University of Tartu
Institute of Computer Science

MTAT.03.229 – Enterprise System Integration
Service-Oriented Architecture for Plant Hire
Plant Supplier (PS5)

Martin Loginov (A72092) - leader, coding, testing, analysis, design, documentation

Hans Mäesalu (A72107) - coding, testing, analysis, design, documentation

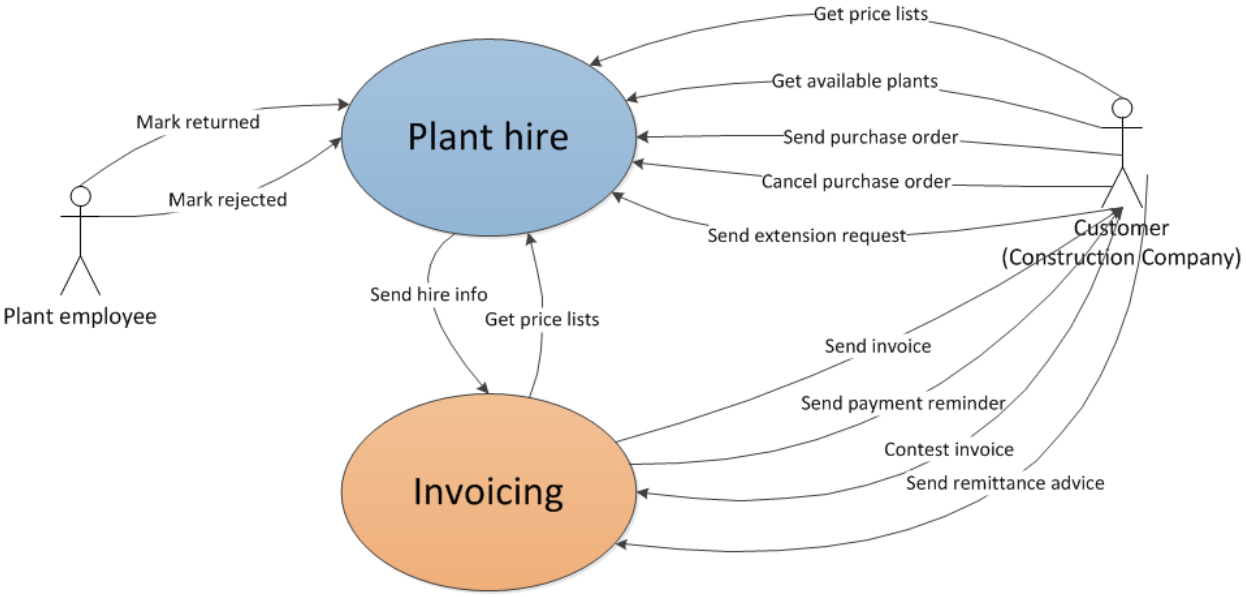
Sven Aller (A60197) - coding, testing, analysis, design, documentation

Tartu 2010

Project description

Our project is a plant supplier system. It consists of plant management and finance management systems. Customers can order plants, cancel the orders, extend orders, when they need plant for longer period, and reject plants, if plant does not fit their needs. When a plant is returned, finance employees can send an invoice to the customer and the customer can then pay the invoice, if everything is fine, or contest the invoice, if there is a disagreement. It is also possible to send invoice reminders, to remind customers of their unpaid invoices.

System has one front-end application for plant and invoice management. It lists all the orders and plant employees can mark plants returned and add message about returned plant condition. Order extension, rejection and cancellation requests are displayed on separate pages where it is possible to accept or decline requests. All invoices are displayed on invoices page. Finance employees change the sum of invoice and send invoice and invoice reminders to customers. Plants are listed on plants page employees can also add new plants.

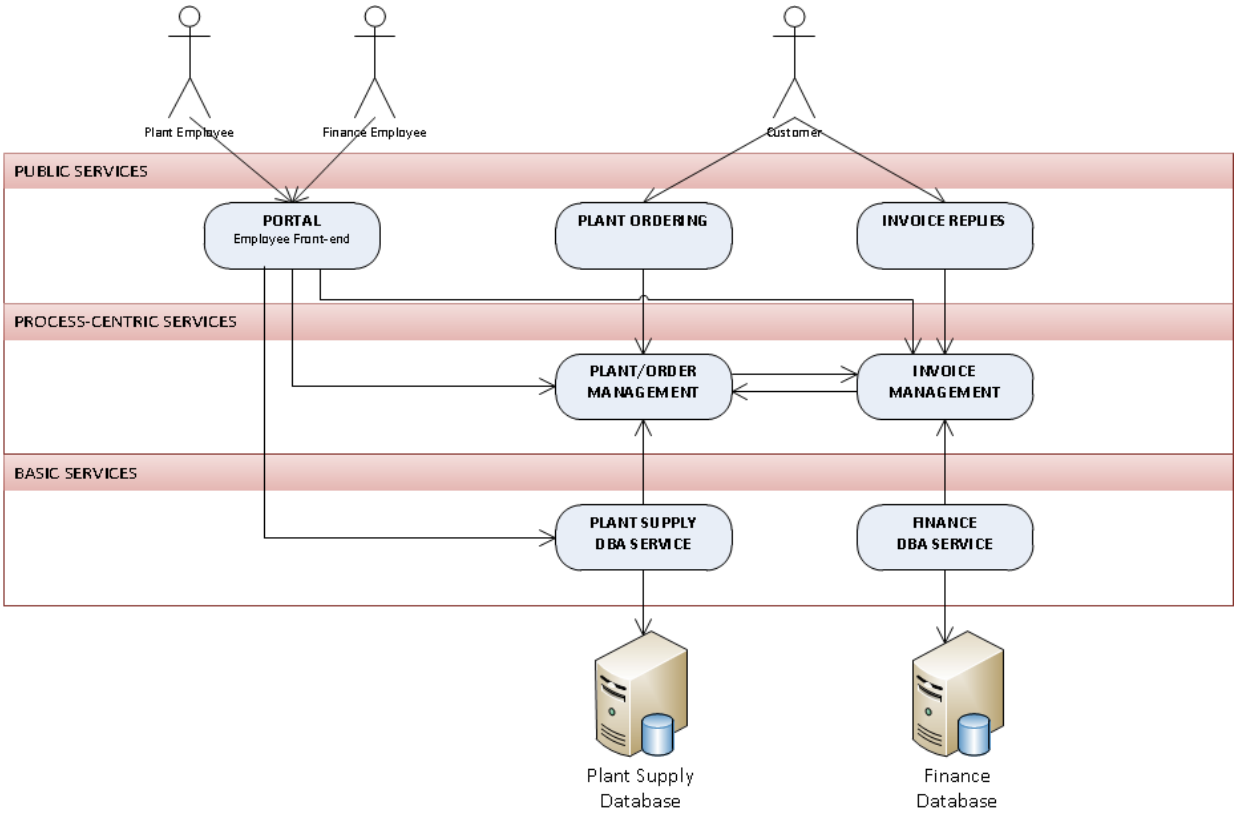


Links

- Service repository: https://sites.google.com/site/ps5repository/
- Front-end application: http://...
- SVN: svn://ats.cs.ut.ee/courses/2010/tvt/PS5

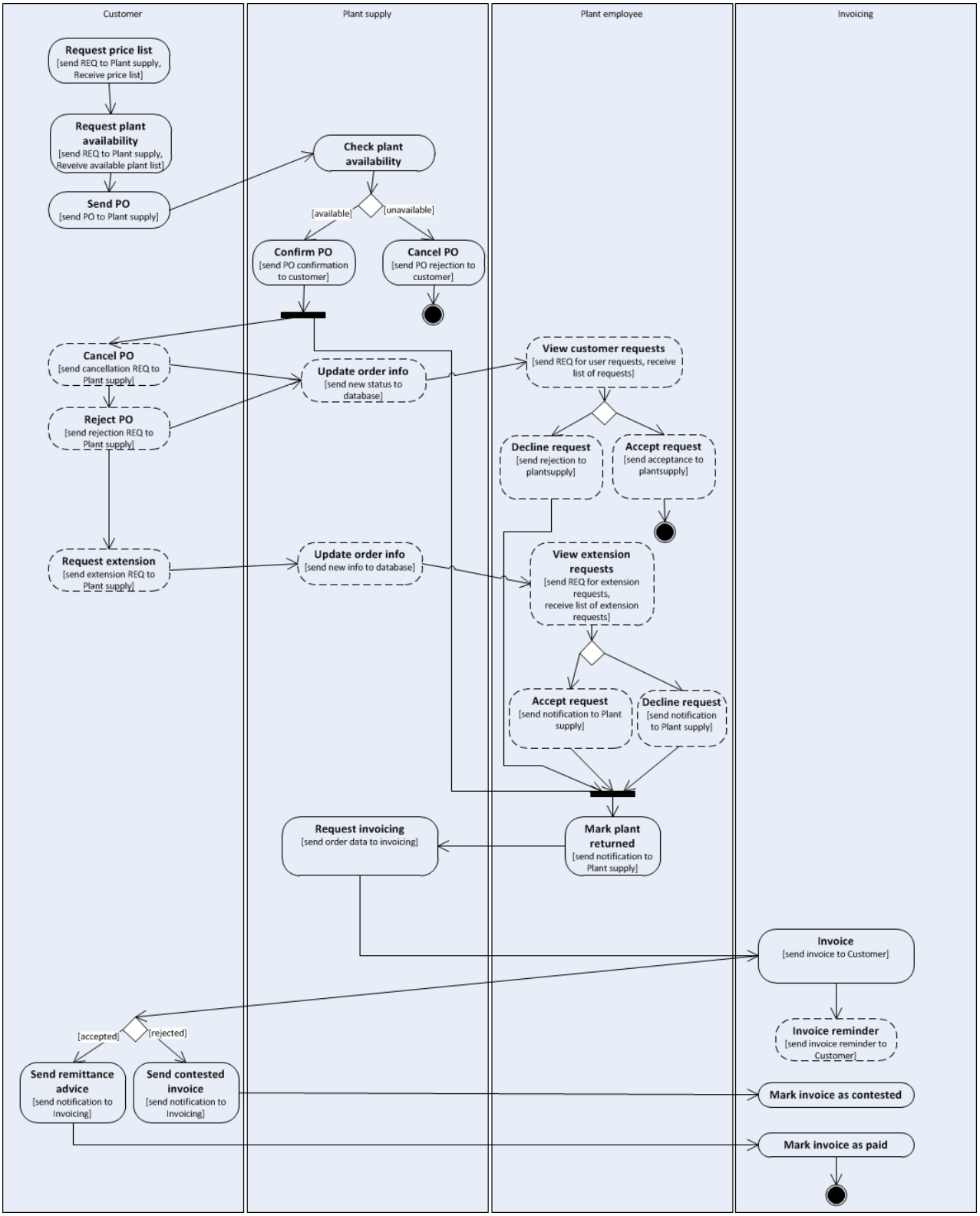
Architecture diagrams

Logical Layered Architecture Diagram



Our logical layered architecture consists of two public services that customers connect to to order plants and notify of payment status. Employee portal connects to internal intermediary and basic services to get necessary information and perform necessary tasks. There are two process centric cervices: plant/ order management and invoice management services. Plant/order management service handles plant and ordering processes. Invoice management handles invoicing process. Two basic data-centric services, plant supply DBA service and finance DBA service, handle database connections.

Choreography



Discussion of design and implementation

Right from the beginning we considered the Plant supply section and Finance section as two separate systems. In our project, plant hiring and invoicing are two completely separate processes implemented as two different process-centric services. Once a plant is returned, data about the rental period is sent to the finance system, which starts the invoicing process.

The plant hiring system is implemented in Java using JAX-WS - it runs on a Linux machine, while the invoicing system is implemented on the .NET platform, using C# and runs on a Windows machine.

All the communication between systems is carried out using SOAP, this includes communication with the partner systems.

The plant hiring system consists of three web services interfaces: Plant Supply WS - for the customers to interact with, Plant Supply Internal - for employee and web front end interaction (eg. approving requests) and Plant Supply DB - a simple data centric service for database access.

The finance section we planned to create in PHP , using NuSOAP for server, but we it seems to be very uncomfortable to create WSDL. Instead we decided to use .NET platform with C# and SQL Server database. Tests with PHP SOAP clients were useful: it was simpler to create the front-end in PHP to use with the services later. For financial department we made three services with several methods. One of this is for internal communication with hiring system (Invoice Management) for adding, reading and updating invoices and sending reminders. The other (Finance Services) is for CC partners systems and serves invoice contesting and remittance services. The third service (Finance Dba Service) is used for communicating with finance department database for reading invoices.

Front-end application is built in PHP. It uses SOAP for connecting to services to read data and perform other necessary actions. PHP makes it easy to quickly develop web applications, so it was perfect in our situation where we did not have much time. We have single front-end application for plant supply and finance systems.

For the integration with our partner systems, we initially planned to use asynchronous communication with both systems, however we couldn't get it working on the Plant supply system, therefore we had to redesign our Plant supply system so that the CC-s could poll the statuses of their orders instead.

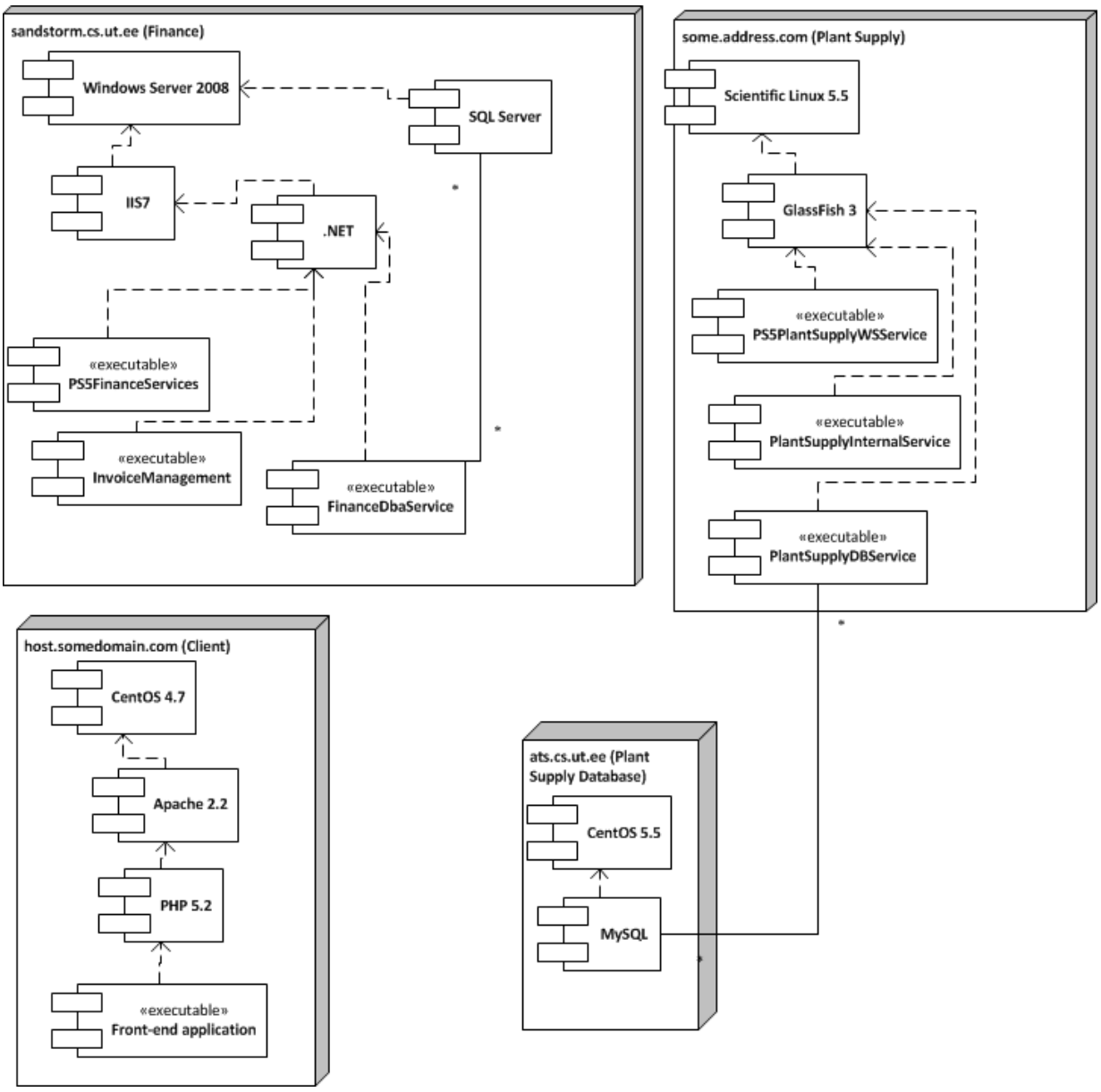
The finance system interacts with the CC systems asynchronously - sending invoices and reminders and allowing them to send contest requests or remittance requests back.

All together we integrated with both the required teams CC5 and CC6 and managed to get our systems working.

Problems

- unstable servers, problems in configuration of server
- type casting between languages and different databases
- lack of time, short period for implementation

Deployment diagram



Services and applications

Service / application	Platform	Language	Technologies	Complexity	Author(s)
PS5PlantSupplyWSService	Scientific Linux 5.5	Java	SOAP	complex	Martin Loginov
PlantSupplyInternalService	Scientific Linux 5.5	Java	SOAP	moderate	Martin Loginov, Hans Mäesalu
PlantSupplyDBService	Scientific Linux 5.5	Java	SOAP	easy	Martin Loginov
PS5FinanceServices	Windows Server 2008, .NET	C#	SOAP	moderate	Sven Aller
InvoiceManagement	Windows Server 2008, .NET	C#	SOAP	moderate	Sven Aller
FinanceDbaService	Windows Server 2008, .NET	C#	SOAP	easy	Sven Aller
Front-end application	CentOS 4.7	PHP		moderate	Hans Mäesalu