## Document Information

<table>
<thead>
<tr>
<th>Document title:</th>
<th>GRADY REPORT I : EJB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document file name:</td>
<td>GradyReport_¹ÚÇöö_.doc</td>
</tr>
<tr>
<td>Revision number:</td>
<td>&lt;0.2&gt;</td>
</tr>
<tr>
<td>Issued by:</td>
<td>&lt;⁵¹ ¿¹ &gt; <a href="mailto:hpark@kebi.com">hpark@kebi.com</a></td>
</tr>
<tr>
<td>Issue Date:</td>
<td>&lt; Dec. 22, 2000&gt;</td>
</tr>
<tr>
<td>Status:</td>
<td>continue</td>
</tr>
</tbody>
</table>

## Content Information

<table>
<thead>
<tr>
<th>Audience</th>
<th>J2EE Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>EJB</td>
</tr>
<tr>
<td>Reference</td>
<td><a href="mailto:hpark@kebi.com">hpark@kebi.com</a></td>
</tr>
<tr>
<td>Benchmark information</td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents

"Grady Report I: EJB"................................. 4

J2EE................................................................................................................... 6
  .................................................................................................................. 9
J2EE Best Practices and Guidelines................................................................. 11
EJB (Enterprise JavaBeans) ............................................................................. 19
  EJB Framework............................................................................................. 21
  Session Beans ............................................................................................... 21
  Entity Beans ................................................................................................. 27
  Transactions in EJB ..................................................................................... 31
  EJB Security .................................................................................................. 34
Design Patterns in Application ........................................................................... 36
  Factory Pattern ............................................................................................. 40
  Facade Pattern .............................................................................................. 41
  Singleton Pattern .......................................................................................... 42
  Reflection Pattern ......................................................................................... 43
  MVC Pattern .................................................................................................. 43
  Observer Pattern ........................................................................................... 44
  Mediator Pattern ........................................................................................... 45

.................................................................................................................. 47

.................................................................................................................. 51
  : XP (eXtreme Programming) ........................................................................ 51
  : ? ............................................................................................................... 55

.................................................................................................................. 56
"Grady Report I: EJB"

(2EE Java 2 Platform, Enterprise Edition)

Security, Transaction, Naming, Concurrency, Persistence
J2EE

J2EE is a Java Technology (Sun) for developing distributed applications. J2EE Standard Edition J2SE, Enterprise Edition J2EE, and Micro Edition J2ME are specifications for different application environments. .NET Framework is a competitor to J2EE.

Components, Containers, Connectors. J2EE is defined by the following components: Application Client, Applets, Web Components, Business Components, HTML, Servlets, JSPs, EJBs, EIS, JTA, JRF, JMS, JMF, JAF, JDBC. J2EE also includes aspects such as Availability, Capacity, Extensibility, Flexibility, Manageability, Performance, Reliability, Scalability, Security, Testability, and others.
EJB Load Balancing Router JNDI Visibility, Reliability, Availability, Scalability, Extensibility, Manageability. J2EE EJB EJBObject EJBHome EJBObjectJ2EE API

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJB</td>
<td>Enterprise JavaBeans is a server component model that provides portability across application servers and implements automatic services on behalf of the application components.</td>
</tr>
<tr>
<td>JNDI</td>
<td>Java Naming and Directory Interface provides access to naming and directory services such as DNS, NDS, LDAP, and CORBA Naming.</td>
</tr>
<tr>
<td>RMI/IIOP</td>
<td>Remote Method Invocation creates remote interfaces for Java-to-Java communications. This extension uses the CORBA standard IIOP communications protocol.</td>
</tr>
<tr>
<td>Java IDL</td>
<td>Java Interface Definition Language creates remote interfaces to support Java-to-CORBA communications. Java IDL includes an IDL-to-Java compiler and a lightweight ORB that supports IIOP. (Java IDL is part of the Java 2 platform, Standard Edition, formerly known as the Java Development Kit.)</td>
</tr>
<tr>
<td>Servlets and JSP</td>
<td>Java servlets and Java Server Pages are server components that run in a Web server that supports dynamic HTML generation and session management for browser clients.</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service supports asynchronous communications using either a reliable queuing or publish-and-subscribe programming model.</td>
</tr>
<tr>
<td>JTA</td>
<td>Java Transaction API provides a transaction demarcation API.</td>
</tr>
<tr>
<td>JTS</td>
<td>Java Transaction Service defines a distributed transaction management service based on CORBA Object Transaction Service.</td>
</tr>
<tr>
<td>JDBC™</td>
<td>JDBC Database Access API provides uniform access to relational databases such as DB2, Informix, Oracle, SQL Server, and Sybase. (JDBC is part of the Java 2 platform, Standard Edition.)</td>
</tr>
<tr>
<td>JavaMail</td>
<td>JavaMail provides a protocol independent framework to build mail and messaging applications. (JavaMail requires the JAF API.)</td>
</tr>
<tr>
<td>JAF</td>
<td>JavaBeans Activation Framework provides standard services to determine the type of an arbitrary piece of data and activate an appropriate JavaBeans component to manipulate the data.</td>
</tr>
</tbody>
</table>

J2EE EJB EJBObject EJBHome EJBObjectJ2EE API
Functional and Non-functional

Capabilities and Design Goals

Fault-tolerance, usability, portability, performance...

Artifacts


Communication mechanisms

HTTP, HTTPS, RMI, IIOP, Messaging, Transaction, (Abstraction), services, functional
Experience can be an architect's best tool.
J 2EE Best Practices and Guidelines
// Filename: HelloApplet.java ---  
import java.applet.Applet;  
import java.awt.Graphics;  
public class HelloApplet extends Applet  
{  
  public void paint(Graphics g)  
  {  
    g.drawString("Hello Grady!", 10, 20);  
  }  
}  

// Filename: HelloApplication.java ---  
public class HelloApplication  
{  
  public static void main(String args[])  
  {  
    System.out.println("Hello Grady!");  
  }  
}
Custom Actions

Custom actions are portable and reusable...

Custom actions are ideal for iterating through data and generating the HTML code needed to render a page.
Datacenter Server

UserSession, Exception, Workflow, Query(DB), Expressions <%=...%>, Taglib Directive, Implicit Object, Stop!... , JNDI, JDNI, URL

WebTier

UI

Message, Debugging, Factory, UserSession, Exception, Workflow, Query(DB), Encoding/Decoding, Security/ACL, Statistic, Multimedia, Locale

Default Logging

OK!
Persistence, Deployment time configuration

```
// Filename: HelloHome.java
import javax.ejb.*;
import java.rmi.RemoteException;
public interface HelloHome extends EJBHome
{
    public HelloObj create () throws RemoteException;
}

// Filename: HelloObj.java
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
public interface HelloObj extends EJBObject
{
    public String getMessage() throws RemoteException;
}
```
public class InganPK implements java.io.Serializable

// Filename: InganPK.java
import java.io.Serializable;
public class InganPK implements java.io.Serializable {

    public void ejbCreate() throws RemoteException, CreateException {
        ...
    }

    // getMessage()
    // ^
    ...
}

// Filename: Hello.java
import javax.ejb.*;
public class Hello implements SessionBean {

    public void ejbCreate() throws RemoteException, CreateException {
        ...
    }

    // SessionBean
    // ...
// EJB 140]  *  DAO(Data access objects), 143]  *  VO(Value objects)  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
// EIS Tier  *  JEE]  *  JMS, JDBC, JNDI, JMS, JavaMail, JavalDL, JNI],  *  ...  *  JEE]  *  "EJB"
EJB (Enterprise JavaBeans)

- RPC (Remote Procedure Call)
- CORBA, COM, EJB
- Stub, Skeleton, Proxy
- ORPC
- MTS (Microsoft Transaction Server)
- RMI (Remote Method Invocation)

Stub, Skeleton, Proxy.
GRADY REPORT I : EJB
Issued by: <...>
Revision: <0.2>
<2001/01/21>

JSTORM
**EJB Framework**

EJB Framework is a distributed programming platform based on the concept of the component model. EJBs (Enterprise JavaBeans) are Java components that provide functionalities such as transaction management, security, and persistence. EJBs are deployed in a container, which manages the lifecycle of the EJBs and provides services like lookup, transaction management, and security.

- **EJB server**: The EJB server is the component that runs the EJB container. It manages the lifecycle of the EJBs and provides services like lookup, transaction management, and security.
- **EJB container**: The EJB container is responsible for managing the lifecycle of the EJBs. It provides services like lookup, transaction management, and security.
- **Home interface**: The Home interface is the interface that the client uses to lookup the EJBs. It provides methods to create, remove, and lookup EJBs.
- **Remote interface**: The Remote interface is the interface that the client uses to interact with the EJBs. It provides methods to invoke business methods of the EJBs.
- **EJB Object**: The EJB Object is the implementation of the EJB. It contains the logic of the EJB.
- **Bean instance**: The Bean instance is the object that is created from the EJB Object. It is the instance of the EJB that the client uses to interact with.

**Session Beans**

Session beans are a type of EJB that is designed to be stateless. Session beans are used for implementing business logic that is not related to a specific entity. Session beans are typically used for implementing business logic that is not related to a specific entity, such as a user interface or a service that is used by multiple users. Session beans can be stateless or stateful. Stateless session beans do not maintain state across invocations of the business methods, while stateful session beans maintain state across invocations of the business methods.

- **Session Bean**: A session bean is a type of EJB that is designed to be stateless. Session beans are used for implementing business logic that is not related to a specific entity. Session beans are typically used for implementing business logic that is not related to a specific entity, such as a user interface or a service that is used by multiple users. Session beans can be stateless or stateful. Stateless session beans do not maintain state across invocations of the business methods, while stateful session beans maintain state across invocations of the business methods.
```java
// Filename: HelloHome.java
import javax.ejb.*;
import java.rmi.RemoteException;
public interface HelloHome extends EJBHome
{
    public HelloObj create () throws RemoteException;
}
```

```java
public interface javax.ejb.EJBHome extends java.rmi.Remote
{
    public abstract EJBMetaData getEJBMetaData() throws java.rmi.RemoteException;
    public abstract HomeHandle getHomeHandle() throws java.rmi.RemoteException;
    public abstract void remove(Handle handle)
throws javax.ejb.RemoveException, java.rmi.RemoteException;
    public abstract void remove(Object primaryKey)
throws javax.ejb.RemoveException, java.rmi.RemoteException;
}
```
public interface javax.ejb.EJBMetaData
{
    public abstract EJBHome getEJBHome();
    public abstract Class getHomeInterfaceClass();
    public abstract Class getRemoteInterfaceClass();
    public abstract boolean isSession();
    public abstract Class getPrimaryKeyClass();
}

public interface javax.ejb.EJBObject extends java.rmi.Remote
{
    public abstract EJBHome getEJBHome() throws java.rmi.RemoteException;
    public abstract void remove() throws java.rmi.RemoteException,
    javax.ejb.RemoveException;
    public abstract Handle getHandle() throws java.rmi.RemoteException;
    public abstract boolean isIdentical(EJBObject eo) throws
    java.rmi.RemoteException;

    // Filename: HelloObj.java
    import javax.ejb.EJBObject;
    import java.rmi.RemoteException;
    public interface HelloObj extends EJBObject
    {
        public String getMessage() throws RemoteException;
    }

    // EJB 1.1 - 82
public abstract Object getPrimaryKey() throws java.rmi.RemoteException;
}

isIdentical(EJBOBJECT eo)  Entity Bean  Bean[]  EJBObject[]  EJBHome[]
EJBOBJECT[]  Remote[]  Remote[]  Remote[]  Remote[]

Home[]  Home[]  Home[]  Home[]  HomeHandle[]  JNDI[]  EJBOBJECT[]  EJBOBJECT[]
Home[]  Handle[]  EJBOBJECT[]  EJBOBJECT[]  EJBOBJECT[]  EJBOBJECT[]  EJBOBJECT[]  EJBOBJECT[]
Home[]  Remote[]  Remote[]  Remote[]  Remote[]  Remote[]  Remote[]  Remote[]

// Filename: Hello.java
import javax.ejb.*;
public class Hello implements SessionBean
{
    public void ejbCreate() throws RemoteException, CreateException {...}
// getMessage()
    // ... ^
...
}

Bean[]  SessionBean[]  EJBSessionBean[]  EJBSessionBean[]  EJBSessionBean[]

public interface SessionBean extends EnterpriseBean
{
    public abstract void setSessionContext(SessionContext ctx)
        throws java.rmi.RemoteException;
    public abstract void ejbPassivate() throws java.rmi.RemoteException;
    public abstract void ejbActivate() throws java.rmi.RemoteException;
    public abstract void ejbRemove() throws java.rmi.RemoteException;
}

setSessionContext (SessionContext ctx)  EJB  EJBCustomContext[]  EJBHome[]  EJBHome[]
transaction[]  security[]  EntityContext[]  EJBContext[]  EJBContext[]  EJBContext[]
public interface EJBContext
{
    public EJBHome getEJBHome();
    public principal getCallerPrincipal();
    public boolean isCallerInRole(String role);
    public UserTransaction getUserTransaction();
    public boolean getRollbackOnly();
    public void setRollbackOnly();
}
**Entity Beans**

- Entity Beans
- Persistence
- Object serialization
- Byte stream
- JDBC

- EJB 1.0
- 1.1
- Java
- R-DB
- OODB
- OR
public interface EntityBean extends EnterpriseBean
{
    public abstract void ejbActivate() throw RemoteException;
    public abstract void ejbPassivate() throw RemoteException;
    public abstract void ejbLoad() throw RemoteException;
    public abstract void ejbStore() throw RemoteException;
    public abstract void ejbRemove() throw RemoteException;
    public abstract void setEntityContext(EntityContext ctx)
        throw RemoteException;
    public abstract void unsetEntityContext() throw RemoteException;
}

public interface EntityContext extends EJBContext
{
    public EJBObject getEJBObject() throw IllegalStateException;
    public Object getPrimaryKey() throw IllegalStateException;
}

EntityBean[] unsetEntityContext? SessionBean[] setEntityContext? EntityContext[]
getPrimaryKey? CMP[]
BMP(Bean-managed persistence)?

public interface EntityContext extends EJBContext
{
    public EJBObject getEJBObject() throw IllegalStateException;
    public Object getPrimaryKey() throw IllegalStateException;
}

EntityBean[] ejbLoad? ejbStore? CMP[]
BMP(Bean-managed Persistence)?
public interface InganHome extends javax.ejb.EJBHome {
    
    ...
    
    public Ingan findByPrimaryKey(InganPK pk) throws FinderException, RemoteException;
    
    public Enumeration findByInganseong(int level) throws FinderException, RemoteException;
}
public interface InganHome extends javax.ejb.EJBHome // Home/CMP/BMP
{
    public InganPK ejbFindByPrimaryKey(InganPK pk) throws FinderException, RemoteException
    {
        ...
        ps = con.prepareStatement("select name from Ingan where name = ?");
        ps.setString(1, pk.name);
        ...
    }
    public Enumeration ejbFindByInganseong(int level) throws FinderException, RemoteException
    {
        ...
    }
}

public class InganEJB implements EntityBean // Only 4 EJB 1.1
{
    public String name;
    public int level;
    private EntityContext ctx;
    ...
    public InganPK ejbCreate(String name, int level) throws CreateException // CMP/BMP
    {
        if(name == null)
        {
        }
throw new CreateException("You. Idiot! The name is required.");
}
this.name = name;
this.level = level;
return null;
}
A(atomic) · C(consistent) · D(durable) · Isolated] · J( subsidiary) · L(isolated) · D( durable) · .

Isolation level] · serializable] · read uncommitted] · read committed]

Locking level] · read committed] · .

begin, commit, rollback] · DB] · .


Deployment descriptor] · .

Unsupported, Required, RequiresNew, Mandatory, Never] · 6] · .

Exception, Never] · .
<ejb-jar>

...<transaction-type> Container </transaction-type>

...<Container-transaction>

<trans-attribute> Required </trans-attribute>

...</ejb-jar>

 STATUS_ACTIVE    STATUS_COMMITTED,
STATUS_COMMITTING,  STATUS_MARKED_ROLLBACK,
STATUS_NO_TRANSACTION, STATUS_PREPARED,
STATUS_PREPARING,  STATUS_ROLLEDBACK,
STATUS_ROLLING_BACK,  STATUS_UNKNOWN

javax.transaction.Status
EJB Security

Security Authentication(Identity), Authority, Integrity, Privacy, Auditability Authentication Authority Integrity Privacy

EJB Security User Entity Principal Role User EJB Entity

EJB Security entity Role

*.properties Active Directory
GRADY REPORT 1 : EJB
Issued by: < >
Revision: <0.2> <2001/01/21>

... weblogic.password.grady=morning
...
...

... ejb-jar.xml role method... getCallerPrincipal isCallerInRole... principa role OK... EJB Security...
Design Patterns in Application

Creational Patterns, Structural Patterns, Behavioral Patterns

...
Basic Level: Class, Object, Event, Message, Inheritance, Delegation, Interface


Library, Design Pattern, Component, Framework, Interface, Class, Object, Event, Message, Inheritance, Delegation, Interface
Factory Pattern

Abstract Factory Pattern

Factory Method Pattern
Facade Pattern

Facade Pattern

WebTier EJB Tier

Facade

Dispatcher

Servlet, JSP

Facade

Dispatcher

Request Parsing

User, Session

index.html,
default.html

*.jsp

EJB
Session Bean

MS
MTS (Microsoft Transaction Server)

Stateless, Stateful

Stateless, Stateful
Singleton Pattern

Singleton Pattern

JDK System, Runtime

Private

Static

System.out.println(...) runtime.getRuntime().n

Static

Factory

VM

System

Runtime

Runtime rt = Runtime.getRuntime().n

Singleton

JSTORM
Reflection Pattern

Reflection Pattern in Java. Reflection is a Java language feature that allows a program to inspect and modify the structure and behavior of its own classes at runtime. It is used extensively in Java, especially in late-breaking environments where code needs to dynamically interact with the runtime environment. Reflection is particularly useful in contexts such as web applications, where the structure of the data and the way it is processed can change at runtime.

MVC Pattern

MVC (Model-View-Controller) is a software design pattern that is used to develop user interfaces. It decomposes the user interface into three distinct functional components: the model, the view, and the controller. The model represents the data and business logic of the application, the view is responsible for displaying the data to the user, and the controller mediates user input and updates the model.

Smalltalk is a programming language that was developed by Alan Kay in 1972. It is one of the first object-oriented programming languages, and it played a significant role in the development of modern software design patterns such as MVC. Smalltalk is known for its powerful visual development tools and its support for dynamic typing and garbage collection.
Observer Pattern

```
Subject
Attach(observer)
Detatch(observer)
Notify()

ConcreteSubject
GetState()
GetState()
subjectState

Observer
update() for all o in observers
Update()

ConcreteObserver
update() subject.getSubjectState()
```

```
RequestId:
Subject
Attach() Detatch() Observer
Notify()

ConcreteSubject
GetState()
GetState()
subjectState

Observer
update() for all o in observers
Update()

ConcreteObserver
update() subject.getSubjectState()
```
Mediator Pattern

```mermaid
graph TD
    Mediator[Mediator] --> mediator[mediator]
    Colleague[Colleague] --> mediator[mediator]
    ConcreteMediator[ConcreteMediator] --> ConcreteColleague1[ConcreteColleague1]
    ConcreteColleague1[ConcreteColleague1] --> mediator[mediator]
    ConcreteColleague1[ConcreteColleague1] --> ConcreteColleague2[ConcreteColleague2]
```

- Facade
- Observer
- Mediator
- Subject

Mediator Pattern

```mermaid
graph TD
    aListBox[owner] --> aViewManager[Owner]
    aTextPane[Owner] --> aViewManager[Owner]
    aButton[Owner] --> aViewManager[Owner]
    aListBox[Owner] --> aViewManager[Owner]
    aTextPane[Owner] --> aViewManager[Owner]
    aButton[Owner] --> aViewManager[Owner]
```

- Mediator
- Facade
- Observer
- Subject

Observer pattern

- Mediator Pattern
- Design pattern
Facade, Factory, MVC
"Driving is about constantly paying attention, making a little correction this way, a little correction that way." This is the paradigm for XP. ... Everything in software changes, the requirements change. The design changes. The business changes. The technology changes. The team changes ...
Objectory (Programming)

ETRI (ETRI)

RUP (Rational Unified Process), SLC (System Life Cycle), XP (Extreme Programming)

Business Illnesses (Martin)

Medicine (Meyer, Odell, Lorenz, Graham, Booch...)

Concept & Guideline, Rational}

Objectory

SDLC, PTC (Parametric Tech. Corp.)

IT

OT, CBD

Guru

Guide
C# | DNS | ActiveDirectory | Kerberos |
--- | --- | --- | --- |
Public Key | ASP+ | DTC (Distributed Transaction Coordinator), ADO+, NLB | Built-In COM+ | CLB (Component Load Balancing) |
Grady Report | Issued by: Project 94
| hpark@kebi.com |
XP (eXtreme Programming)

XPExtreme Programming is a software development approach that emphasizes collaboration, flexibility, and continuous improvement. XP was developed by Ken Schwaber and Ron Jeffries in 1998, and it is based on the ideas of eXtreme Programming (XP) and other agile methodologies.

XP is characterized by a set of core values, practices, and techniques that are designed to help development teams deliver high-quality software rapidly and flexibly. Some of the key practices of XP include:

- Pair programming: Two programmers work together at a single workstation to develop code. This helps ensure that the code is of high quality.
- Refactoring: Periodically, the team will refactor the code to improve its structure and make it easier to maintain.
- Continuous integration: Code is checked into a shared repository frequently, and the team runs automated tests to catch bugs early.
- Maintenance: XP is focused on maintaining the software throughout its lifecycle, not just during development.

XP is often contrasted with traditional software development approaches, which emphasize predictability and control. XP advocates for a more flexible and responsive approach, where the team can adapt to changing requirements and respond to new information.

XP is heavily influenced by the ideas of Kent Beck and the other members of the Extreme Programming Community. The XP community is known for its focus on collaboration, transparency, and continuous improvement.

XP has been applied in a wide range of organizations and industries, and it has become a popular approach for software development teams around the world. XP is often used in conjunction with other agile methodologies, such as Scrum or Kanban.

For more information on XP, you can visit the official XP website:

http://www.extremeprogramming.org/
Communication, Simplicity, Feedback, Courage

If short iterations are good, we'll make the iterations really, really short—seconds and minutes and hours, not weeks and months and years (the Planning Game).

Pair programming? CRC card? Test programming? Refactoring? Integrating and releasing code into the code repository every few hours! (Anybody can change any line of code to add functionality, fix bugs, or refactor.) Move people around? Developers will go to the customer and receive a detailed description of the requirements face to face!...
Beck and others... Inappropriate Intimacy, Alternative Classes with Different Interfaces, Incomplete Library Class, Data Class, Refused Bequest, Comments... Improving the design of existing code, without changing its observable behavior... Duplicate code, Long Method, Large class, Long Parameter List, Divergent Change, Shotgun Surgery, Feature Envy, Data Clumps, Primitive Obsession, Switch Statements, Parallel Inheritance Hierarchies, Lazy Class, Speculative Generality, Temporary Field, Message Chains, Middle Man, Inappropriate Intimacy, Alternative Classes with Different Interfaces, Incomplete Library Class, Data Class, Refused Bequest, Comments... "Refactoring"... "eXtreme Programming eXplained", "Planning eXtreme Programming", "eXtreme Programming Installed"... "Cost of Change"...
Contemporary Cost of Change

Time or Lifecycle Phase

- Pilot
- Design
- Maintenance

Diagram showing cost of change over time with phases labeled as Pilot, Design, and Maintenance.
GRADY REPORT 1 : EJB

Revision: <0.2>  <2001/01/21>

JSTORM  <55/56>
JSTORM