The State Transition Engine
Development of PL/SQL Applications
with a Business Rules Approach

Robert F. Edwards & Dr. Paul Dorsey
Dulcian, Inc.
Agenda

- Problem & Solution
- Business Rules Architecture
- STE Concepts
- STE Development
- Demo – Timesheet App
The Problem

- Tax agency
  - Hundreds of documents
  - Each document has a different process
  - Processes were highly changeable (major changes each year)
- No way to do this in a traditional environment
How we solved the problem

- **New Idea** – Articulate business process flows and let users write the code.
- **Natural way to think about business events**
- **State Transition Engine (STE)**
- **Using this approach** -
  - Users write the code.
  - STE provides better code management.
  - Generator creates better code.
- **The STE supports application development.**
Advantages of Business Rule Environment

- Users participate in design.
- UML model (80% of structural rules)
  - Still hard to read (20% participation)
  - Users can’t build them (except to add, modify attributes)
- Process Flows (95% of process rules)
  - 95% participation
  - Users can build them!!!
Why use a state transition engine?

- It is a way of looking at an object (business event).
- It replaces 90% of application logic.
- Things that the STE replaces:
  - Default values
  - Object access
  - Field-level edit privileges
  - Process flow steps
  - Program logic
  - Item ordering in applications
- With an STE, applications become object viewers.
- It is virtually a complete programming language.
State Transition Language

- Process Flow = State Transition Language
- Business Process Flow Diagram = analysis
- Communicate business events to users
- Flow diagrams are graphical
- In STE, process flows are the source code.
- Process flows loaded into STE repository
Architecture

- Business Rules Repository

Structural
- Data Model

Process
- Business Event
- Process Flows

Business Rules

Data

Applications
- Domain Builder
- Object Builder
- User-friendly applications
Use State Transition Engine (STE) idea to support application development.

Place all process-related business rules (code) in STE repository.

Generate code (PL/SQL).

Run entire system with ONE application.
Process Rules Architecture

[Diagram showing the process rules architecture with labels for Traditional Apps, Generic Apps, Process Engine, UML Class Models, Translator, Repository, PL/SQL Code, Code Generator, Data.]
“The only reason you are able to build so cheaply is that you foist the programming off onto your users.”

Ulka Rodgers
Generates simple PL/SQL procedures
User maintainable
ProcessBusinessEvent(event_oid, result_id)
All generated code, no overhead
STE Concepts

- **State** – An activity at a point in a process flow
- **State Events** – Predefined trigger points
- **Transitions** – A business event changes state
- **Task** – A line of code
States

◆ State:
  ➢ A point in time in a process flow where an activity may occur

◆ Manual
  ➢ When an object is in a manual state, it stays there until some event moves it to a different state

◆ Automatic
  ➢ When an object is in an automatic state, it executes some behavior (code) and automatically transitions to another state
State Types

- **Manual States**
  - Begin
  - End
  - Inbox
  - Wizard
  - Suspend
  - Error

- **Automatic States**
  - Automatic
  - Auto Begin
Listed in the order in which they may occur:

- On_Set
- Expiration
- Before_Open
- Manual_Processing
- Auto_Bail
- Bail
- Manual_Decision
- Automatic_Decision
- Listener
Transitions

◆ Manual
  - Manual Decision

◆ Automatic
  - All other events

◆ Rules for transitions
  - Automatic – like a case statement
  - Manual – validation rule
- Line of executable code
- Types used in STE
  - **Assignments** –
    - Salary := 5000
    - Party_Name:=First_Name||’ ‘||Last_Name
  - **Function calls** –
    - Create_JE_YN := Create_Journal_Entry_YN(OID)
    - Obj_ID:=Create_Bus_Event_ID(‘Add_Employee’)
Tasks (2)

◆ Attach to events
  - Before_Open
  - On_Set

◆ Attach to transitions
  - Auto transitions
    - Expire
    - Listener
    - Bail
    - Auto_Decision
  - Manual Decision
    - Rule_Success, Rule_Fail
STE Development

- **Traditional**
  - Requirements, process flow, code C/C++

- **STE**
  - Requirements, process flow (the code)

- A new paradigm in development

- Easier to design, code, test, debug and maintain
Define Flow on Business Event

Business events impact core data structures

- Party
PO Approval process

Diagram:
- Create PO
- Process - 19
- Approve1
- Deny - 20
- Denied
- Approve2
- Approve - 10
- Approved
Implementation

- Generate procedures
- Procedures call each other
- All variables reside in PL/SQL table
procedure p_auto_565(SelfOID in Number) is
Begin

ste.doc(8890).ValueDT:=glste.f_mature_dt(SelfOID,ste.doc(8868).ValueTX);
/*MatureDate :=
glste.f_mature_dt(SelfOID,ErrorMessage) */
if (ste.doc(8890).ValueDT>stepl.f_sysdate) then
/*(MatureDate > stepl.f_sysdate)*/
    ste.SetEventState(SelfOID,573);
elsif 1=1 then
   /*No Rule*/
    ste.SetEventState(SelfOID,570);
else
    raise uml.e_ste_rule_failure;
end if;
End;
Advantages of using the STE

- All code specifications are stored in the repository.
- Actual code is generated.
  - We can change the generation algorithm at will.
  - Improved performance – standardized structure
  - Supports multi-tasking
  - Enforces record locking for entire business event
- Excellent performance – PL/SQL tables
- No logic in the application
  - We can write specific applications, if desired
<table>
<thead>
<tr>
<th>Structural</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Model (UML)</td>
<td>State Transition Engine</td>
</tr>
<tr>
<td>Domains</td>
<td>Functions</td>
</tr>
<tr>
<td>Triggers</td>
<td></td>
</tr>
</tbody>
</table>

- Almost no business logic outside of the repository
- 90% of entire system is generated
- Analysis = Production
Timesheet Application
  - Process Flow Development
<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robert F. Edwards</td>
<td><a href="mailto:redwards@dulcian.com">redwards@dulcian.com</a></td>
</tr>
<tr>
<td>Dr. Paul Dorsey</td>
<td><a href="mailto:paul_dorsey@dulcian.com">paul_dorsey@dulcian.com</a></td>
</tr>
</tbody>
</table>

www.dulcian.com