
- Убрзан процес развоја

Предности

- Поузданост
  - софтвер који користимо је већ испробан и тестиран у системима који раде
  - претходне употребе су показале недостатке, који су временом поправљани
- Олакшано управљање ризицима
- Коришћење знања експерата из дате области
- Усаглашеност са стандардима
- Убрзан процес развоја

Предности

- Поузданост
- Олакшано управљање ризицима
  - знамо тачно колико морамо да платимо за готов софтвер, док никада не можемо са прецизношћу да израчунамо колико би коштао његов развој
  - посебно долази до изражаја код већих пројеката
- Коришћење знања експерата из дате области
- Усаглашеност са стандардима
- Убрзан процес развоја

Предности

- Поузданост
- Олакшано управљање ризицима
- Коришћење знања експерата из дате области
  - уместо да експерти раде исти посао на различитим пројектима могу удружити своја знања и развијати поново употребљив софтвер
- Усаглашеност са стандардима
- Убрзан процес развоја

Предности

- Поузданост
- Олакшано управљање ризицима
- Коришћење знања експерата из дате области
- Усаглашеност са стандардима
  - неки стандарди попут стандарда корисничког интерфејса могу бити имплементирани као скуп стандардних поново употребљивих компоненти
  - стандардизован кориснички интерфејс повећава поузданост, јер корисници ређе греше када користе познате интерфејсе
- Убрзан процес развоја

Предности

- Поузданост
- Олакшано управљање ризицима
- Коришћење знања експерата из дате области
- Усаглашеност са стандардима
- Убрзан процес развоја
  - некада је брзина пласирања производа на тржиште далеко битнија од укупне цене развоја
  - коришћење поновно употребљивих компоненти убрзава фазе развоја и тестирања што битно скраћује укупно време развоја

Проблеми

- Повећана цена одржавања
- Недостатак алата
- “Није направљено овде” синдром
- Прављење и одржавање библиотеке
- Проналажење, разумевање и прилагођавање поново употребљивих компоненти

Проблеми

- Повећана цена одржавања
  - ако компонента коју користимо није отвореног кода онда се трошкови одржавања увећавају како компонента постаје неусаглашена са променама система
- Недостатак алата
- “Није направљено овде” синдром
- Прављење и одржавање библиотеке
- Проналажење, разумевање и прилагођавање поново употребљивих компоненти

Проблеми

- Повећана цена одржавања
- Недостатак алата
  - већина CASE (Computer-aided software engineering) алата не подржава развој са коришћењем поново употребљивих компоненти
- “Није направљено овде” синдром
- Прављење и одржавање библиотеке
- Проналажење, разумевање и прилагођавање поново употребљивих компоненти

Проблеми

- Повећана цена одржавања
- Недостатак алата
- “Није направљено овде” синдром
  - неки софтверски инжењери радије бирају да испочетка развију компоненту, јер верују да могу да направе бољу
  - ово се дешава делимично због неповерења, а делимично због тога што делује изазовније написати оригиналан софтвер него користити туђи
- Прављење и одржавање библиотеке
- Проналажење, разумевање и прилагођавање поново употребљивих компоненти

Проблеми

- Повећана цена одржавања
- Недостатак алата
- “Није направљено овде” синдром
- Прављење и одржавање библиотеке
  - попуњавање библиотеке поновно употребљиве компоненте и обезбеђивање да софтверски инжењери могу да је користе може бити скупо
  - технике за класификацију, каталогизацију и преузимање компоненти софтвера које нам стоје на располагању су прилично ‘незреле’
- Проналажење, разумевање и прилагођавање поново употребљивих компоненти

Проблеми

- Повећана цена одржавања
- Недостатак алата
- “Није направљено овде” синдром
- Прављење и одржавање библиотеке
- Проналажење, разумевање и прилагођавање поновно употребљивих компоненти
  - софтверске компоненте морају прво бити пронађене у библиотеци, схваћене, а неретко и прилагођене специфичним потребама
  - инжењери морају бити довољно сигурни да ће у библиотеци пронаћи компоненту која им треба пре него укључе тражење у свој регуларни процес развоја

Проблеми

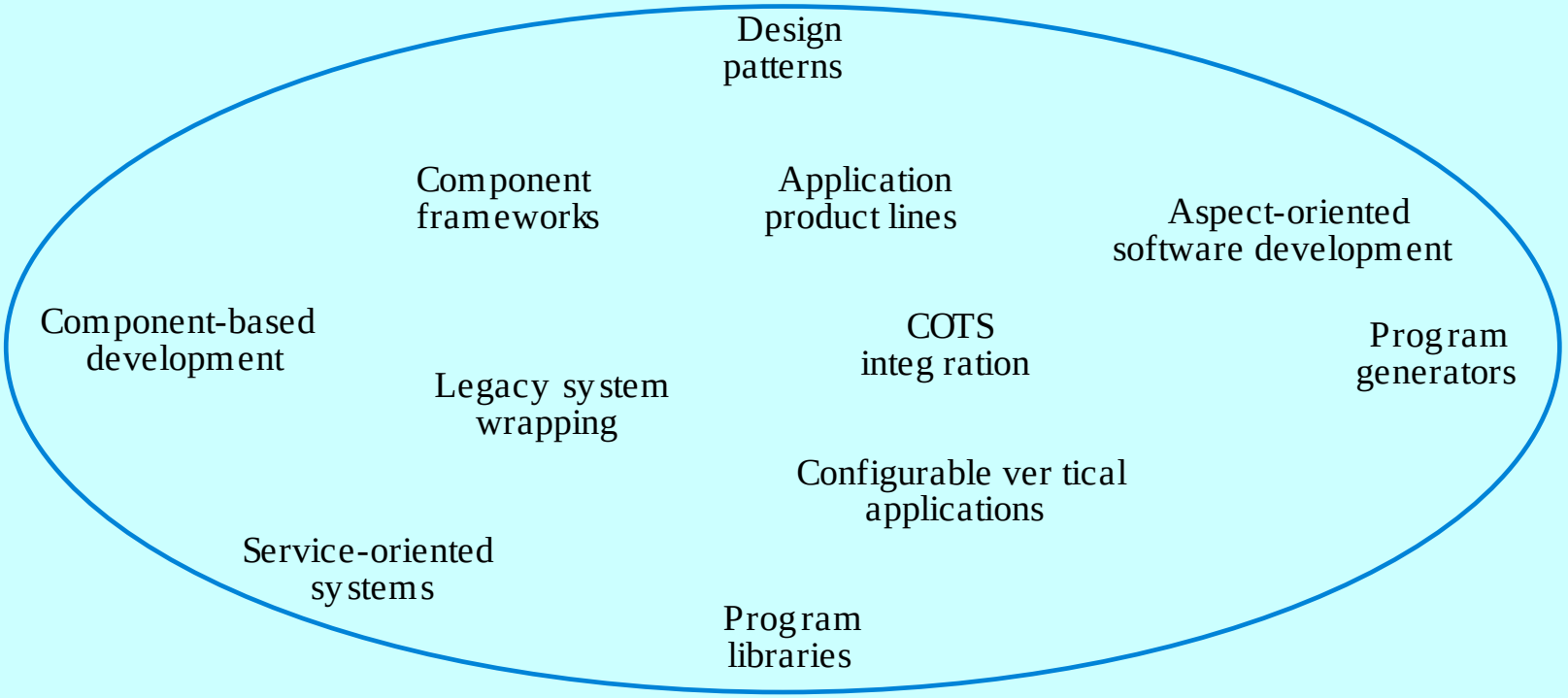

WE HAVE DECIDED TO REUSE THE CODE OF OUR PAYROLL SOFTWARE ALSO IN DEVELOPING OUR POINT OF SALE SYSTEMS!


AT A SUPERMARKET...
THERE IS SOMETHING WRONG WITH THIS SYSTEM. DURING WEEKENDS, AND HOLIDAYS, IT AUTOMATICALLY CHARGES A PREMIUM TO THE ITEM PRICE!

Технике коришћења поновно употребљивих компоненти

- Иако се поновна употребљивост обично посматра само као поновна употреба компоненти неког система, постоји велики број различитих начина на који се она може извршити
- Поновна употребљивост је могућа на различитим нивоима од једноставних функција до целовитих апликација

Технике коришћења поновно употребљивих компоненти



Технике (1)

- Узорци за пројектовање
  - генеричке апстракције које се јављају у различитим апликацијама се представљају у виду узорака за пројектовање
- Развој заснован на компонентама
  - системи се развијају путем интегрисања компоненти (колекција објеката) које одговарају стандардима модела компоненти
- Апликациони оквири
  - колекције апстрактних и конкретних класа које могу бити прилагођене и проширене при прављењу апликационог система

Технике (2)

- Обмотавање застарелих система
  - застарели системи се могу ‘обмотати’ скупом интерфејса преко којих се омогућава приступ овим системима
- Системи оријентисани ка сервисима
  - системи се развијају повезивањем дељених сервиса који нам неко нуди са стране
- Апликационе линије производа
  - један тип апликације је генерализован на основу често коришћене архитектуре тако да може бити прилагођаван на различите начине за потребе различитих корисника

Технике (3)

- COTS интеграција
  - системи се развијају уградњом постојећих апликационих система
- Конфигурабилне вертикалне апликације
  - генерички систем се пројектује тако да може бити прилагођен потребама појединачних корисника
- Програмске библиотеке
  - Библиотеке класа и функција које имплементирају често употребљаване апстракције

Технике (4)

- Генератори програма
  - систем генератор има уграђено знање о специфичним типовима апликација и може да генерише цео систем или неке његове делове у том домену
- Развој оријентисан према аспектима
  - дељене компоненте су ‘уткане’ на различитим местима у апликацији при компилацији програма

Фактори при планирању поновне употребе

- Планирани распоред развојаж
- Очекивани животни век софтвера
- Искуство и вештине чланова развојног тима
- Критичност функције коју софтвер обавља и нефункционални захтеви
- Домен апликације
- Платформа на којој ће се софтвер користити

Поновна употреба концепта

- Када поново користимо програм или неку компоненту, морамо се држати одлука везаних за дизајн које су донели првобитни развијаоци
- Ово може бити ограничавајући фактор
- Апстрактнија форма поновне употребе је поновна употреба концепта где је одређени приступ објашњен тако да не зависи од конкретне имплементације и на основу њега се врши имплементација у складу са потребама

Поновна употреба концепта

- Два главна приступа су:
  - Узорци за пројектовање
  - Генеративно програмирање

Узорци за пројектовање

- Узорци за пројектовање су вид поновне употребе апстракције знања о проблему и његовом решењу
- Узорак представљаопис проблема и суштину његовог решења
- Мора бити довољно апстрактан да може бити поново примењен у различитим окружењима и околностима
- Узорци се често ослањају на неке карактеристике објеката као што су наслеђивање и полиморфизам

Елементи узорка за пројектовање

- Име
  - у складу са наменом
  - на основу имена узорак се јединствено идентификује
- Опис проблема
- Опис решења
  - не конкретан дизајн већ шаблон на основу кога се могу правити решења у различитим околностима
- Последице
  - обухватају резултате и оцене примене узорака

Генеративно програмирање

- Укључују поновну употребу стандардних решења и алгоритама
- Они су уграђени у генератор и параметризовани од стране корисника, након чега се програм аутоматски генерише
- Могуће је када апстракција домена и њихово мапирање у извршни код може бити идетификовано
- Језик специфичан за домен се користи за компоновање и контролу ових апстракција

Врсте генератора програма

- Врсте генератора програма
  - генератори апликација за обраду пословних података
  - парсери и лексички анализатори за обраду језика
  - генератори кода за CASE алате
- Веома су погодни са аспекта цене, али је њихова примена ограничена на релативно мали број апликационих домена
- За крајње кориснике је лакше да развијају програм користећи генераторе него остале видове поновне употербе заснованена компонентама

Поновна употребљивост кроз генерисање кода