Migrating Legacy Systems to the Web: Challenges for Migration to SOA

Andrea De Lucia

Software Engineering Lab
Department of Mathematics and Informatics
University of Salerno

adelucia@unisa.it

Outline

- Motivations
- System Decomposability and Migration to SOA and Web based environments
- Migrating nondecomposable systems to the web: strategy, tool support, and experiments
- Conclusion and discussion points

Motivations for Migration

- Legacy Systems are business critical, but
  - Developed in some obsolete language/technology
  - Lack of interoperation with other applications
- Needs
  - business process reengineering
  - accessing legacy systems through the web
- Mass replacement of a LIS too risky
  - high effort required to develop the system from scratch
  - lack of documentation
  - business logic encoded in the programs
  - original developers long since departed
- Wrapping and incremental migration represent a viable alternative
  - The migration strategy is strongly influenced by the system decomposability

Migrating Legacy Systems to SOA

- Reverse engineering the legacy system (and not only it …) to abstract the underlying business process
  - The process can be automated with workflow management technologies
- Extracting the legacy components implementing the business functions and encapsulating them into services
  - These can be orchestrated by the workflow
- Business Process Reengineering is now much easier, but … is this always feasible ?

Challenges for Migrating to SOA

- Technical challenges …
  - Unless the system is decomposable, costs and risks of migrating to SOA are very high
  - Unfortunately, most legacy systems are nondecomposable
- Managerial challenges
  - In most cases, the primary goal of the owner of the system is just to get the system running on the web, by changing as least as possible
  - Organizations are not keen to spend money and take risks just to have a better and more flexible software architecture

Project METAMORPHOS: Survey conducted with Italian Companies

- Architecture migration
  - 63% -> Web
  - 15% -> C/S
  - 11% -> SOA
- UI migration
  - 50% Traditional GUI -> Web UI
  - 39% Text UI -> Web UI

System decomposability (1)

- The main technical factor affecting legacy system migration
- Decomposability into Layers
  - User Interface
  - Application Logic
  - Data Management
- Decomposability into Partitions
  - Each partition implements a clear functionality
  - Each partition is loosely coupled with other partitions

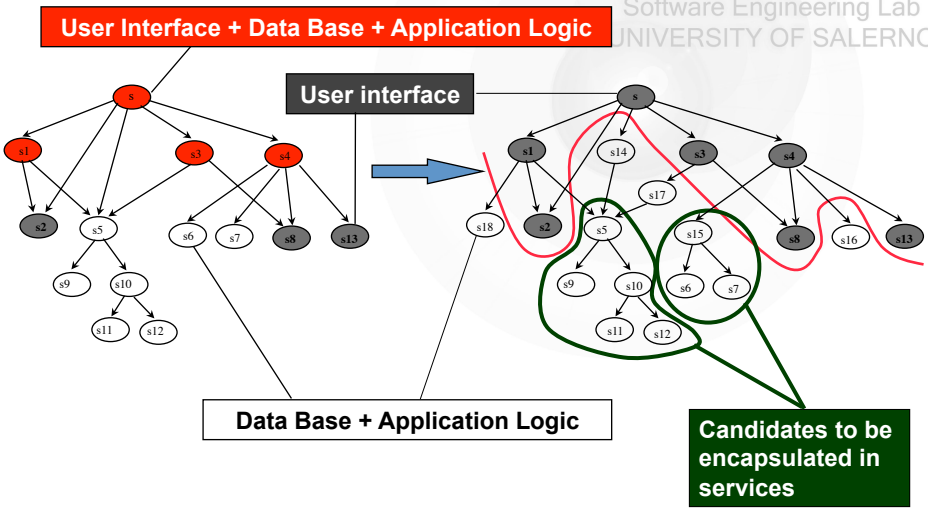
System decomposability (2)

- Decomposable systems
  - The application logic components are independent of each other and interact with the data management and user interface components
  - Both decomposition into layers and partitions
- Semidecomposable systems
  - Only the user interfaces are separate components, while application logic components and data management services are not separated
- Nondecomposable systems
  - The system is a black box with no separated components (the worst architecture)

System decomposability and legacy system migration to SOA

- Migrating to SOA requires the system be decomposable
  - Application logic and database components can be encapsulated and migrated to services
- For semi-decomposable system at least a good decomposability into partitions is required
  - However, separating the application logic components from the data management components is also required
- For non-decomposable system migrating to SOA is quite unfeasible
  - The system architecture is preliminarily restructured to a client server style …

Restructuring to semi-decomposable systems



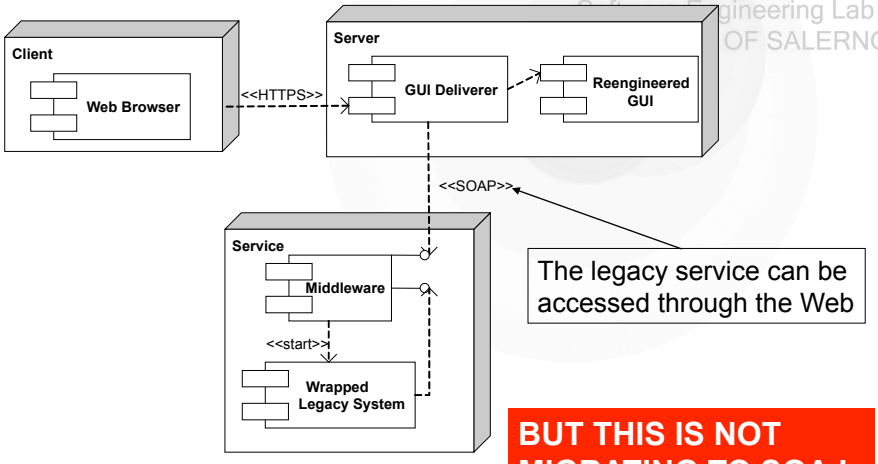
Project METAMORPHOS: our experience

- Transferring migration technologies to industry
  - Partner company: an Italian small software enterprise
  - Goal: identifying the best strategy and supporting technology to migrate the legacy systems of the company (written in ACUCOBOL-GT) towards a web based architecture
- Adopted Methodology
  - Selecting and assessing the most meaningful legacy systems of the organization
  - Defining and testing the migration strategy
  - Developing tool support for the migration process
  - Conducting controlled experiments and case studies
  - Releasing the tool to the company and training the personnel

Migration Strategy

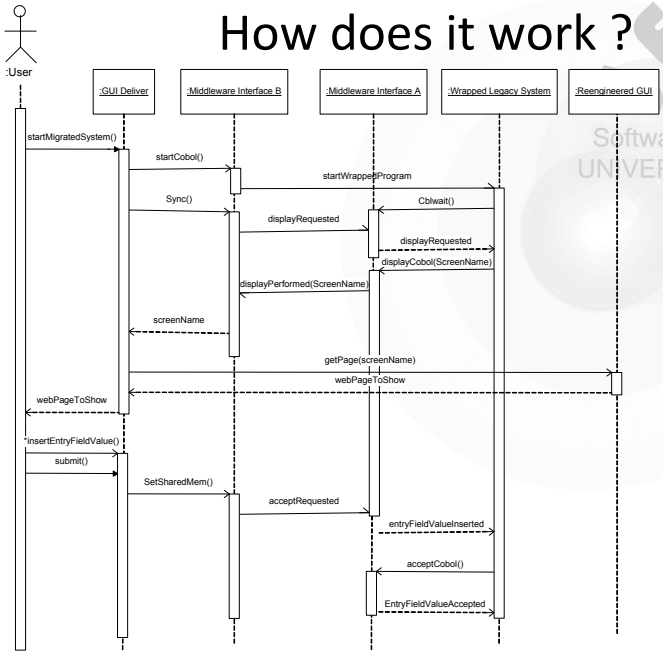
- Results of the legacy system assessment:
  - Low decomposability degree and spaghetti-like code
  - Embedded control flows (BEFORE and AFTER statements) in the user interface components
- Defined a migration strategy for non-decomposable systems
  - Reengineering the user interface
  - Wrapping the legacy system at the user interface level
  - Using a communication middleware developed as a DLL
- Further Problems and Risks
  - Embedded Control flows in the user interface component prevents a fully automated migration process

Target Architecture for Legacy System Encapsulation

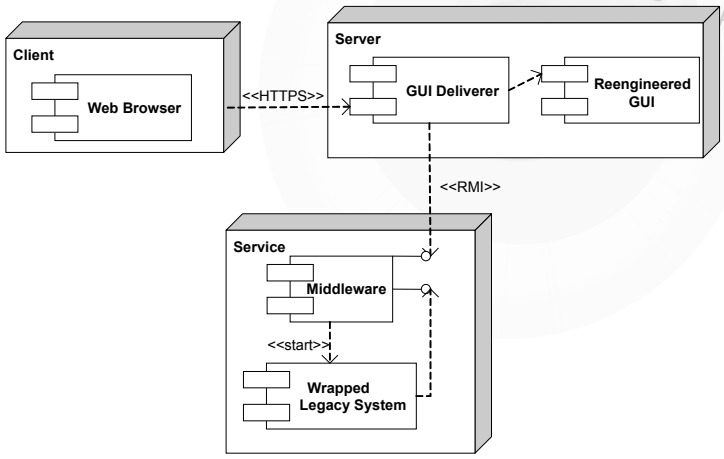


BUT THIS IS NOT MIGRATING TO SOA !

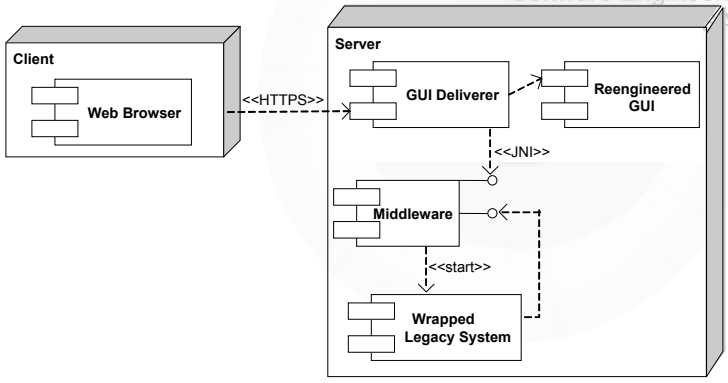
How does it work ?



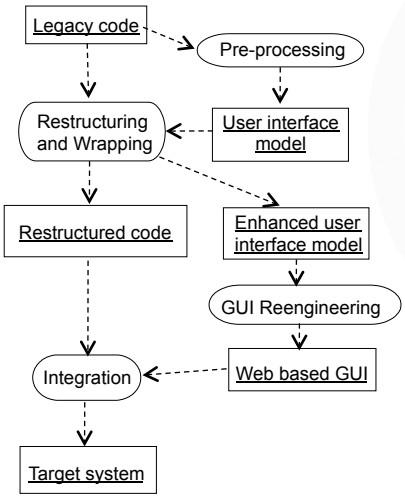
Other options: Accessing the service within the Intranet



Other options: Web server and Service on the same node

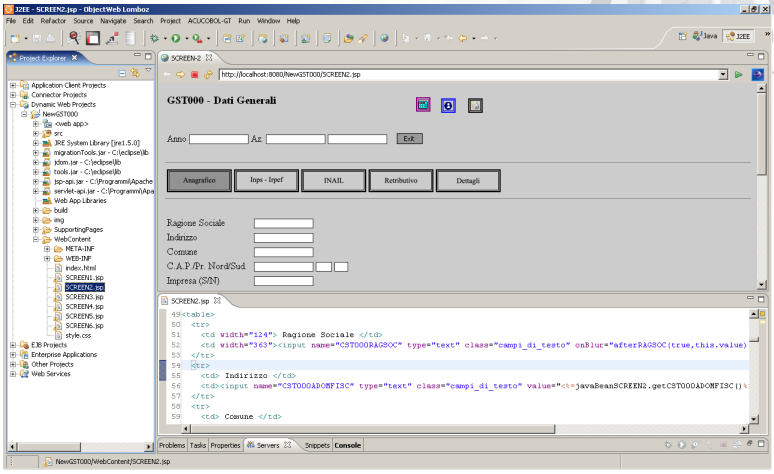


Migration Process



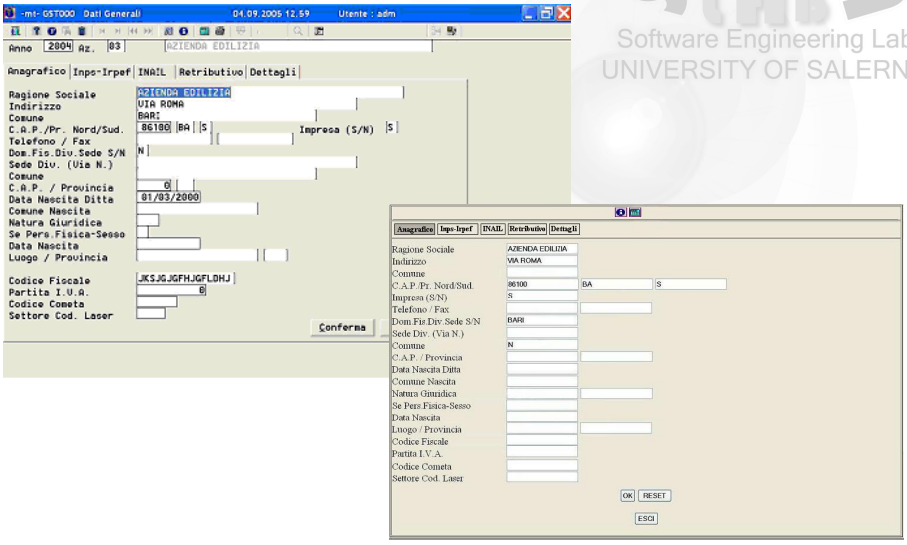
Semi-automatic process

Tool Support

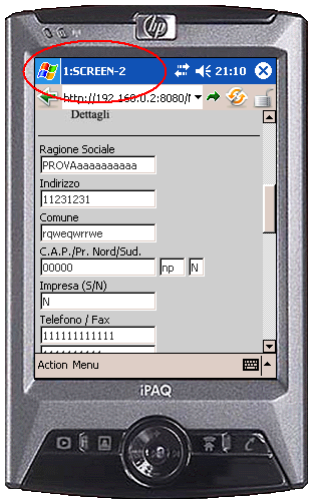


MELIS (Migration Environment for Legacy Information Systems)

Original and Migrated GUI



Migrated GUIs on mobile devices



Controlled experiments

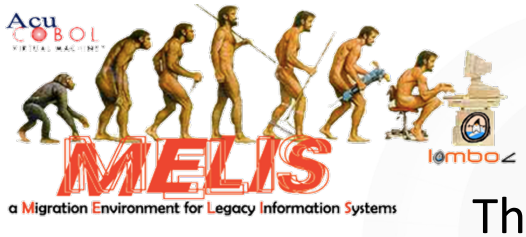
- First experiment: 28 master students
  - randomly grouped in 14 development teams
- Replicated experiment: 4 practitioners (COBOL experience) and 4 junior researchers (J2EE experience)
  - grouped in 4 teams, each composed of a practitioner and a junior researcher
- Same counterbalanced design in the two experiments
  - Each team used MELIS in one migration task and traditional development tools in another migration task
- Main Results
  - MELIS improves the productivity of four or five times
  - MELIS reduces the experience gap, by providing more benefits for less expert software engineers

Case studies

- Selected the most business critical legacy system of the partner company
  - More than 600 KLOCs and more than 500 programs
- Migrating meaningful subsystems of the selected systems
  - LOCs ranging from 4000 up to 14000
- Conducted with mixed teams of academic and practitioners
  - Each team used MELIS in one migration task and the traditional development tools in another migration task
- Improvement in productivity of seven or eight times

Conclusion

- Migrating legacy systems to SOA is hard
  - In particular for nondecomposable legacy systems
- More often companies only ask to access their systems from the web
  - Not really interested in just reengineering the software architecture
- Discussion Points
  - Is it really a need (or an advantage) migrating legacy systems to SOA ?
    - Or is migration to the web enough ?
  - If migrating to SOA is needed:
    - Is it possible to define cost-effective migration strategies to SOA for nondecomposable legacy systems ?
    - How to convince managers of the advantages of such strategies ?



Thank you !

Comments / questions ?

Contact Information:
Andrea De Lucia
Dipartimento di Matematica e Informatica
Università di Salerno - Via Ponte Don Melillo
84084 Fisciano (SA)
email: adelucia@unisa.it