20. Проектиране и интегриране на софтуерни системи. (ПИСС)

Анотация: Изложението по въпроса трябва да включва следните по-съществени елементи:
1. Характеристики на разпределените софтуерни системи – дефиниции, видове системи и техните тенденции.
2. Междупроцесна комуникация – отдалечено извикване, мултикаст.
3. Разпределени обекти и компоненти.
4. Уеб услуги – дефиниции, шаблони за комуникация. Стандарти за уеб услуги – SOAP, UDDI, WSDL.

1. Характеристики на разпределените софтуерни системи

1.1. Дефиниции
Налични са следните дефиниции:
- Разпределената софтуерна система е като група от независими компютри, които за потребителя изглеждат като една цялостна система.
- Система, в която хардуерни или софтуерни компоненти, принадлежащи на компютри (възли), свързани в мрежа, комуникират един с друг и координират своите действия само чрез изпращане на съобщения.

1.2. Видове системи
Има следните видове:
- разпределени изчислителни системи;
- разпределени информационни системи;
- широкоразпространени разпределени системи.

Разпределените изчислителни системи се използват за изчислителни задачи изискващи висока производителност. Има следните видове:
- **Клъстер** – една програма работи паралелно върху много възли. Тези възли са идентични (откъм хардуер), с еднаква ОС, близко са разположени физически и са свързани в една мрежа.
- **Грид** – хетерогенни системи, където може ОС, хардуера или мрежата на възлите да се различават. Често възлите са отдалечени физически.

Разпределените информационни системи се използват за свързване с множество приложения, които си комуникират по мрежата и може да са оперативно несъвместими. Интеграцията на такива приложения е на няколко нива, като основните са:
- **Client-Server** – на ниско ниво се позволява клиентите да групират заявки към сървър, които се изпълняват атомарно (в цялост).
- **Peer-to-Peer** – на по-високо ниво се позволява приложенията да комуникират едно с друго.

Широкоразпространените разпределени системи (Pervasive distributed systems) за разлика от другите два вида, които са стабилни, са нестабилни. Това се налага поради появата на мобилни и вградени изчислителни системи, където нестабилността е неимоверна. Възлите в такива мрежи са мобилни, малки, черпещи енергия от батерии и обичайно използват безжична връзка. Характеризират се с липса на нужда от непрекъснат човешки надзор.

1.3. Тенденции
- Използването на широкоразпространени мрежови технологии.
- Подобрява се интеграцията на малки и мобилни изчислителни устройства в разпределените системи. Такива са телефони, умни часовници, навигационни устройства, табла на коли и други.
- Интернет е една голяма разпределена система, която постоянно се разширява.
- Увеличава се търсенето на мултимедийни услуги.
- Разпространение на on-demand разпределени системи.

2. Междупроцесорна комуникация

Комуникацията между процеси се извършва чрез обмен на съобщения (Inter Process Communication (IPC)). ОС предоставя възможност за имплементация на IPC чрез механизми като:

- message queue;
- семафори;
- shared memory.

IPC механизмите на ОС обичайно не се използват директно, а посредством middleware програми, които служат като обвивка около тях, предоставящи абстракция чрез интерфейс. Има два вида комуникация между процеси:

- **connected-oriented** – чрез изграждане на канал специално за комуникацията;
- **message-oriented** – чрез преизползване на вече съществуващ канал за комуникация.

Важна част от middleware представлява message-passing, който е механизъм за комуникация без използване на споделена памет.

2.1. Отдалечено извикване

Отдалечените извиквания представляват начин за използване на функционалност на процес A от порцес B, като B извиква функционалност от A по прозрачен начин за себе си, т.е. замаскирвайки, че A е друг процес. Може да се имплементират чре RPC и RMI.

Remote Procedure Call (RPC) се базира на предоставянето на интерфейси, специфициращи процедури предлагани от друг процес. По този начин се скрива реализацията на другия модул. Интерфейсите се дефинират посредством Interface Definition Language (IDL), който е проектиран с цел да позволи извикването на процедура от един език в друг език.

Семантиките на RPC са:

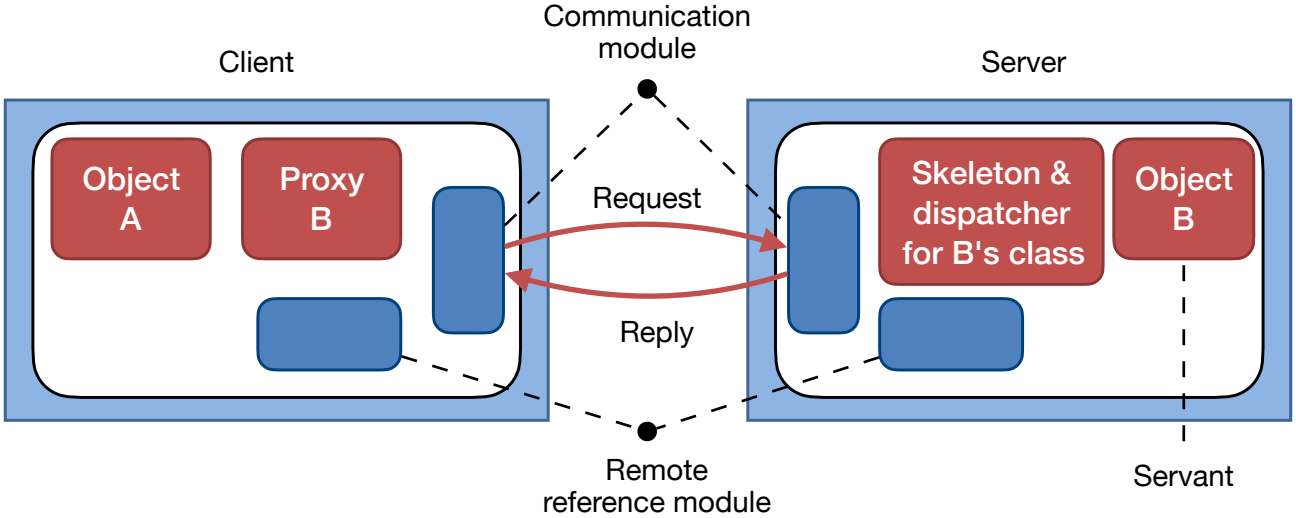
- maybe – отдалечената процедура се изпълнява веднъж или въобще не се изпълнява.
- at-least-once – клиентът получава резултат и знае, че процедурата се изпълняввa поне веднъж или получава информация за липса на резултат.
- at-most-once – клиентът получава резултат и знае, че процедурата се е изпълнила най-много веднъж, или получава информация за липса на резултат.

Remote Method Invocation (RMI) е обектно-ориентиран метод за отдалечено извикване на методи.
RMI прилича на RPC по това, че:

1. Поддържа програмиране с интерфейси;
2. Базира се на протокол заявка-отговор;
3. Поддържа подобно ниво на прозрачност.

RMI се различава от RPC по това, че:

1. Позволява работа с обектно-ориентираната парадигма;
2. Предоставя възможност за предаване на референции към обекти като параметри.



Реализацията на RMI включва:

- **Комуникационен модул**, който препраща заявките и отговорите между клиента и сървъра и осигурява семантиката на съобщението. От страна на сървъра комуникира с dispatcher-а на класа за извикания обект.
- **Remote reference module**, който е отговорен за съпоставката между локалните и отдалечените референции на обекти, създаването на отдалечени референции, както и тяхната маршализация/демаршализация.
- **Servant** – инстанция на отдалечен обект.
- **Proxy/Stub** – имплементира отдалечения интерфейс от страна на клиента, като служи за препращане на заявките за извикване на методи към съответния отдалечен обект.
- **Skeleton** – имплементира отдалечения интерфейс от страна на сървъра, като делегира работата на Servant.
- **Dispatcher** – получава заявките от комуникационния модул и използва идентификатора на заявената операция за да извика съответния метод на Skeleton обекта.

2.2. Мултикаст

Мултикаст представлява начин за изпращане на информация от един до много възли в една мрежа. Топологията на тази мрежа може да бъде образувана като възлите оформят:

- **дърво** и има директен достъп между всеки два възела;
- **mesh** и може да има няколко пътя от един връх до друг.

При мултикаст, рутерите не се борят за възли, следователно може рутирането да не е толкова оптимално както на мрежовото ниво.

Разпространението на информацията се оптимизира посредством алчния алгоритъм/техника gossip (Gossip-Based Data Dissemination), където аналогично на разпространението на човешки вирус, всеки възел предава информацията на съседите си. Възел, който държи информация се нарича заразен, като той може да я предаде на своите съседи. При епидемологичните алгоритми предаването на информация за това кога да се изтрие информация е трудно. Поради тази причина се симулира изтриването на информация чрез нейното модифициране.

3. Разпределени обекти и компоненти

3.1. Разпределени обекти

Този подход обуславя използването на обектно-ориентирания модел при разработването на разпределени системи. При него комуникиращите единици са обекти, които комуникират чрез RMI или разпределени събития.

Разпределените обекти предоставят добри абстракции и могат да се възползват от обектно-ориентираните принципи за дизайн, средства и техники за раработка на разпределени системи.

Пример за разпределен обектно-ориентиран middleware е RMI.

При този подход обаче могат да настъпят редица проблеми:

- интерфейсите на обектите не описват от какво зависи имплементацията им, което ги прави трудни за разработка и поддържане;
- програмистите трябва да са наясно с детайли от по-ниско ниво като например имоплементацията на middleware;
- програмистите са длъжни да мислят за сигурността, обработката на грешки, конкурентността и други подобни детайли;
- няма out-of-the-box поддържа за сложни конфигурации от обекти.

3.2. Разпределени компоненти

Проблемите на разпределените обекти обуславят нуждата от друга парадигма, наречена разпределени компоненти. Софтуерен компонент е градивна единица, за която е описано какви интерфейси трябва да предлага и какви са нейните контекстни зависимости.

Примери за разпределени компоненти са Enterprise Java Beans и CORBA.

4. Уеб услуги

4.1. Дефиниции

Интерфейс на уеб услуга наричаме колекция от операции, които могат да бъдат достъпвани през интернет. Операциите може да се имплементират от различни ресурси (програми, бази от данни и други). Уеб услугите се достъпват посредством HTTP протокола. Стандартни начини за имплементация са чрез обработване на XML SOAP съобщения или използвайки REST архитектура. Целта на уеб услугите е да абстрахира изпълнението на разпределени изчисления от начина, по който те са имплементирани (програмни езици и прочие).

4.2. Шаблони за комуникация

Има следните видове комуникация:

- **Синхронна комуникация** – обмен на съобщения, при който клиента блокира до изчакване на отговор;
- **Асинхронна комуникация** – обмен на съобщения, при който клиента не се блокира до изчакване на отговор;
- **Комуникация базирана на събития** – получаване на съобщения при настъпване на събития (например webhooks).

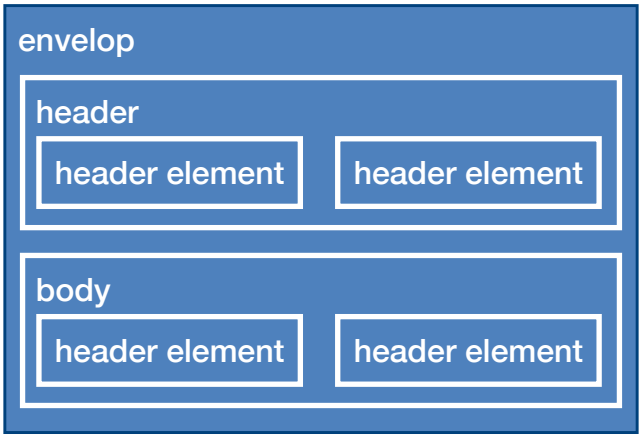
4.3. Стандарти за уеб услуги

4.3.1. SOAP

SOAP (Simple Object Access Protocol) е протокол имплементиран над HTTP или SMTP като цели размяна на съобщения под формата на XML обекти. Той позволява синхронна и асинхронна комуникация през интернет.

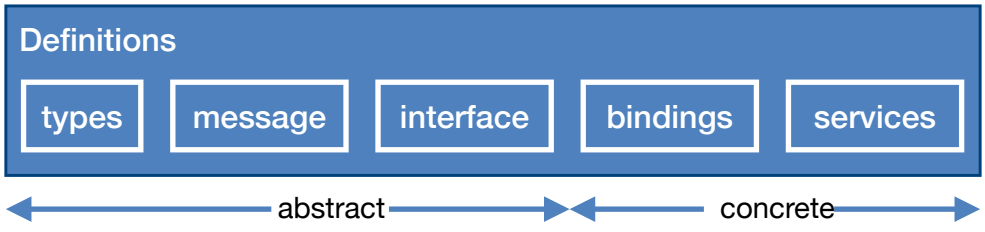
SOAP дефинира:

- структурата на съдържанието на съобщенията в XML;
- схема за обмен на XML обекти при комуникация от тип заявка-отговор;
- схема за обмен на документи;
- правила за обработка на XML елементите от получателите.



4.3.2. WSDL

Web Service Description Language (WSDL) специфицира интерфейса на уеб услуга, както и допълнителна информация за начина на комуникация с нея. Той се използва като допълнение към други протоколи като например SOAP. WSDL е език базиран на XML.



4.3.3. UDDI

Universal Description Discovery and Integration (UDDI) е директорийна услуга, която се ползва от организации, предоставящи уеб услуги. Тя цели да даде достъп на клиентите до тези услуги, описвайки организациите, техните услуги и взаимодействието с тях. Накратко е нещо като телефонен указател.

Тя се състои от:

- **Бели страници** – информация за доставчиците на услуги;
- **Жълти страници** – използват се за класифициране на услугите;
- **Зелени страници** – техническа информация за уеб услугите.