An Internet Banking Framework with Perl

Javier Rodríguez, Carlos de la Guardia.
Aldea Internet, México.

The Challenge

- In the summer of 1997, Banco del Atlántico (a Mexican bank) asked us to develop a web front end for their remote banking application
- Our mission: to create a middleware capable of communicating with a session-oriented COBOL application running in a UNISYS mainframe, which could interface with a stateless web server
- Obstacles included a small time frame and the need to interact with old applications, coded in different languages and very poorly documented

Our goals

- Extensibility
  - implement and integrate new business functions easily
- Scalability
  - Handle huge numbers of users
  - Work with multiple servers if necessary
- Security
  - Secure server and protected CGI execution environment
- Portability
  - Easily migrate to other Operating Systems
  - Database engine independence

Our Approach

- Use of a Modular Architecture
  - hide the implementation details from the main application and each other
  - encapsulate tasks like communication with legacy systems and business logic into separate modules
- Use of persistent connections to solve the ‘stateless protocol’ problem (fastcgi was the tool of choice)
- Use of a process manager to route petitions to the first available server process, to add new servers easily as the number of customers required it
- Use of existing code wherever possible to improve development cycle time

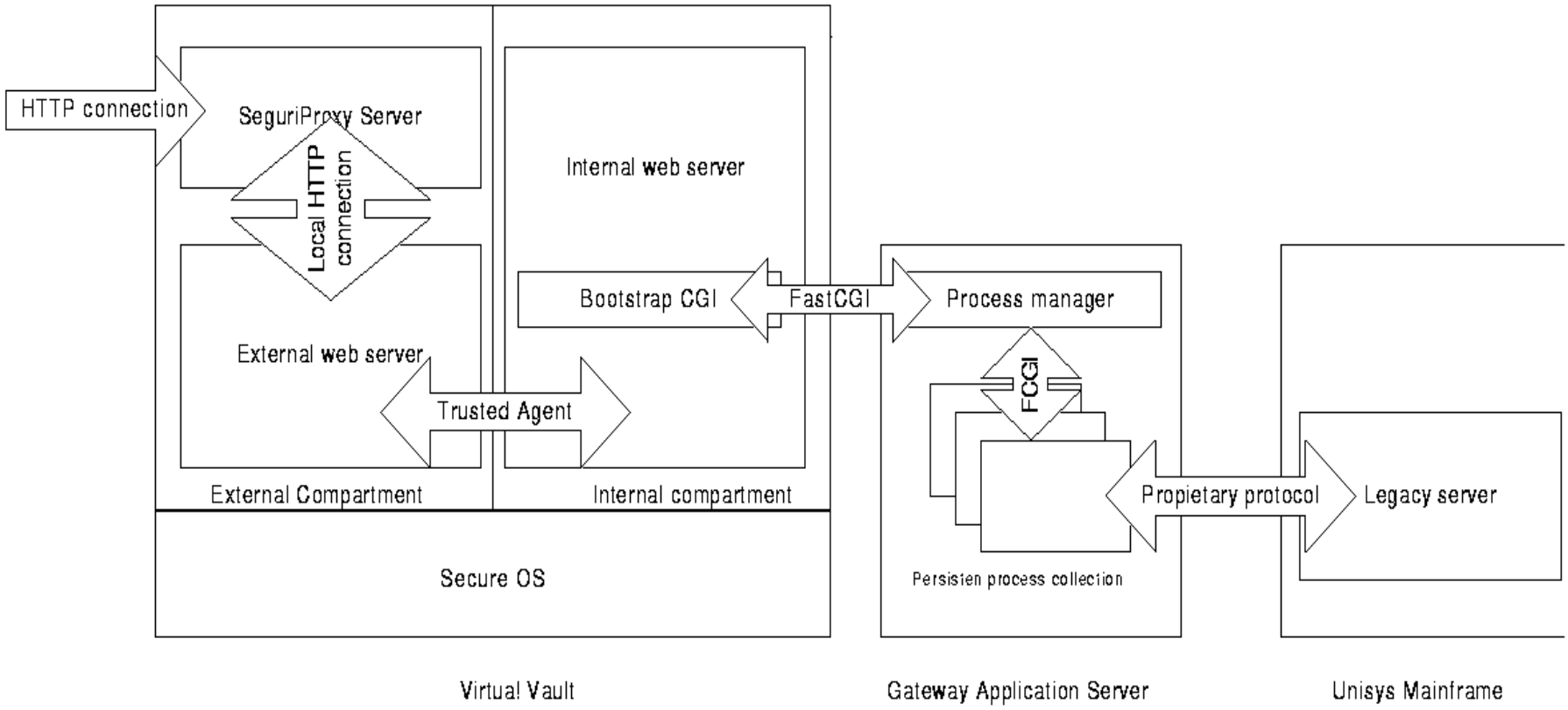
Why Perl?

- Deciding to use Perl is easy, convincing people in corporate circles that it is the way to go is a lot harder
- Perl has a lot of advantages meaningful to our project:
  - Short development cycle
  - Maintainability
  - Excellent language features
  - Code reuse through the addition of existing modules
  - We love Perl!
- There were a couple of drawbacks too:
  - No multithreading
  - Considerable footprint

Perl in Action

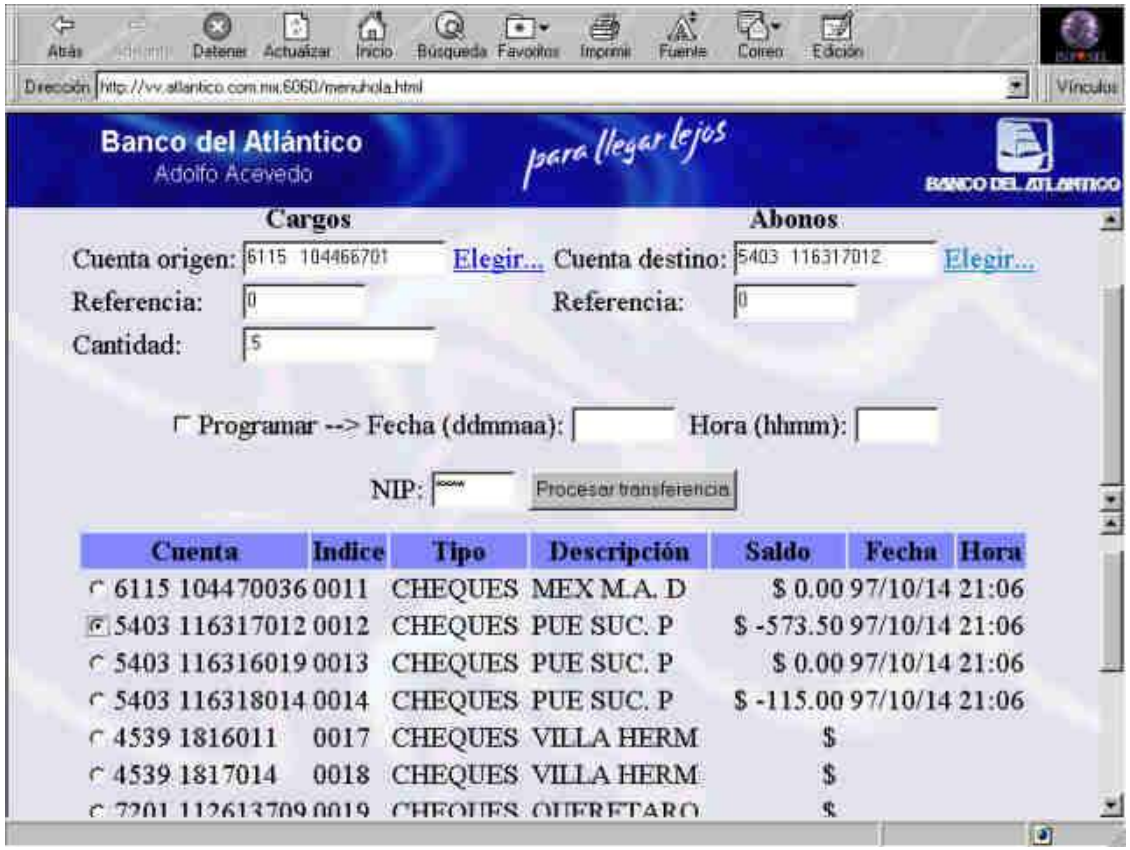
- We developed a low-level communications module to talk directly to the legacy server (telnet.pm came in handy)
- We also developed a mid-level operations module to encapsulate all needed banking operations
  - A single line of perl code in the main CGI script could then commit a complete transaction: $message=commitTransfer($nip,$date,$time,\%source,\%destination);
- The process manager, the tool to interface with session based legacy systems using stateless protocols, was also programmed in Perl
- The use of DBI, CGI.pm and other modules simplified our work
- Quickly assembled debugging tools later became monitoring applications for the system's managers

Architecture



The User Side

- The user interacts with the application using a web browser
- All user operations are sent to the Mainframe via a CGI script which uses the developed modules
- The proccess manager takes care of asigning the connections



Javier Rodríguez, jrodrigu@aldea.com.mx.
Carlos de la Guardia, cguardia@aldea.com.mx

Conclusion

- The project proved to many unbelievers that Perl can really offer solutions to complex problems in the real world of financial institutions
- You can save time and money using Perl, and also develop a strong and scalable solution
- The extensibility we achieved using Perl, allowed us to extend the application to interact with point of sale equipment and serve as a front end for a ‘mini bank’ in a little over a month!
- As a sad colophon to this story we must state that the Banco del Atlántico was bought by another bank shortly after the project was completed. The new administration dropped the project completely