TDDD05
Component-Based Software

Evaluation:
Black-Box Component Systems

- **COTS summary** [Szyperski: Chapter 17]
- **COTS evaluation as a composition system**
  U. Assmann: *Invasive Software Composition*, Springer 2003, Section 3.3
- **Empirical evaluation of COTS systems**
  C. Karlsson: *Designing an Experiment to Compare Component Systems*. Master thesis, MSI, Växjö university, Sweden, 2006.
  (Draft by courtesy of Welf Löwe, MSI, Växjö university)

Christoph Kessler, IDA,
Linköpings universitet

Components-off-the-shelf (COTS) systems

- CORBA, Java RMI, EJB, COM, .NET, Web Services

Common features: object-based component model + adapting / glueing

- Encapsulation
  - CORBA objects, EJB containers, COM interfaces
- (Remote) Method invocation
- Support for late binding and brokering
- Abstraction from component implementation language
- Abstraction from component location
- Common communication formats (Java serialization/deser.; IIOP)
- Common shipping formats (CAR files, JAR archives, CLI assemblies)
- Meta-information (introspection, repositories)
- Support for persistence, serialization
- Support for property management
- Support for events / asynchronous communication

COTS: Evaluation as a composition system (1)

Component model

- Mechanisms for secrets and transparency: *Very good*
  - Interfaces and implementation repository
  - Component language hidden (interoperability)
  - Component location hidden
  - Life-time and identity of service hidden
- No parameterization of components
- Standardization: *Fairly good but many standards*
  - standardized interfaces
  - standardized services

COTS: Evaluation as a composition system (2)

Composition technique

- Mechanisms for connecting components: *Strong*
  - Mechanisms for adapting and glueing components
    - IDL, stubs, skeletons, object adapters
    - Or common binary data format + similar programming languages
  - But binding time / mechanism (static / dynamic invocation) is hardcoded, cannot be exchanged automatically: *Weak scalability*
- Mechanisms for aspect separation: *Weak*
  - Multiple interfaces
- Nothing for extension of components ("Black-box")
- Mechanisms for meta-modeling and introspection

COTS Evaluation as a composition system (3)

Composition language:

- *Weak for classical COTS systems*
  - CORBA IDLscript provides a facility to write glue code, but only black-box composition
  - Similar: VisualBasic scripts for glueing COM components
- For Web Services: strong composition language BPEL
  - (still black-box composition only)

Empirical evaluation of 4 COTS systems

- Master thesis project at Växjö university, 2006
- Simple, small application: Address data administration

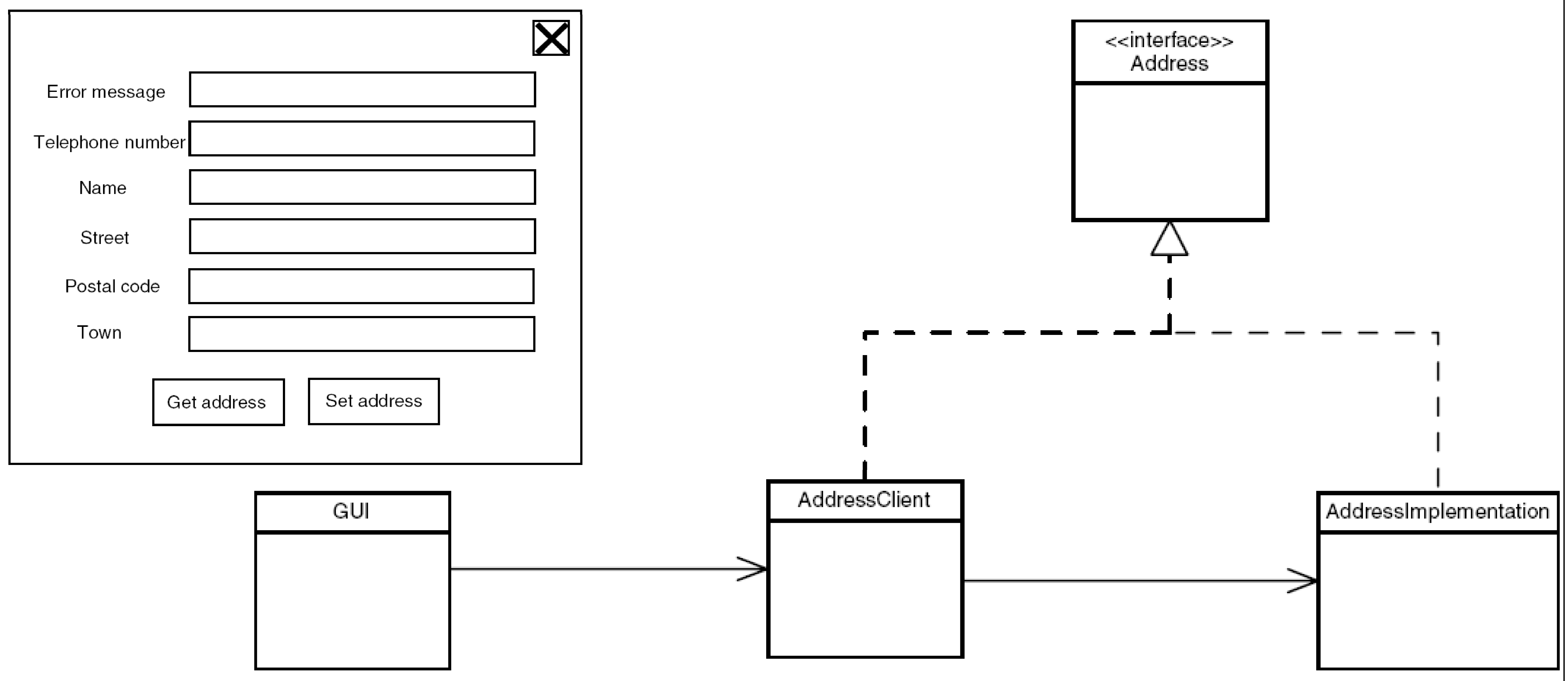


Image source: C. Karlsson, Master thesis, MSI Växjö university, 2006.

Figure 2: An UML class diagram of the local solution.

Server implementation (in Java)

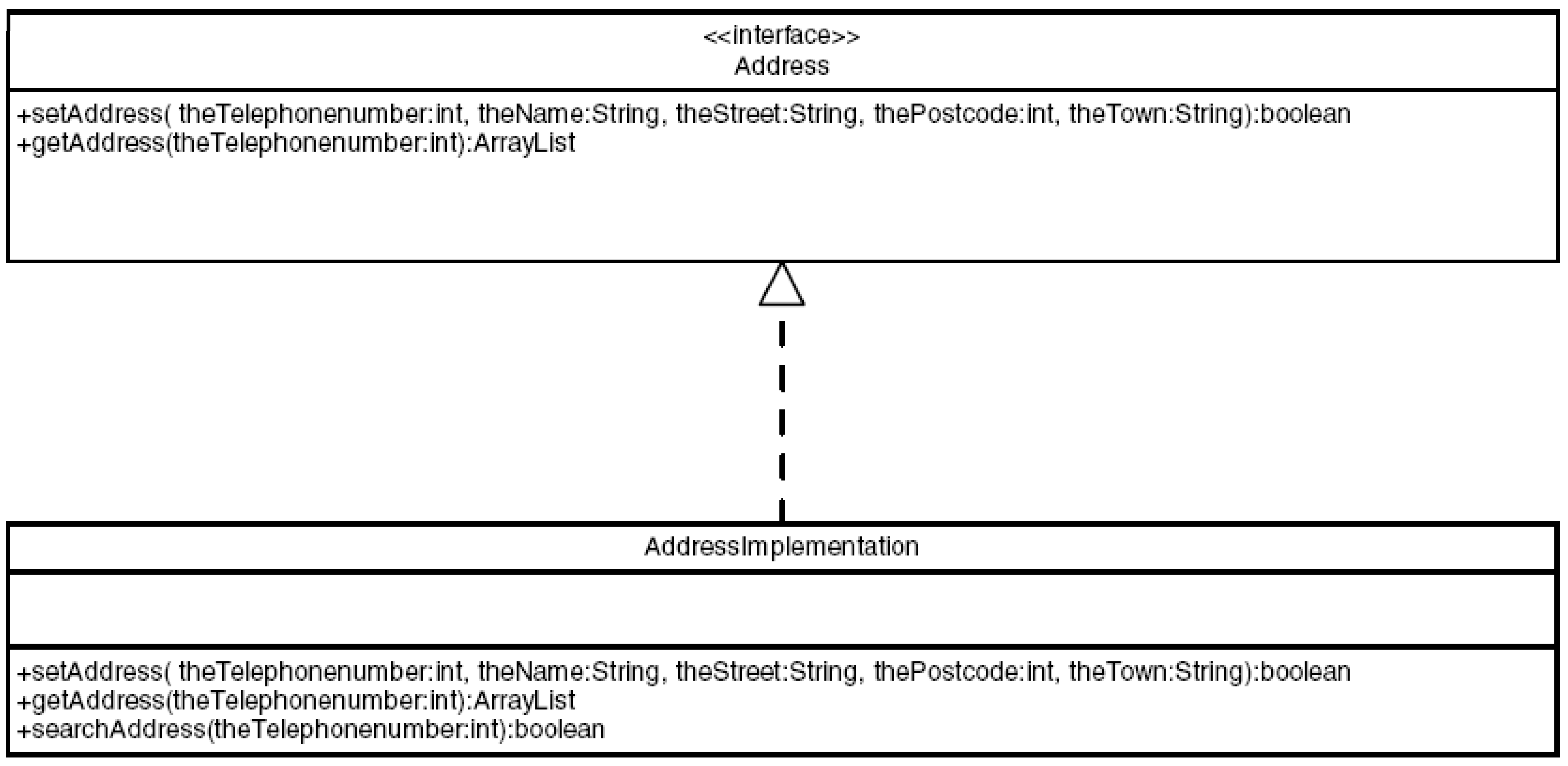
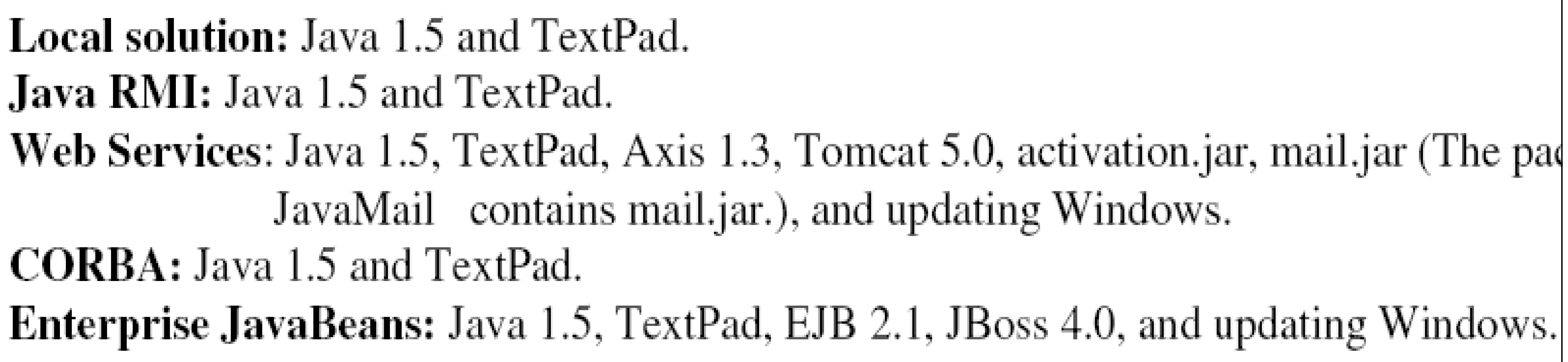

<<interface>>
Address
+setAddress(theTelephonenumber:int, theName:String, theStreet:String, thePostcode:int, theTown:String):boolean
+getAddress(theTelephonenumber:int):ArrayList
AddressImplementation
+setAddress(theTelephonenumber:int, theName:String, theStreet:String, thePostcode:int, theTown:String):boolean
+getAddress(theTelephonenumber:int):ArrayList
+searchAddress(theTelephonenumber:int):boolean

Figure 9: Class diagram of the AddressImplementation class.

Image source: C. Karlsson, Master thesis, MSI Växjö university, 2006.

Empirical evaluation of 4 COTS systems

- Implemented as
  - local solution (Java)
  - for 3 COTS systems and
  - as Web service



Local solution: Java 1.5 and TextPad.
Java RMI: Java 1.5 and TextPad.
Web Services: Java 1.5, TextPad, Axis 1.3, Tomcat 5.0, activation.jar, mail.jar (The pa
JavaMail contains mail.jar.), and updating Windows.
CORBA: Java 1.5 and TextPad.
Enterprise JavaBeans: Java 1.5, TextPad, EJB 2.1, JBoss 4.0, and updating Windows.

Image source: C. Karlsson, Master thesis,
MSI Växjö university, 2006.

Evaluation (1)

Comparison between the total time to learn, install, and implement

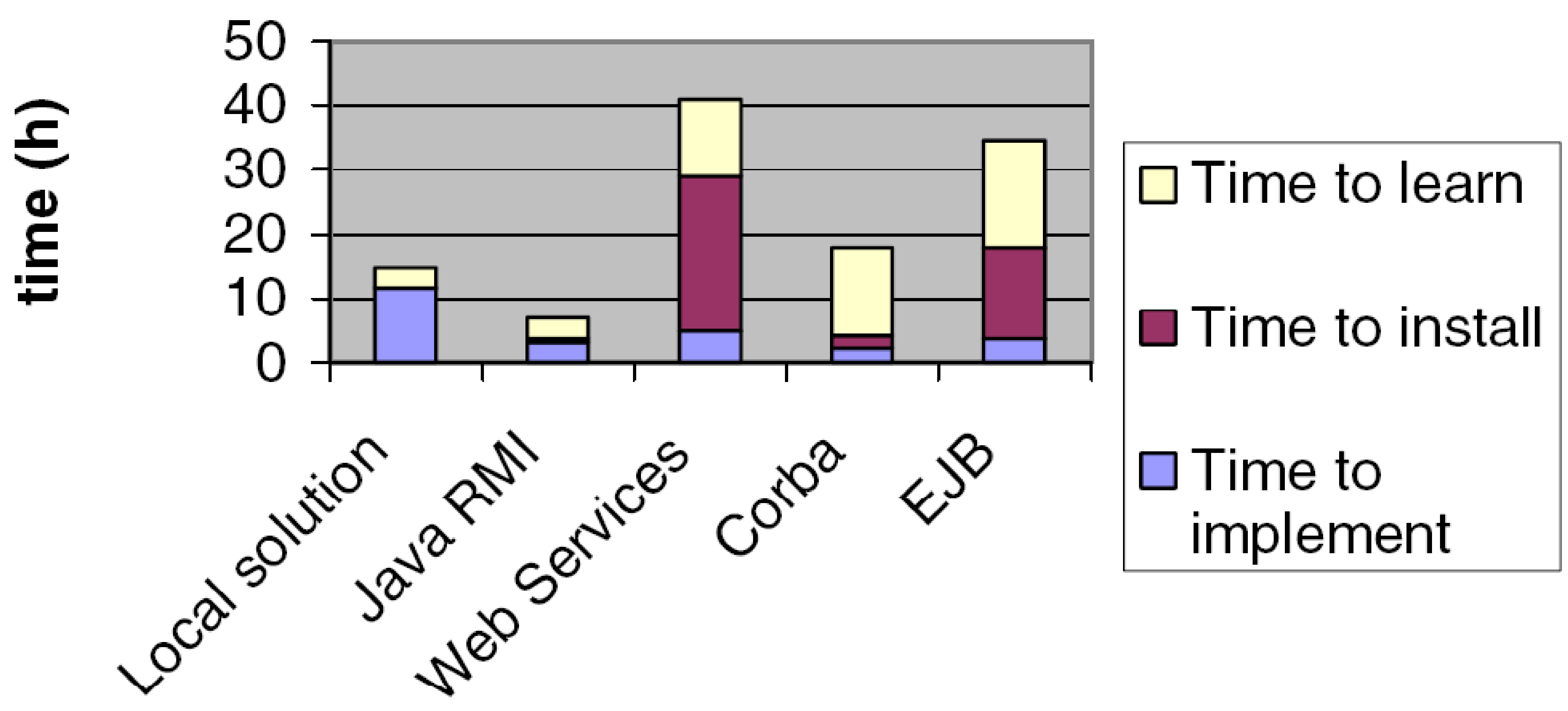


Image source: C. Karlsson, Master thesis, MSI Växjö university, 2006.

Evaluation (2)

The number of logical and compile time errors

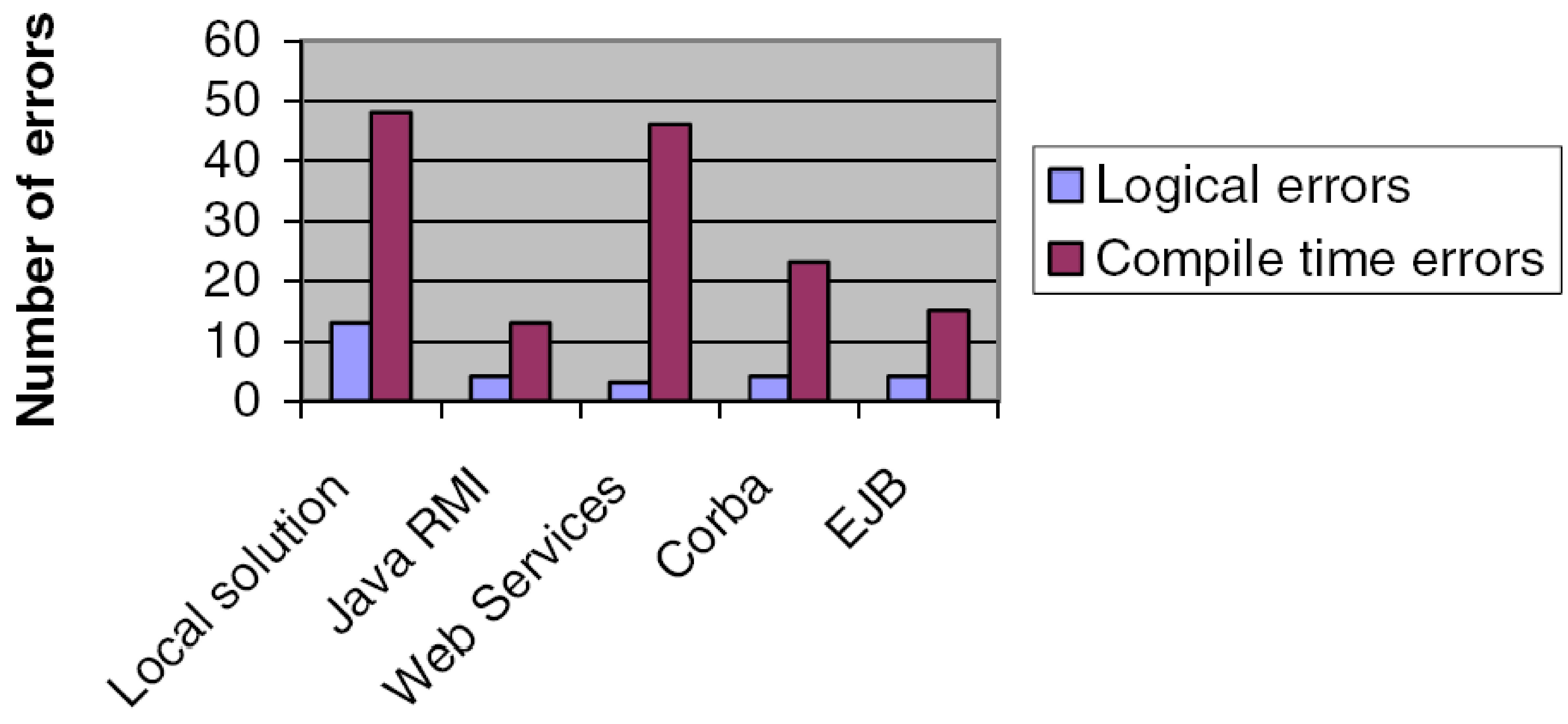


Image source: C. Karlsson, Master thesis, MSI Växjö university, 2006.

Evaluation (3)

- Runtime overhead

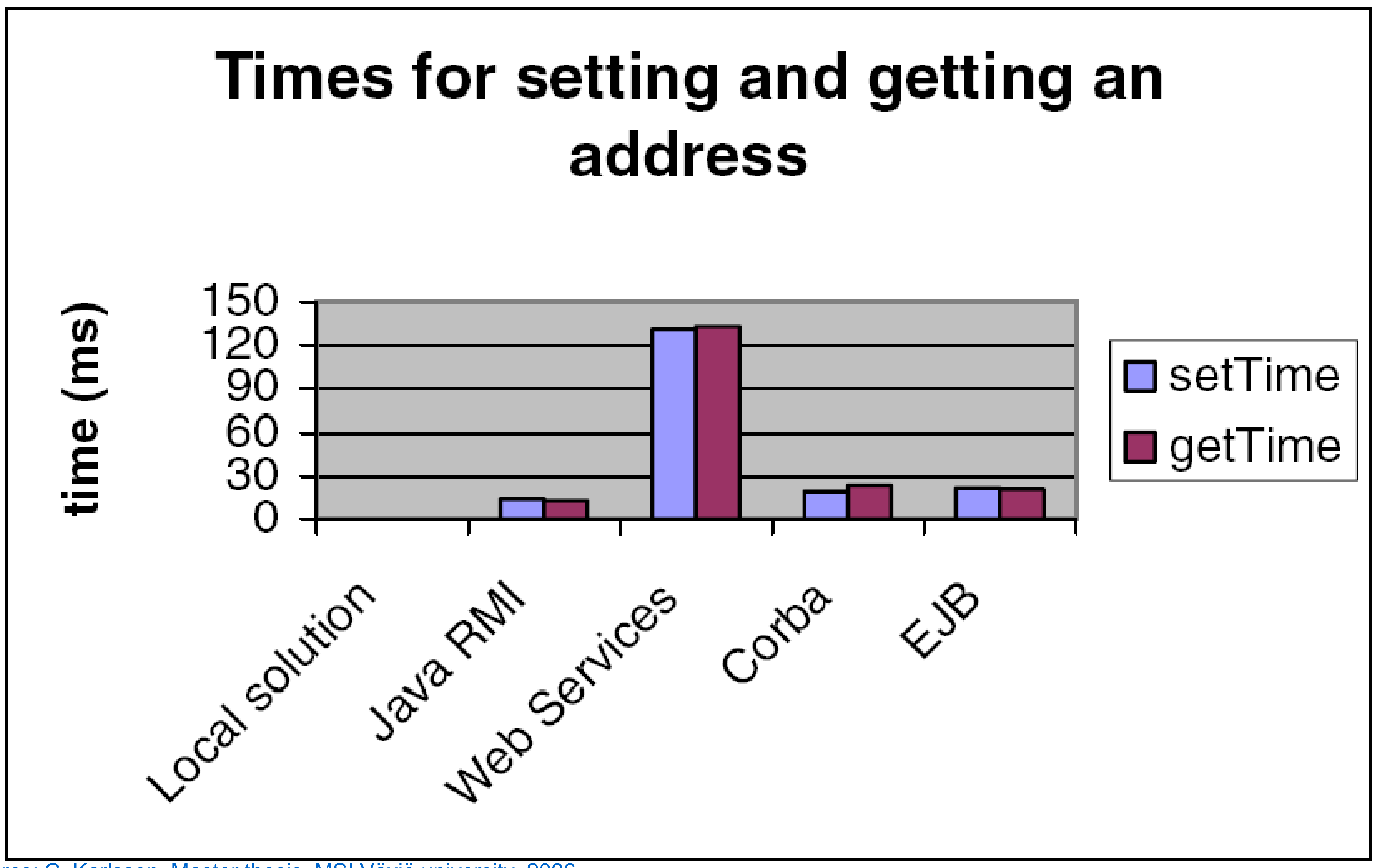


Image source: C. Karlsson, Master thesis, MSI Växjö university, 2006.

Evaluation (4)

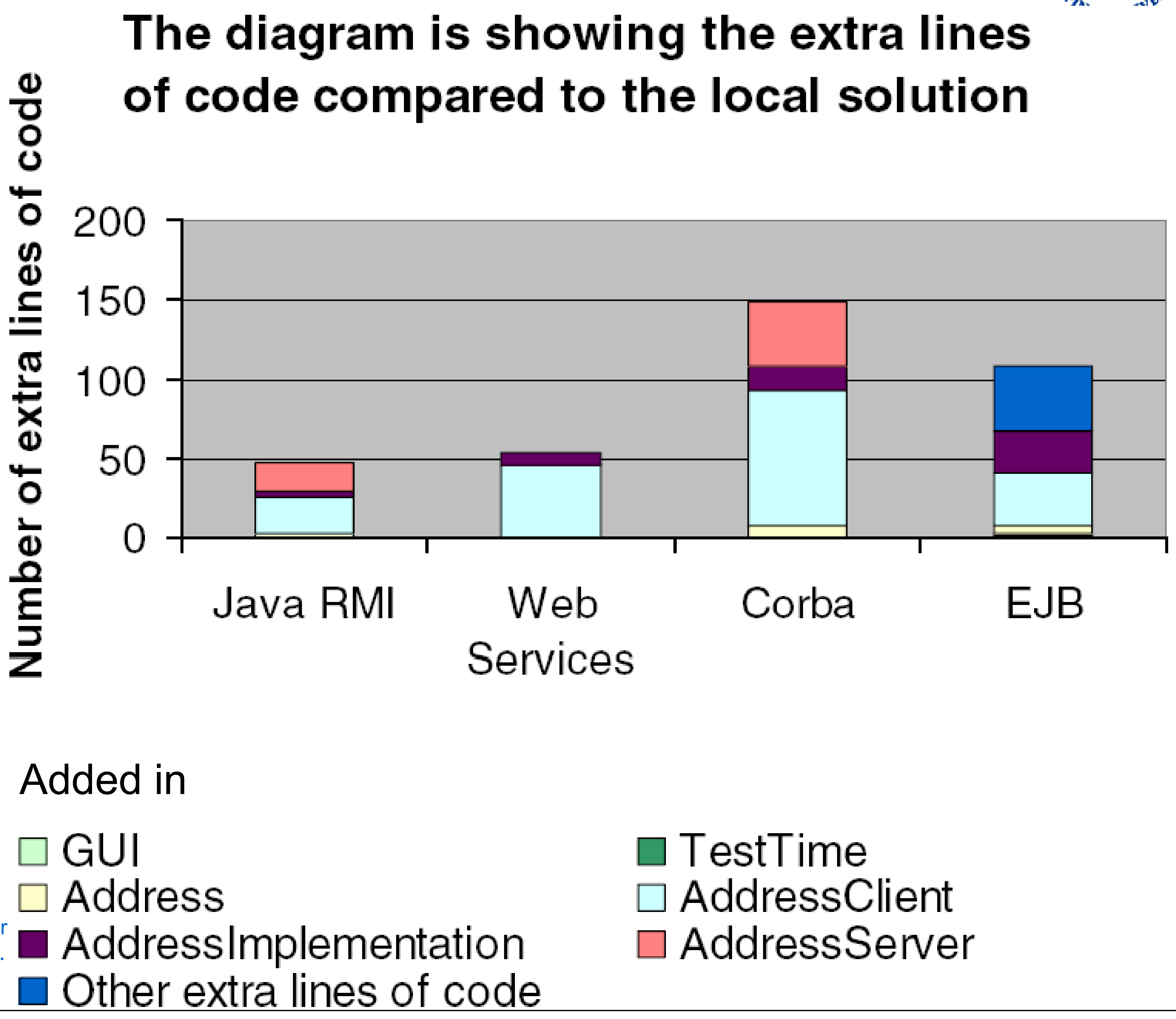


Image source: C. Karlsson, Master thesis, MSI Växjö university, 2006.