A Peer-to-peer Architecture for Workflow in Virtual Enterprises

John Noll
jnoll@scu.edu

Computer Engineering Department, Santa Clara University
Workflow

The automation of a business process ... during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.
Example

requirements

Design

Implement

design
code

test
design
test
suite

Design Tests

Implement Tests

Test
PEOS Process Enactment Demonstration

spell_check_document

State: BLOCKED

Required: document = $$

Resources: document = $$

Provided: document = $$

Resources: document = $$

Script: "Use your editor's spell check feature to verify the correct spelling of your document."

Start  Finish  Suspend  Abort

©2004 Santa Clara University. Please direct your inquiries: jnoll@scu.edu
Benefits

- *Automation* of repetitive, routine tasks.
Benefits

*Automation* of repetitive, routine tasks.

*Guidance* to users as to which steps to perform when, according to the process description;
Benefits

- *Automation* of repetitive, routine tasks.
- *Guidance* to users as to which steps to perform when, according to the process description;
- *Monitoring* of the progress of process performance, to provide process visibility for management and historical data for analysis and planning;
Benefits

- Automation of repetitive, routine tasks.
- Guidance to users as to which steps to perform when, according to the process description;
- Monitoring of the progress of process performance, to provide process visibility for management and historical data for analysis and planning;
- Control to ensure that processes are performed correctly.
Conventional Approach

Client-server architecture with central workflow engine and database.
Conventional Approach

Client-server architecture with central workflow engine and database.

- Global visibility
Conventional Approach

Client-server architecture with central workflow engine and database.

- Global visibility
- Tight control
Conventional Approach

Client-server architecture with central workflow engine and database.

- Global visibility
- Tight control
- Close coordination
Problem

Conventional client-server architecture has limitations:
Problem

Conventional client-server architecture has limitations:

- Single point of failure
Problem

Conventional client-server architecture has limitations:

- Single point of failure
- Requires central administration & infrastructure
Problem

Conventional client-server architecture has limitations:

- Single point of failure
- Requires central administration & infrastructure
- Organization centric
Solution

Peer-to-peer architecture. Addresses:
Solution

Peer-to-peer architecture. Addresses:

1. Autonomy - each participating individual and group retains control over who has access to data and activities.
Solution

Peer-to-peer architecture. Addresses:

1. Autonomy - each participating individual and group retains control over who has access to data and activities.

2. Heterogeneity - participating individuals and groups work in preferred/existing environments, including tools, platforms, and data management.
Solution

Peer-to-peer architecture. Addresses:

1. Autonomy - each participating individual and group retains control over who has access to data and activities.

2. Heterogeneity - participating individuals and groups work in preferred/existing environments, including tools, platforms, and data management.

3. Distribution and Mobility - groups and individuals can be widely distributed, with varying degrees of connectivity.
Architecture

actor

user interface

task list

process events

event dispatcher

task list

process events

resource events

recompute()

virtual machine

resource state

create/modify/delete object

state query

process & resource events

(to/from other nodes)
Coordination

Peers are grouped into *domains* that share events among group to coordinate activities:
Coordination

Peers are grouped into *domains* that share events among group to coordinate activities:

1. Replication domains - share *process* events: coordination through task sequencing.
Coordination

Peers are grouped into *domains* that share events among group to coordinate activities:

1. Replication domains - share *process* events: coordination through task sequencing.

2. Coordination domains - share *resource* events: coordination through shared resources.
Coordination

Peers are grouped into *domains* that share events among group to coordinate activities:

1. Replication domains - share *process* events: coordination through task sequencing.
2. Coordination domains - share *resource* events: coordination through shared resources.
3. Events distributed within a domain via flooding.
Example

1: checkin(code)
2: done(Implementation)
3: recompute()
4a: notify(FINISHED, Implementation)
5a: recompute()
6a: 'What are my tasks?'
7a: 'Test task is Ready'
8a: start(Test)
9a: recompute()
10a: notify(STARTED, Test)
4b: notify(CREATED, code)
5b: recompute
6b: 'What are my tasks?'
7b: 'Test task is Ready'
8b: start(Test)
9b: recompute

programmer

tester

volunteer
Issues

1. How do peers find and join domains?
Issues

1. How do peers find and join domains?
2. Is there a better method than flooding?
Issues

1. How do peers find and join domains?
2. Is there a better method than flooding?
3. Where are process models stored for replication domains?
Conclusions

1. Loose coupling: Actors can choose to work closely, or they may share events anonymously.
Conclusions

1. Loose coupling: Actors can choose to work closely, or they may share events anonymously.

2. Architecture defines the interaction among nodes, but not the implementation of a node.
Conclusions

1. Loose coupling: Actors can choose to work closely, or they may share events anonymously.
2. Architecture defines the interaction among nodes, but not the implementation of a node.
3. Tasks are allocated bottom-up: “I don’t have to do anything I don’t want to.”