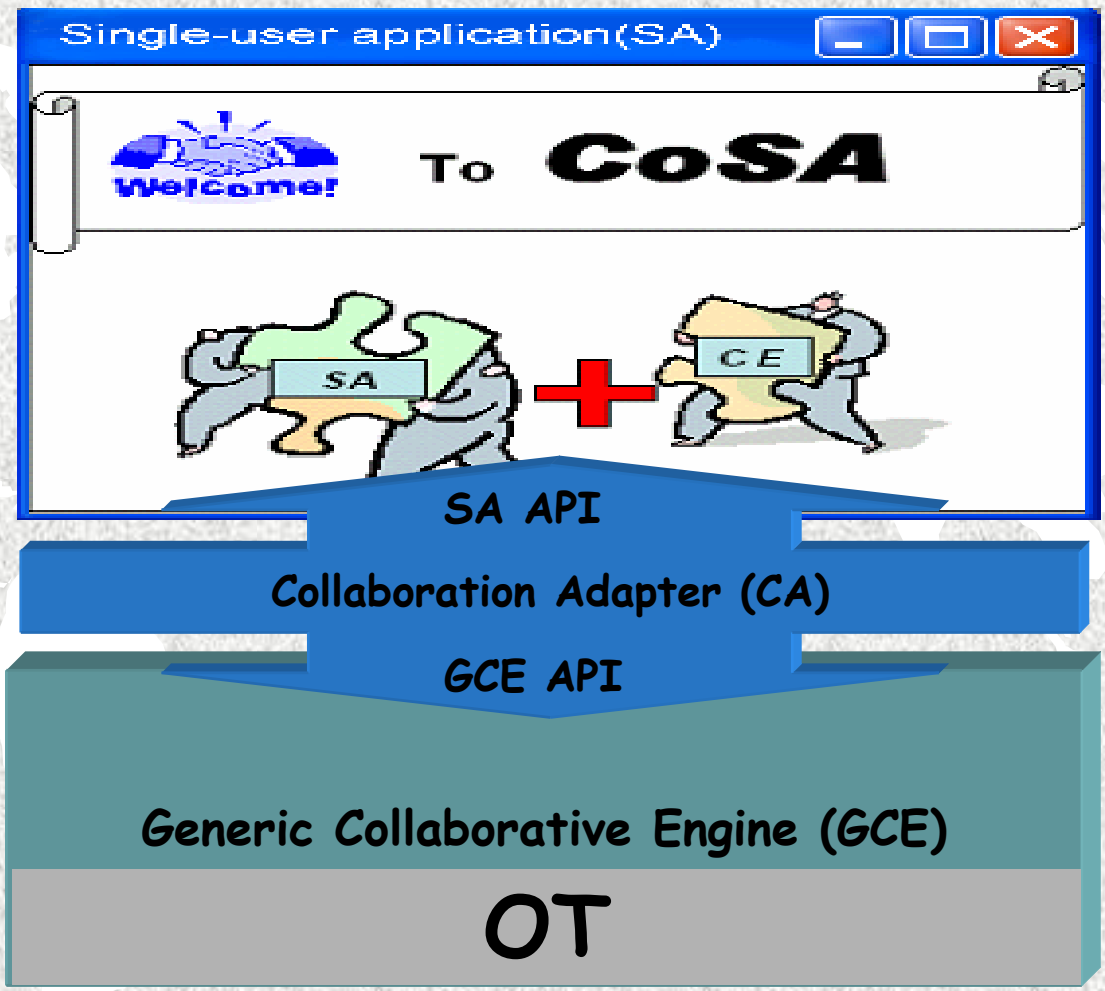
Using time-space diagrams for testing real OT systems

David Sun
Department of Electrical Engineering and Computer Science
University of California at Berkeley, USA
DavidSun@berkeley.edu

Chengzheng Sun, Steven Xia, Jian Guo
School of Computer Engineering
Nanyang Technological University, Singapore

The Transparent Adaptation (TA) technology

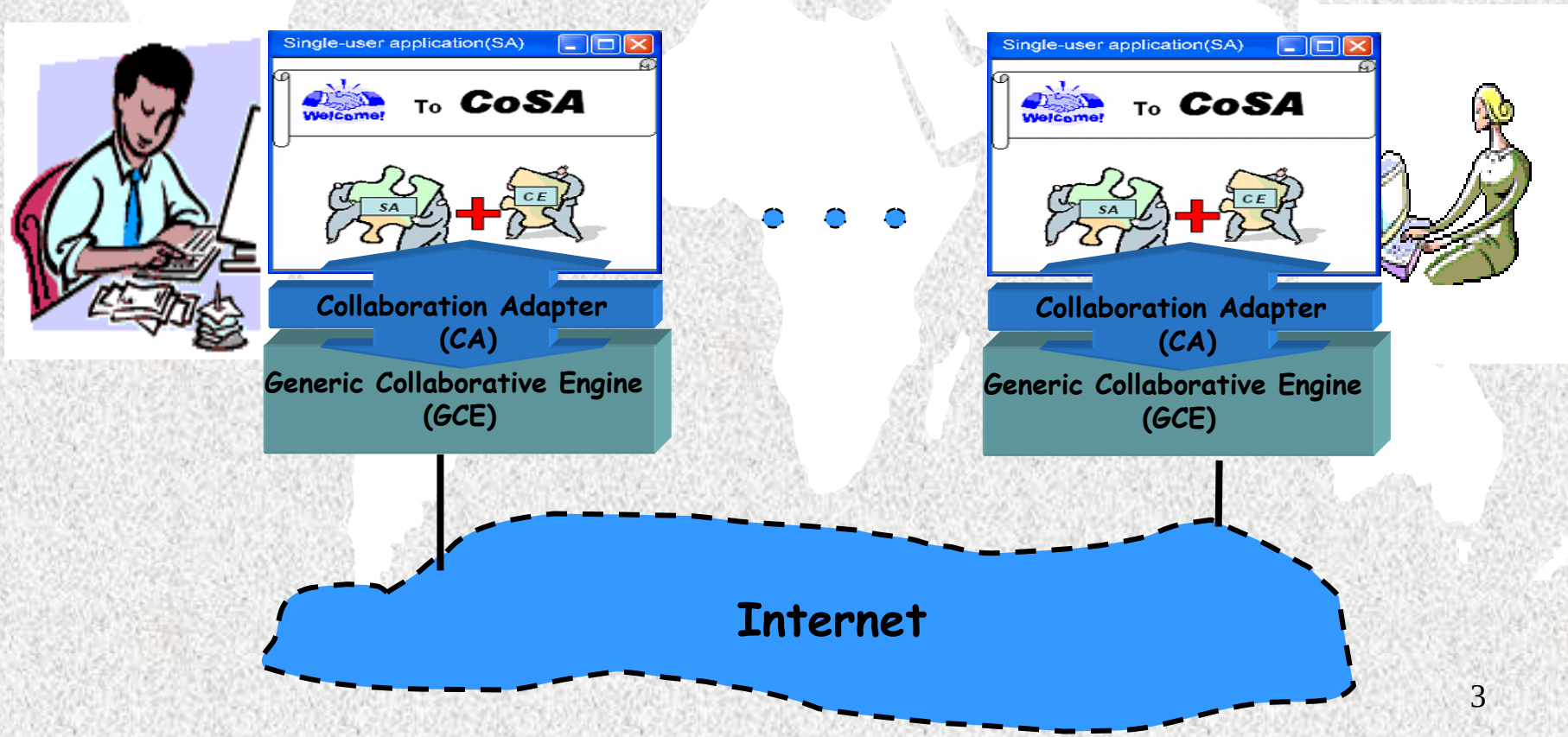


OT is the cornerstone of the GCE.

Comprehensive testing is crucial in validating a real OT system

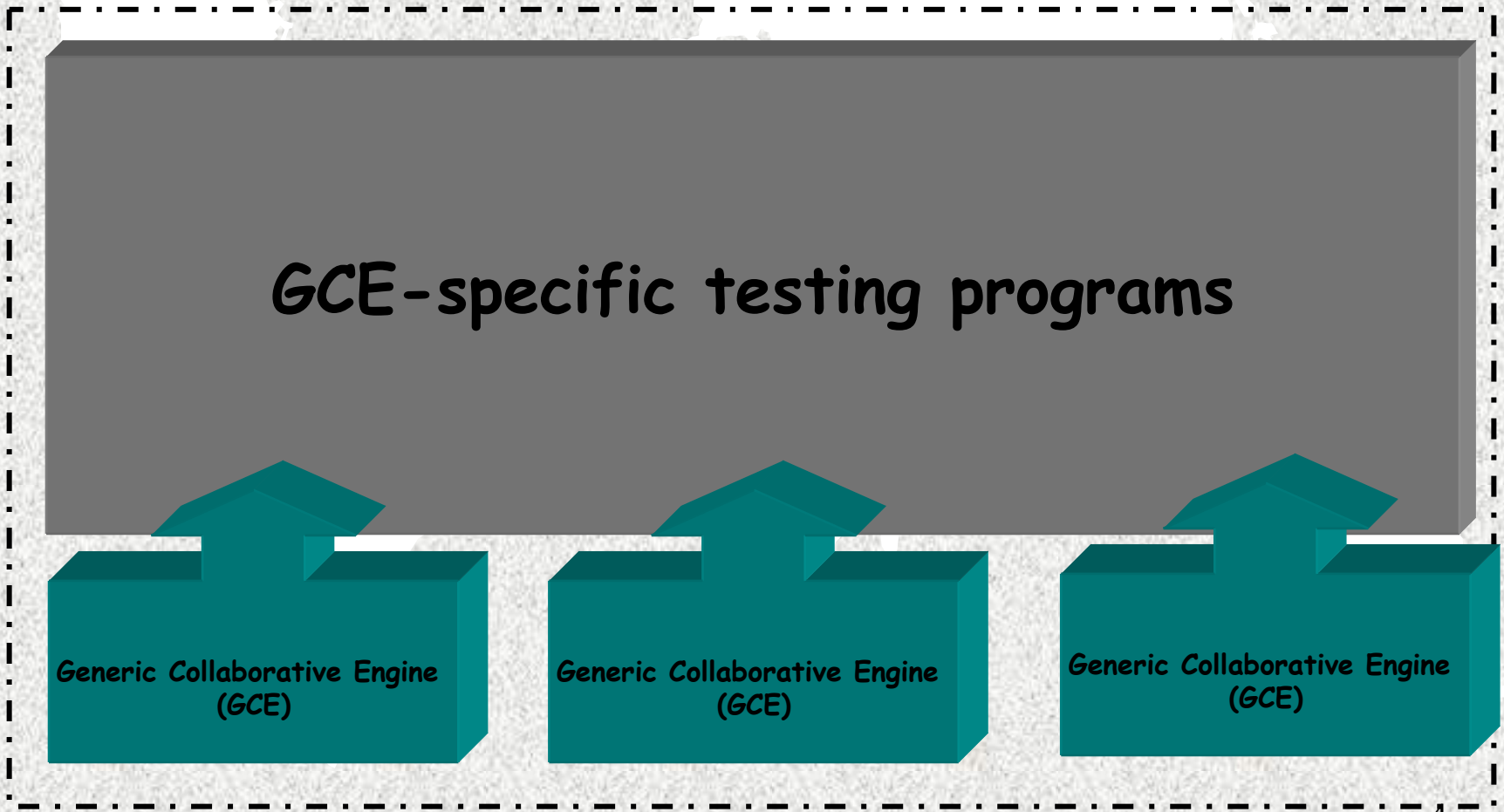
Testing is difficult due to:

- Involvement of multiple users and computers
- Coordination of operation generation (concurrency and causality)
- Control of operation propagation (communication latency)
- Mixture of SA,CA and GCE (OT) issues (difficult to differentiate)



Independent testing of the OT-based GCE

- Separate GCE issues from the rest of the system
- Easy to generate operations with arbitrary relationships

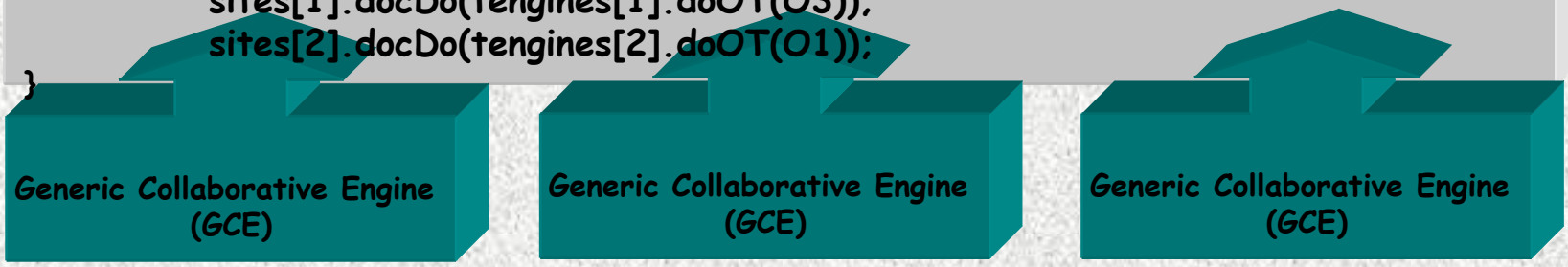


An example GCE testing program

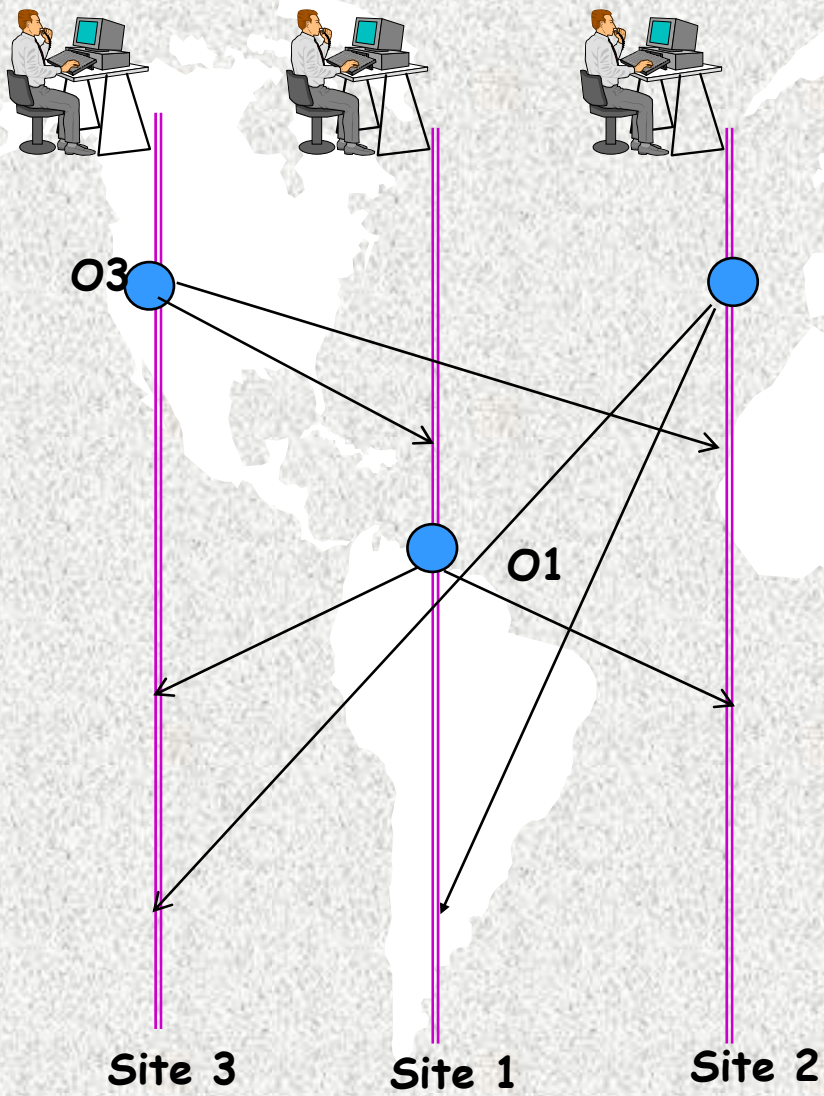
```
Void main( ) {// a GCE test program
                // initialize document state and engines
          char* initDoc = "abc";
          tengines = createEngines(3);
          createDocStates(sites, initDoc, 3);

            // create and do local operations
          OP O1 = createOp(OP_INSERT, 1, 0, "1", "1");
          OP O2 = createOp(OP_INSERT, 2, 1, "2", "2");
          OP O3 = createOp(OP_INSERT, 3, 2, "3", "3");
          sites[0].docDO(O1, islocal);  tengines[0].localOP(O1);
          sites[1].docDO(O2, islocal);  tengines[1].localOP(O2);
          sites[2].docDO(O3, islocal);  tengines[2].localOP(O3);

          // do remote operations
          sites[0].docDo(tengines[0].doOT(O3));
          sites[1].docDo(tengines[1].doOT(O1));
          sites[2].docDo(tengines[2].doOT(O2));
          sites[0].docDo(tengines[0].doOT(O2));
          sites[1].docDo(tengines[1].doOT(O3));
          sites[2].docDo(tengines[2].doOT(O1));
}
```

Drawbacks:

- hand-coded,
- GCE-specific,
- non-intuitive.



Time-Space Diagrams (TSD)

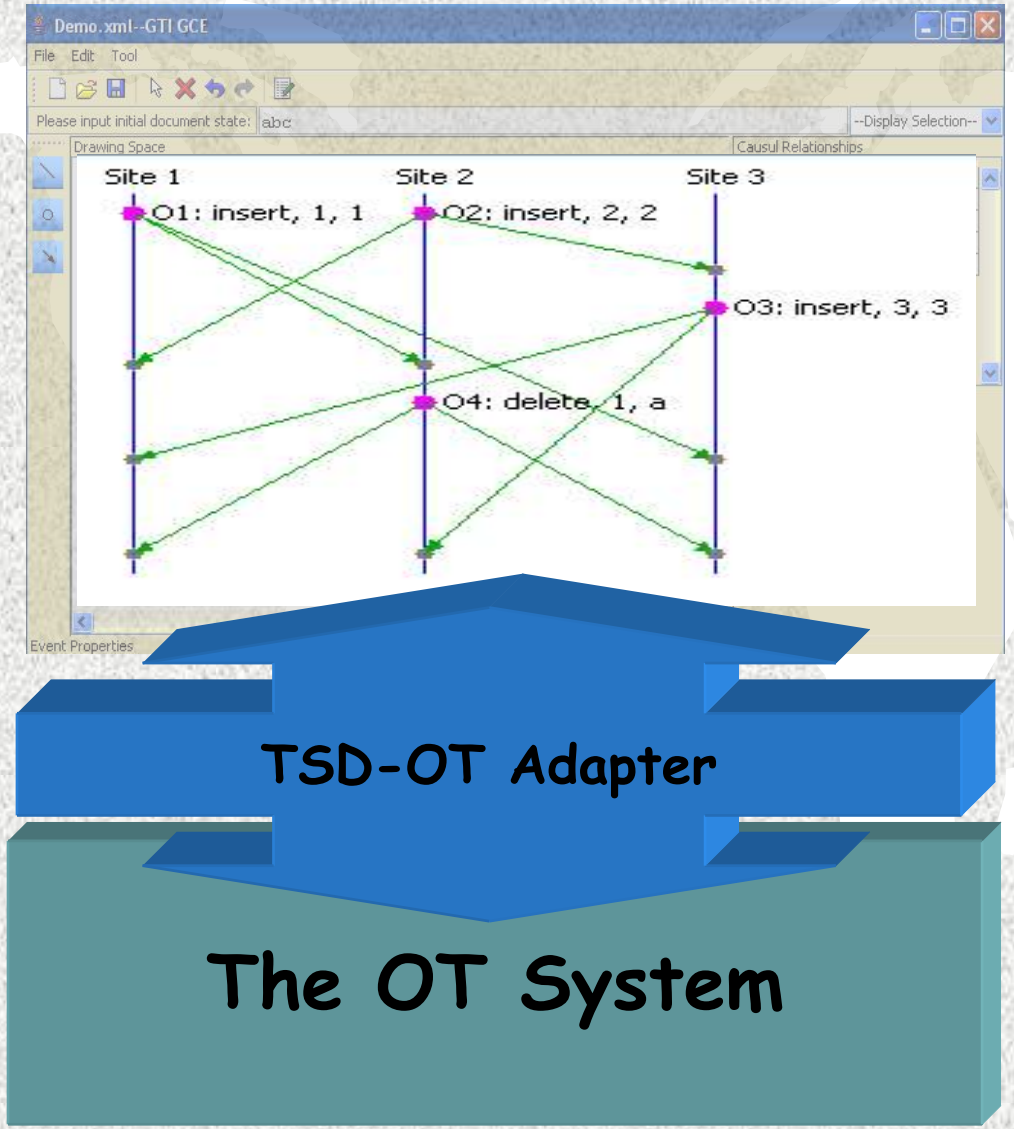


1. TSD is an effective tool for describing operations with arbitrary dependency and independency relationships .

2. TSD is commonly used to describe representative scenarios for illustrating the correctness or potential problems of an OT system.

3. All known OT-puzzle scenarios have been expressed in TSD, which can be utilized as a comprehensive testing suite for OT systems.

A TSD-OT testing system architecture



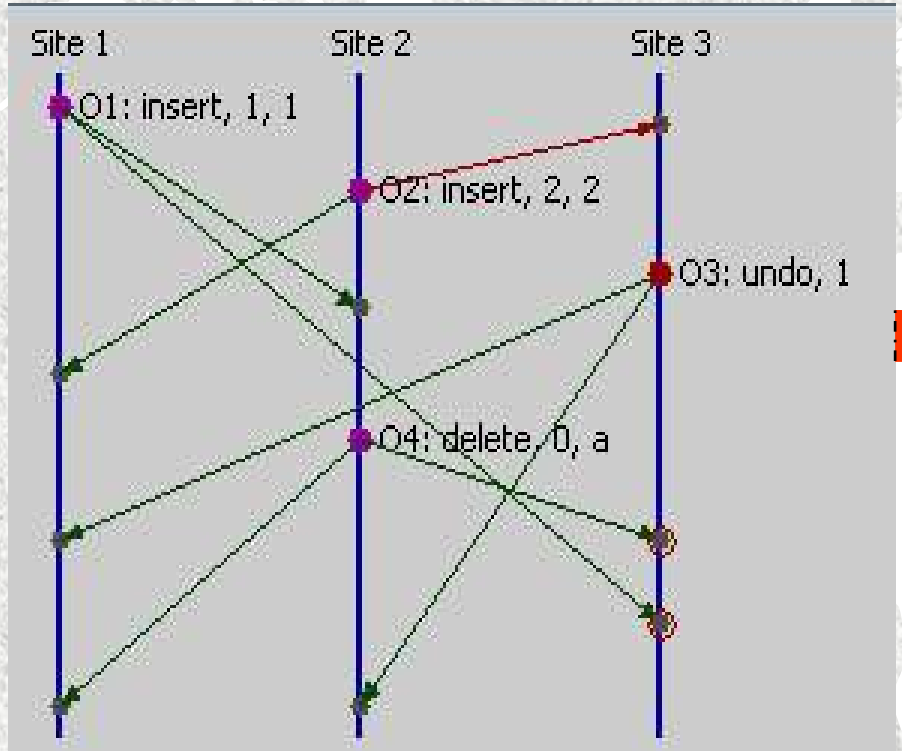
1. TSD-GUI provides a graphic user interface for describing OT testing scenarios and for displaying OT system outputs.

2. TSD-OT adaptor provides conversion between the generic TSD description and the OT-system-specific testing code and outputs.

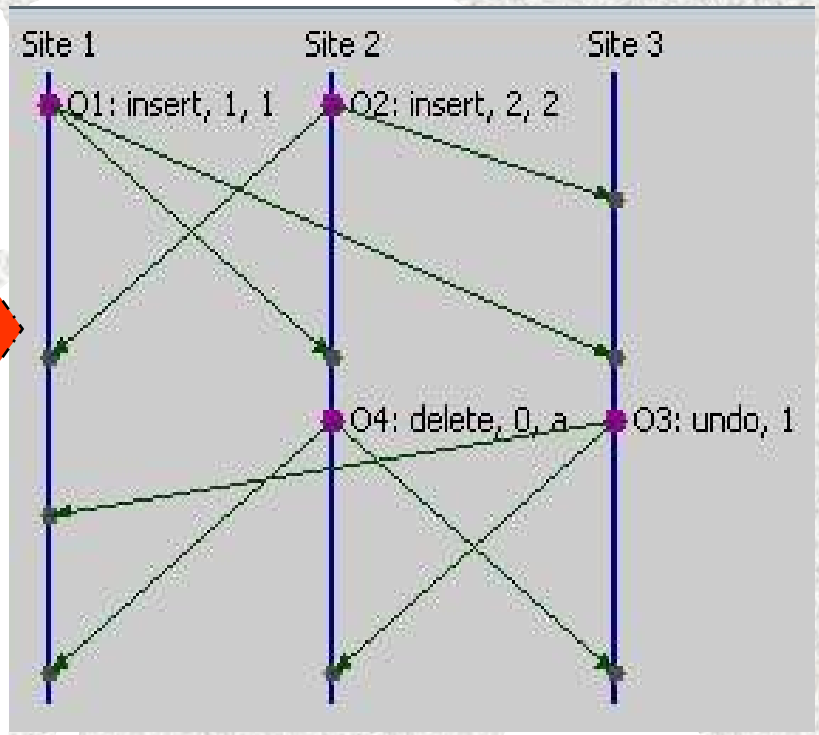
3. OT-based system (e.g. GCE) implements the OT algorithms to be tested.

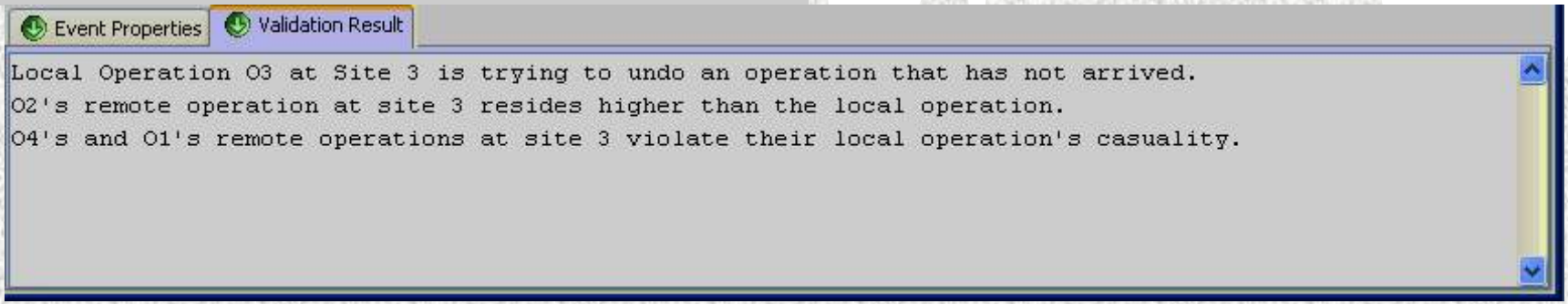
Design issues
(1) Validating TSD

Invalidate TSD

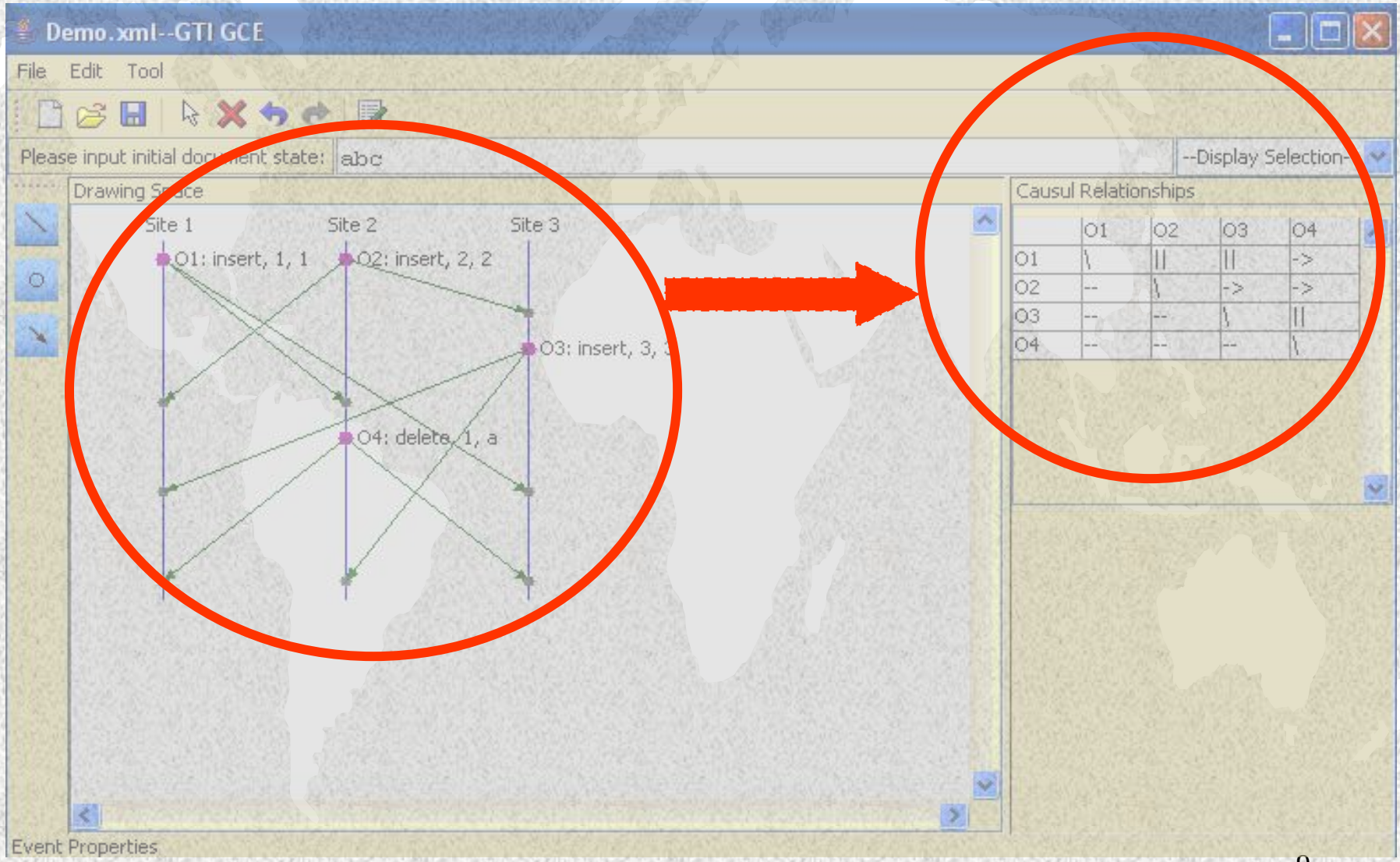


Validate TSD

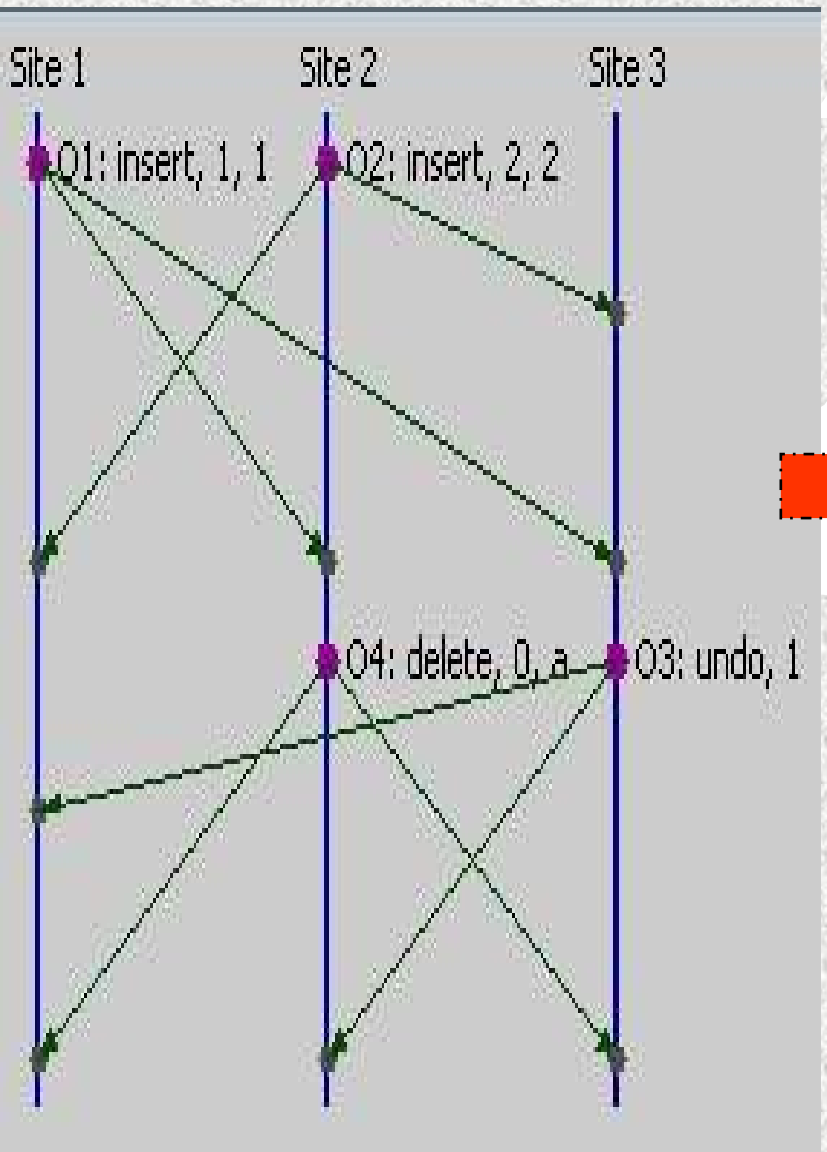




(2) Deriving causal relationships from TSD



(3) Converting TSD into TSD-script




```
<doc>
    abc
</doc>
<site>
</site>
```

```
<site>
    <siteid>3</siteid>
    <event>
        <eventid>2</eventid>
        <local>false</local>
        <statevector>{0,1,0}</statevector>
    </event>
    <event>
        <eventid>1</eventid>
        <local>false</local>
        <statevector>{1,1,0}</statevector>
    </event>
    <event>
        <eventid>3</eventid>
        <local>true</local>
        <op>undo</op>
        <undoid>1</undoid>
        <statevector>{1,1,1}</statevector>
    </event>
    <event>
        <eventid>4</eventid>
        <local>false</local>
        <statevector>{1,2,1}</statevector>
    </event>
</site>
```

```
</event>
</site>
</site>
```

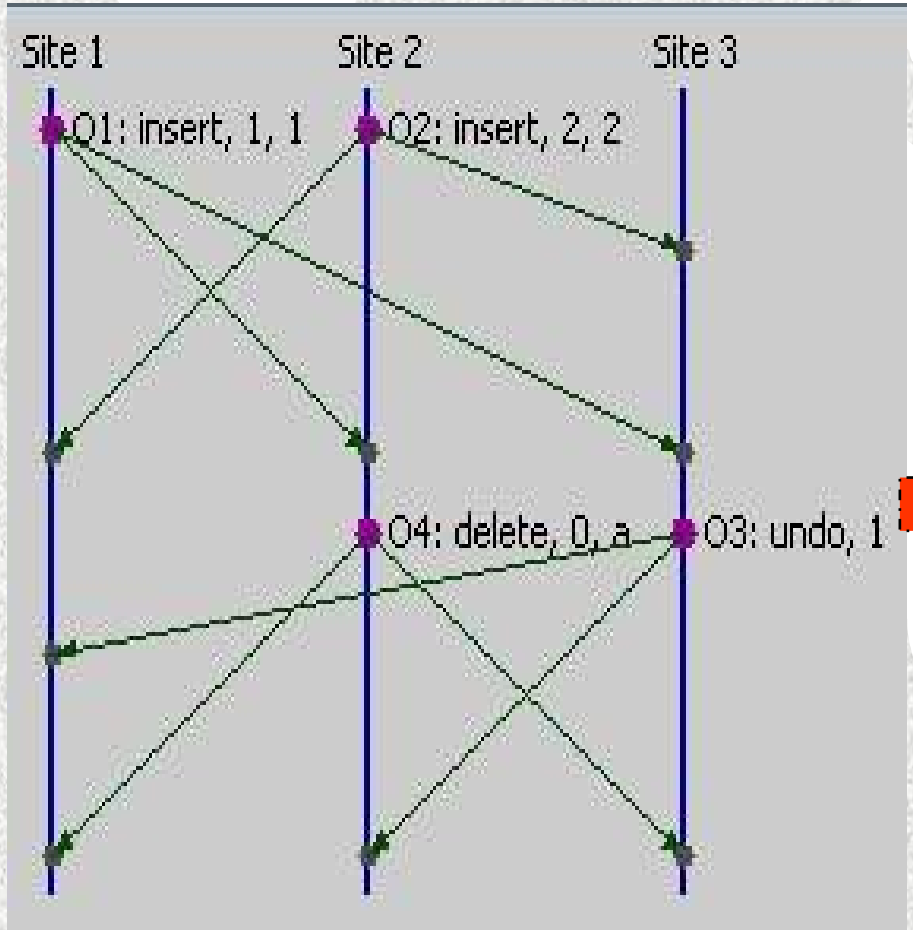
(4) Converting TSD-script into GCE-testing code

```
<site>
<site>
<siteid>2</siteid>
<doc>
    abc
</doc>
<site>
        <siteid>1</siteid>
        <event>
                <eventid>1</eventid>
                <local>true</local>
                <op>insert</op>
                <pos>1</pos>
                <char>1</char>

        <statevector>{1,0,0}</statevector>
        </event>
        <event>
                <eventid>2</eventid>
                <local>false</local>

        <statevector>{1,1,0}</statevector>
        </event>
        <event>
                <eventid>3</eventid>
                <local>false</local>

        <statevector>{1,1,1}</statevector>
        </event>
        <event>
                <eventid>4</eventid>
                <local>false</local>

        <statevector>{1,2,1}</statevector>
        </event>
</site>
```

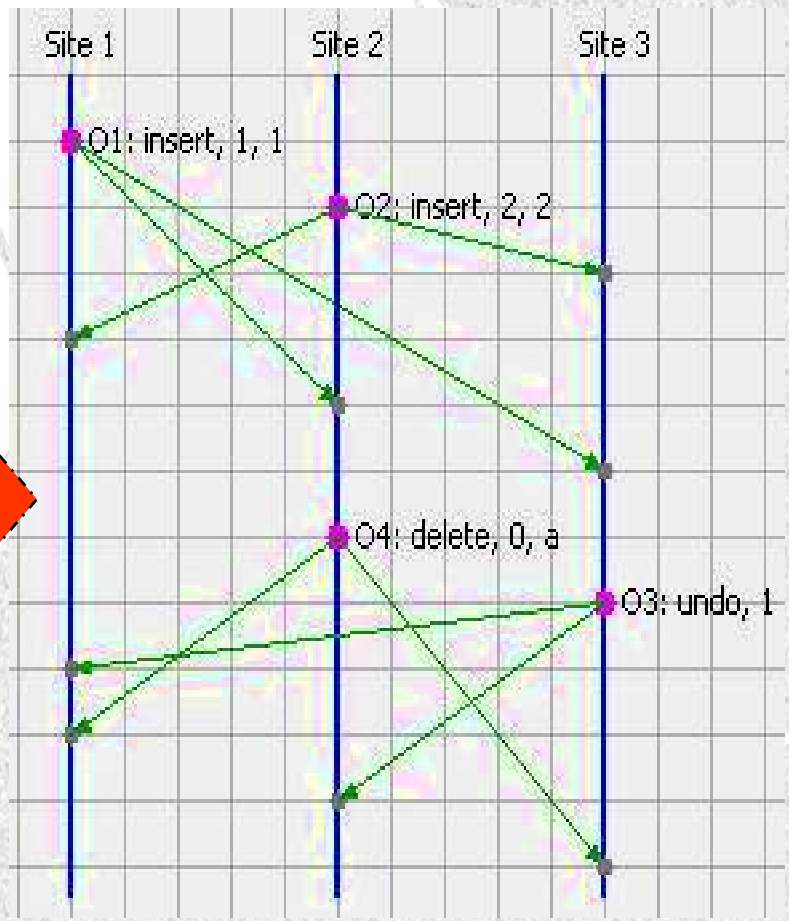
```
void Example() {
    vector<DocState> sites;
    char* initDoc = "abc";
    vector<TE> tengines = createEngines(3);
    createDocStates(sites, initDoc, 3);

    OP O1 = createOp(OP_INSERT, 1, 0, "1", "1");
    OP O2 = createOp(OP_INSERT, 2, 1, "2", "2");
    OP O4 = createOp(OP_DELETE, 0, 1, "a", "4");

    //local do
    sites[0].docDO(O1, islocal);
    tengines[0].localOP(O1);
    sites[1].docDO(O2, islocal);
    tengines[1].localOP(O2);

    //remote do
    sites[2].docDO(tengines[2].doOT(O2));
    sites[0].docDO(tengines[0].doOT(O2));
    sites[1].docDO(tengines[1].doOT(O1));
    sites[2].docDO(tengines[2].doOT(O1));

    //local do
    sites[1].docDO(O4, islocal);
    tengines[1].localOP(O4);

    //local undo
    OP O1U = O1;
    tengines[2].makeUndoCmd(O1U);
    sites[2].docDO(tengines[2].undoOT(O1U));

    //remote undo
    O1U = O1;
    tengines[0].makeUndoCmd(O1U);
    sites[0].docDO(tengines[0].undoOT(O1U));

    //remote do
    sites[0].docDO(tengines[0].doOT(O4));

    //remote undo
    O1U = O1;
    tengines[1].makeUndoCmd(O1U);
    sites[1].docDO(tengines[1].undoOT(O1U));

    //remote do
    sites[2].docDO(tengines[2].doOT(O4));
}
```

TSD-serialization underlying the GCE code generation

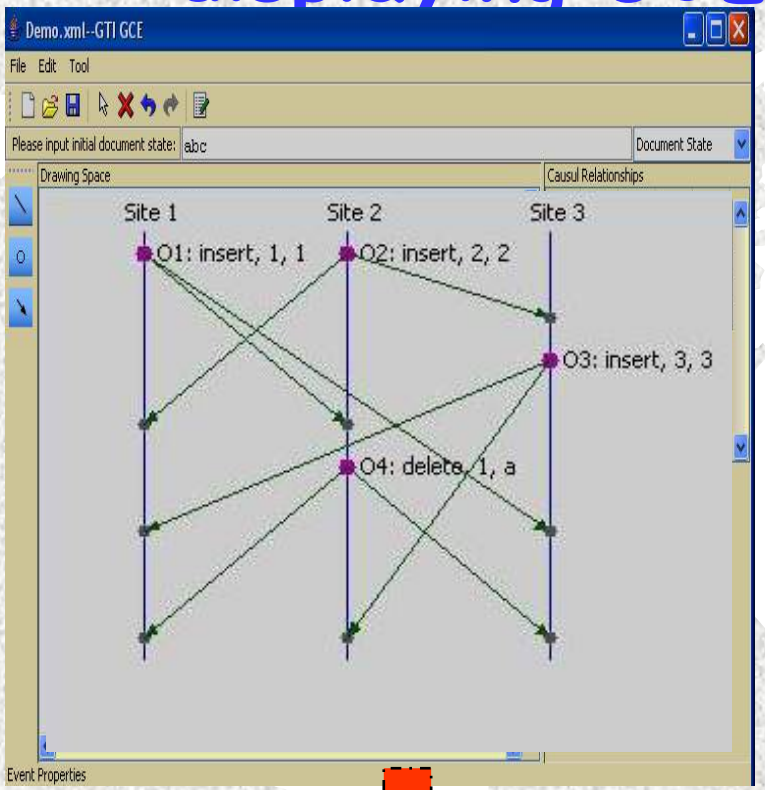
Normal TSD

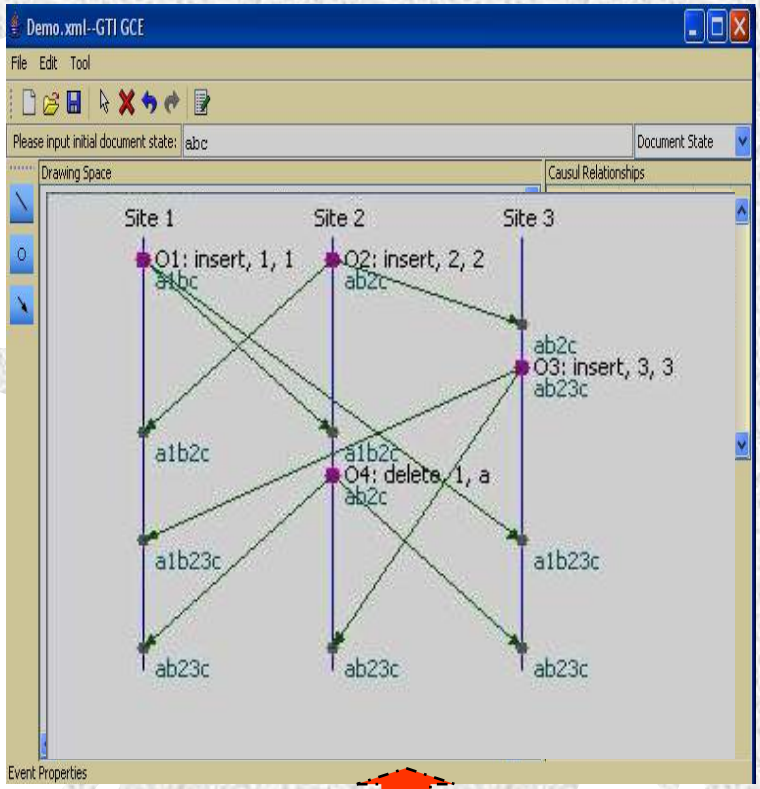


Serialized TSD



(5) Executing TSD-generated code and displaying GCE-generated results





TSD-generated Testing Code

TSD-GCE Adaptor

GCE-generated Testing Results

Generic Collaborative Engine (GCE)

Generic Collaborative Engine (GCE)

Generic Collaborative Engine (GCE)

Concluding remarks

1. The TSD-based OT testing tool has helped u
   to test the GCE used in CoWord and other
   follow-up TA-based applications.

2. The TSD-based testing suite are independent
   of the OT system being tested, and can be
   used as general validation benchmark for OT
   systems.

3. We continue to improve and enhance the
   functionality the TSD-based OT testing tool,
   and to produce a polished API, which can be
   used to adapt this tool to other OT systems.