APPENDIX A

LSD Account for the Warehouse

```
AGENT foreman(w) {
    STATE
    ORACLE
        rf_on_transport[4],                  /* Information in RF4, filled by Forklift Operator */
        rf_redistribution_confirm[2]
    HANDLE
        rf_warehouse[1...5],
        tf_foreman_name[1...5],
        rf_job_number[1...5],
        rf_item[1...5],
        rf_quantity[1...5],
        rf_from_place[1...5],
        rf_to_warehouse[1...5],
        rf_date[1...5]
    DERIVATE
    PROTOCOL
        *** Deciding the distribution ***
            --> write rf_warehouse[1...5]; write tf_foreman_name[1...5];
                write rf_job_number[1...5]; write rf_item[1...5];
                write rf_quantity[1...5]; write rf_from_place[1...5];
                write rf_to_warehouse[1...5]; write rf_date[1...5];
            --> *** Keep RF5 and pass RF1...4 to warehouse worker; ***
        (rf_on_transport[4] == 'Yes') || (rf_redistribution_confirm[2] == 'Done')
            --> *** Update quantity ***      /* After finishing loading/unloading, update quantity */
        (rf_redistribution_confirm[2] != 'Done')
            --> *** Pass RF2 back to office *** /* Redistribution error! */
}

AGENT warehouseWorker(w) {
    STATE
    ORACLE
        rf_item[1...4],
        rf_quantity[1...4],
        rf_from_place[1...4],
        rf_date[1...4],
        rf_item[2],
        rf_quantity[2],
        rf_date[2]
    HANDLE
        rf_moving_pending[1...4],
        rf_loading_time[1...4],
        rf_loading_platform[1...4],
        rf_redistribution_confirm[1...4],
        rf_to_place[2],
        rf_redistribution_confirm[2]
    DERIVATE
    PROTOCOL
        *** Receiving RF1...4 from foreman ***
```

```
            --> read rf_item[1...4]; read rf_quantity[1...4];
                read rf_from_place[1...4]; read rf_date[1...4];
                write rf_moving_pending[1...4];
            --> write rf_loading_time[1...4];
                write rf_loading_platform[1...4];
            --> *** Pass RF1,2,3 to office and RF4 to forklift operator; ***
                                          /* For loading */
        *** Item not enough ***
            --> write rf_redistribution_confirm[1...4] == ('Error' && 'Item not enough');
            --> *** Pass RF1...4 back to foreman; ***
        *** Receiving RF2 from office ***
            --> read rf_item[2]; read rf_quantity[2]; read rf_date[2];
            --> write rf_to_place[2];
            --> *** Pass RF2 to forklift operator; *** /* For unloading */
        *** All places full ***
            --> write rf_redistribution_confirm[2] == ('Error' && 'All places full');
            --> *** Pass RF2 back to foreman; ***
}

AGENT forkliftOperator(w) {
    STATE
    ORACLE
        rf_item[4],
        rf_quantity[4],
        rf_from_place[4],
        rf_date[4],
        rf_loading_time[4],
        rf_loading_platform[4],
        rf_item[2],
        rf_quantity[2],
        rf_date[2],
        rf_arrival_time[2],
        rf_unloading_platform[2]
    HANDLE
        rf_on_transport[4],
        rf_redistribution_confirm[4],
        rf_redistribution_confirm[2]
    DERIVATE
    PROTOCOL
        *** Receiving RF4 from warehouse worker ***
            --> read rf_item[4]; read rf_quantity[4];
                read rf_from_place[4]; read rf_date[4];
                read rf_loading_time[4]; read rf_loading_platform[4];
            --> *** Have the items ready when/where the truck is expected; ***
            --> write rf_on_transport[4] == 'Yes';/* After loading */
            --> *** Pass RF4 to foreman; *** /* Finsahing loading */
        *** Truck didn't come ***
            --> write rf_redistribution_confirm[4] == ('Error' && 'Unable to load items');
            --> *** Pass RF4 back to foreman; ***
        *** Receiving RF2 from warehouse worker ***
            --> read rf_item[2]; read rf_quantity[2];
                read rf_date[2]; read rf_arrival_time[2];
                read rf_unloading_platform[2];
            --> *** Move the items to new place; ***
            --> write rf_redistribution_confirm[2] == 'Done';
                                          /* After unloading */
            --> *** Pass RF2 to foreman; *** /* Finishing unloading */
```

```
        *** Truck didn't arrive ***
            --> write rf_redistribution_confirm[2] == ('Error' && 'Unable to unload items');
            --> *** Pass RF2 back to foreman; ***
}

AGENT office {
    STATE
    ORACLE
        rf_warehouse[1,2,3],
        rf_job_number[1,2,3],
        rf_to_warehouse[1,2,3],
        rf_date[1,2,3],
        rf_redistribution_confirm[2],
        tf_driver,
        tf_on_going,
        tf_from_warehouse,
        tf_loading_date,
        tf_to_warehouse,
        ttp_driver[1],
        ttp_date[1],
        ttp_from_warehouse[1],
        ttp_to_warehouse[1]
    HANDLE
        rf_driver[1,2,3],
        rf_arrival_time[1,2,3],
        rf_unloading_platform[1,2,3],
        rf_redistribution_confirm[1,2,3],
        ttp_on_going[1],
        tf_driver,
        tf_on_going,
        tf_from_warehouse,
        tf_loading_date,
        tf_to_warehouse
    DERIVATE
    PROTOCOL
        *** Receiving RF1,2,3 form warehouse worker ***
            --> read rf_warehouse[1,2,3]; read rf_job_number[1,2,3];
                read rf_to_warehouse[1,2,3]; read rf_date[1,2,3];
            --> *** According to TF decide which driver; ***
            --> write rf_driver[1,2,3]; write rf_arrival_time[1,2,3];
                write rf_unloading_platform[1,2,3];
            --> *** Keep RF1,2 and pass RF3 to the driver; ***
        *** Receiving TTP1 from driver ***
            --> read ttp_driver[1]; read ttp_date[1];
                read ttp_from_warehouse[1]; read ttp_to_warehouse[1];
                write ttp_on_going[1]; write tf_driver;
                write tf_on_going; write tf_from_warehouse;
                write tf_loading_date; write tf_to_warehouse;
            --> *** Pass RF2 to destination warehouse; ***
        *** If no truck available for that date ***
            --> write rf_redistribution_confirm[1,2,3] == ('Error' && 'No truck available');
            --> *** Pass RF1,2,3 back to foreman; ***
                                            /* Redistribution error! */
        *** (receive RF2 from destination foreman)
        rf_redistribution_confirm[2] != 'Done'
            --> *** Pass RF2 back to the original foreman; ***
                                            /* Redistribution error! */
```

```
}

AGENT driver(d) {
    STATE
    ORACLE
        rf_warehouse[3],
        rf_job_number[3],
        rf_to_warehouse[3],
        rf_date[3],
        rf_loading_time[3],
        rf_loading_platform[3],
        rf_driver[3],
        rf_arrival_time[3],
        rf_unloading_platform[3]
    HANDLE
        ttp_driver[1,2],
        ttp_date[1,2],
        ttp_job_number[1,2],
        ttp_from_warehouse[1,2],
        ttp_loading_platform[1,2],
        ttp_loading_time[1,2],
        ttp_to_warehouse[1,2],
        ttp_unloading_platform[1,2],
        ttp_arrival_time[1,2]
    DERIVATE
    PROTOCOL
        *** Receiving RF3 from office
            --> read rf_warehouse[3]; read rf_job_number[3];
                read rf_to_warehouse[3]; read rf_date[3];
                read rf_loading_time[3]; read rf_loading_platform[3];
                read rf_driver[3]; read rf_arrival_time[3];
                read rf_unloading_platform[3];
                write ttp_driver[1,2]; write ttp_date[1,2];
                write ttp_job_number[1,2]; write ttp_from_warehouse[1,2];
                write ttp_loading_platform[1,2];
                write ttp_loading_time[1,2]; write ttp_to_warehouse[1,2];
                write ttp_unloading_platform[1,2];
                write ttp_arrival_time[1,2];
            --> Keep TTP2 and pass TTP1 back to office; ***
        *** Time is 'up' --> Drive truck to the warehouse; ***
}

AGENT rf(n=1...5) {                          /* Redistribution Form */
    STATE
        rf_warehouse[n]=@,                   /* Information in RF1,2,3,4,5 */
        tf_foreman_name[n]=@,                /* Information in RF1,2,3,4,5 */
        rf_job_number[n]=@,                  /* Information in RF1,2,3,4,5 */
        rf_item[n]=@,                        /* Information in RF1,2,3,4,5 */
        rf_quantity[n]=@,                    /* Information in RF1,2,3,4,5 */
        rf_from_place[n]=@,                  /* Information in RF1,2,3,4,5 */
        rf_to_warehouse[n]=@,                /* Information in RF1,2,3,4,5 */
        rf_date[n]=@,                        /* Information in RF1,2,3,4,5 */
        rf_moving_pending[n]=@,              /* Information in RF1,2,3,4 */
        rf_loading_time[n]=@,                /* Information in RF1,2,3,4 */
        rf_loading_platform[n]=@,            /* Information in RF1,2,3,4 */
        rf_driver[n]=@,                      /* Information in RF1,2,3 */
        rf_arrival_time[n]=@,                /* Information in RF1,2,3 */
```

```
        rf_unloading_platform[n]=@,          /* Information in RF1,2,3 */
        rf_on_transport[n]=@,                /* Information in RF4 */
        rf_to_place[n]=@,                    /* Information in RF2 */
        rf_redistribution_confirm[n]=@       /* May appear in RF1,2,3,4 */
    ORACLE
    HANDLE
    DERIVATE
    PROTOCOL
}

AGENT ttp(t=1,2) {                           /* Truck Transportation Plan */
    STATE
        ttp_driver[t]=@,
        ttp_date[t]=@,
        ttp_job_number[t]=@,
        ttp_from_warehouse[t]=@,
        ttp_loading_platform[t]=@,
        ttp_loading_time[t]=@,
        ttp_to_warehouse[t]=@,
        ttp_unloading_platform[t]=@,
        ttp_arrival_time[t]=@,
        ttp_on_going[t]=@
    ORACLE
    HANDLE
    DERIVATE
    PROTOCOL
}

AGENT tf {                                    /* Transportation Form */
    STATE
        tf_driver=@,
        tf_on_going=@,
        tf_from_warehouse=@,
        tf_loading_date=@,
        tf_to_warehouse=@
    ORACLE
    HANDLE
    DERIVATE
    PROTOCOL
}
```

APPENDIX B

Details of the Agency in the Warehouse Process

Agent: Foreman

1. (Redistribution between warehouses) foremen decide the redistribution jobs for the specific week, with reference of Table G (items/places board), fill into Table A and then send to the office.
   - **Oracle**: place(G), item(G), quantity(G).
   - **Handle**: item(A), quantity(A), fromPlace(A), toWarehouse(A), finishByDate(A).
2. (Local redistribution) foremen decide the items to be moved within the warehouse, with reference of Table G (items/places board), fill into Table B and then send to the warehouse workers. [Note: in this stage the moving time and forklift in Table B is empty.]
   - **Oracle**: place(G), item(G), quantity(G).
   - **Handle**: item(B), quantity(B), fromPlace(B), toPlace(B), Date(B).

Agent: Warehouse Worker

1. (Redistribution between warehouses) after receiving Table C & D from office (via foreman), the warehouse workers decide the platform and forklift operator for each job, and which place to keep the items received, with reference of Table F (platform details), Table G (place details) and Table E (forklift operator daily schedule).
   - **Oracle**: item(C), quantity(C), loadingTime(C), item(D), quantity(D), unloadingTime(D), platform(F), time(F), forklift(F), time(E), place(G), item(G), quantity(G).
   - **Handle**: platform(C), platform(D), toPlace(D), forklift(B), forklift(C), forklift(D).
2. (Local redistribution) after receiving Table B from foreman, warehouse workers decide the forklift operator and moving time for each job with reference of Table E (forklift operator daily schedule).
   - **Oracle**: time(E).
   - **Handle**: movingTime(B).
3. Inform truck drivers the loading and unloading platforms.
4. Warehouse workers update Table E (forklift schedule) and Table F (platform details) according to Table B, C and D.
   - **Oracle**: platform(C), platform(D), forklift(F).

- **Dependency**: item(E) **is** item(B/C/D), quantity(E) **is** quantity(B/C/D), from(E) **is** fromPlace(B/C)/platform(D), to(E) **is** toPlace(B)/platform(C/D), time(E) **is** movingTime(B)/loading-Time(C)/unloadingTime(D), driver(E) **is** driver(C/D), item(F) **is** item(C/D), quantity(F) **is** quantity(C/D), fromToPlace(F) **is** fromPlace(C)/toPlace(D), time(F) **is** loadingTime(C)/unloadingTime(D), forklift(F) **is** forklift(B/C/D), driver(F) **is** driver(C/D).

5. Pass Table E to each forklift operator.
6. After receiving Table E from forklift operators, update Table G (place details).
   - **Oracle**: item(E), quantity(E), from(E), to(E), jobDone(E).
   - **Handle**: item(G), quantity(G).

Agent: Forklift Operator

1. Forklift operators do the redistribution jobs according to his daily schedule (Table E).
2. After finishing the redistribution jobs, forklift operators fill into jobDone in Table E and send it back to warehouse workers.
   - **Oracle**: jobKind(E), item(E), quantity(E), from(E), to(E), time(E).
   - **Handle**: jobDone(E).

Agent: Office

1. When received Table A from foremen, office will check Table H (truck schedule) and Table I (each driver daily schedule), then decide which driver and loading time for each job. Fill these information in Table I. [Note: in this stage the platform and forklift in Table C, the loading and unloading platforms in Table I are empty.]
   - **Oracle**: item(A), quantity(A), toWarehouse(A), finishByDate(A), driver(H), warehouses(H), time(H), item(I), quantity(I).
   - **Handle**: loadingTime(I).
   - **Dependency**: foreman(C, #) **is** foreman(A, #), item(C, #) **is** item(A, #), quantity(C, #) **is** quantity(A, #), fromPlace(C, #) **is** fromPlace(A, #), toWarehouse(C, #) **is** toWarehouse(A, #), fromWarehouse(I, #) **is** fromWarehouse(C, #), toWarehouse(I, #) **is** toWarehouse(C, #), item(I, #) **is** item(C, #), quantity(I, #) **is** quantity(C, #).
2. According to Table C, office produce Table D which shows all the redistribution jobs to the warehouse (for example Coventry) at the specified date.
   - **Oracle**: toWarehouses(C), date(C).
   - **Dependency**: item(D, #) **is** item(C, #), quantity(D, #) **is** quantity(C, #), fromWarehouse(D, #) **is** fromWarehouse(C, #), toWarehouse(I, #) **is** toWarehouse(C, #).
3. With reference of Table H and I, office decide the driver and unloading time for each job. [Note: at this stage the platform, toPlace and forklift in Table D are empty.]
   - **Oracle**: item(D), quantity(D), fromWarehouse(D), driver(H), warehouses(H), time(H), item(I), quantity(I).

- **Handle**: unloadingTime(I).
- **Dependency**: fromWarehouse(I, #) **is** fromWarehouse(D, #), toWarehouse(I, #) **is** toWarehouse(D, #), item(I, #) **is** item(D, #), quantity(I, #) **is** quantity(D, #).

4. Office fill/update Table C, D and H (according to Table I).
   - **Oracle**: jobNo(I).
   - **Dependency**: loadingtime(C, #) **is** loadingTime(I, #), unloadingtime(D, #) **is** unloadingTime(I, #), driver(C, #)/driver(D, #) **is** driver(I, #), warehouse(H, #) **is** fromWarehouse(I, #)/toWarehouse(I, #), time(H, #) **is** loadingTime(I, #)/unloadingTime(I, #).
5. After deciding the trucks, office will send Table C (without platform details) as well as Table D (unloading information) back to the warehouse. [Note: (1) the platforms in both Table C and Table D and toPlace in Table D are empty; (2) not all jobs in Table A will be allocated on 10 June 2000.]
6. Office send each truck driver Table I.

Agent: Truck Driver

Truck driver drives lorry to warehouses according to his daily schedule (Table I).

APPENDIX C

Paper-Based Tables/Forms Used During the Warehouse Process

Table A-1: One-Week Redistribution Jobs from Warehouse Coventry (sent by foreman)

Job #	Foreman	Items	Qty	From (P)	To Warehouse	Finished by
100001	*Bill*	*Screws*	*550*	*4*	*London*	*30 Nov 2001*
100002	*Susan*	*Bananas*	*1500*	*2*	*London*	*10 Nov 2001*
100003	*Bill*	*Oil Drums*	*200*	*1*	*Manchester*	*20 Nov 2001*
100004	*Susan*	*Computers*	*120*	*5*	*London*	*20 Nov 2001*
100005	*Susan*	*Bananas*	*3000*	*3*	*London*	*30 Nov 2001*

Table A-2: One-Week Redistribution Jobs from Warehouse London (sent by foreman)

Job #	Foreman	Items	Qty	From (P)	To Warehouse	Finished by
200001	*Nick*	*Bananas*	*3000*	*3*	*Coventry*	*10 Nov 2001*
200002	*Nick*	*Bananas*	*1500*	*2*	*Manchester*	*20 Nov 2001*
200003	*Buck*	*Oil Drums*	*700*	*5*	*Manchester*	*20 Nov 2001*
200004	*Buck*	*Screws*	*1000*	*4*	*Coventry*	*30 Nov 2001*

Table A-3: One-Week Redistribution Jobs from Warehouse Manchester

Job #	Foreman	Items	Qty	From (P)	To Warehouse	Finished by
300001	*Chris*	*Computers*	*30*	*1*	*Coventry*	*30 Nov 2001*
300002	*Alex*	*Oil Drums*	*950*	*3*	*London*	*20 Nov 2001*
300003	*Alex*	*Oil Drums*	*40*	*4*	*London*	*30 Nov 2001*
300004	*Alex*	*Screws*	*1000*	*5*	*Coventry*	*10 Nov 2001*

Table B: Local Redistribution Jobs within Warehouse Coventry

Job #	Foreman	Items	Qty	From (P)	To (P)	Moving Time	Forklift
000001	*Susan*	*Computers*	*60*	*4*	*5*	*09:30-10:30*	*Lucy*
000002	*Bill*	*Oil Drums*	*125*	*3*	*1*	*11:00-12:00*	*Tim*
000003	*Susan*	*Screws*	*1930*	*1*	*4*	*15:30-16:30*	*Tim*
000004	*Bill*	*Bananas*	*900*	*3*	*2*	*14:30-15:30*	*Lucy*
Within Warehouse: Coventry		*Date: 10 November 2001*					

Table C: Redistribution Jobs from Warehouse Coventry (10 November 2001)

Job #	Fore-man	Items	Qty	From (P)	To Ware-house	Loading Time	Plat-form	Driver	Fork-lift
100002	*Susan*	*Bananas*	*1500*	*2*	*London*	*09:00-10:00*	*B*	*Gale*	*Tim*
100003	*Bill*	*Oil Drums*	*200*	*1*	*Manchester*	*11:00-12:00*	*A*	*Adam*	*Lucy*
100005	*Susan*	*Bananas*	*3000*	*3*	*London*	*13:00-14:00*	*C*	*David*	*Tim*
From Warehouse: Coventry		*Date: 10 November 2001*							

Table D: Redistribution Jobs to Warehouse Coventry (10 November 2001)

Job #	Items	Qty	From Ware-house	Plat-form	Unloading Time	Driver	To (P)	Forklift
200001	*Bananas*	*3000*	*London*	*A*	*17:00-18:00*	*David*	*2*	*Tim*
300001	*Computers*	*30*	*Manchester*	*C*	*17:00-18:00*	*Gale*	*4*	*Lucy*
To Warehouse: Coventry		*Date: 10 November 2001*						

Table E: Forklift Operator (Lucy) Daily Schedule (10 November 2001)

Job #	Job Kind	Items	Qty	From	To	Time	Driver	Job Done
000001	*Moving*	*Computers*	*60*	*4*	*5*	*09:00-10:00*	***	
100003	*Loading*	*Oil Drums*	*200*	*1*	*A*	*11:00-12:00*	*Adam*	
000004	*Moving*	*Bananas*	*900*	*3*	*2*	*14:30-15:30*	***	
300001	*Unloading*	*Computers*	*30*	*C*	*4*	*17:00-18:00*	*Gale*	

Table F: Platform Schedule Details

Plat-form	Job #	L/UL	Items	Qty	From/To Place	Time	Forklift	Driver
A	*100003*	*Loading*	*Oil Drums*	*200*	*1*	*11:00-12:00*	*Lucy*	*Adam*
A	*200001*	*Unloading*	*Bananas*	*3000*	*2*	*17:00-18:00*	*Tim*	*David*
B	*100002*	*Loading*	*Bananas*	*1500*	*2*	*09:00-10:00*	*Tim*	*Gale*
C	*100005*	*Loading*	*Bananas*	*3000*	*3*	*13:00-14:00*	*Tim*	*David*
C	*300001*	*Unloading*	*Computers*	*30*	*4*	*17:00-18:00*	*Lucy*	*Gale*
Date: 10 November 2001								

Table G: Items/Place Details

Place	Item	Qty	Item	Qty	Item	Qty	Item	Qty
1	*Bananas*	*550*	*Computers*	*100*	*Oil Drums*	*1230*	*Screws*	*300*
2	*Bananas*	*2100*	*Computers*	*45*	*Oil Drums*	*0*	*Screws*	*3000*
3	*Bananas*	*450*	*Computers*	*92*	*Oil Drums*	*240*	*Screws*	*0*

Table H: Truck Schedule (10 November 2001)

Driver	Job #	Warehouses	Time	Job #	Warehouses	Time
Adam	*100003*	*Coventry–Manchester*	*11:00-14:00*	*300002*	*Manchester–London*	*15:00-18:00*
David	*100005*	*Coventry–London*	*13:00-15:00*	*200001*	*London–Coventry*	*16:00-18:00*
Gale	*100002*	*Coventry–London*	*09:00-12:00*	*200003*	*London–Manchester*	*12:30-15:00*
	300001	*Manchester–Coventry*	*15:30-18:00*			

Table I: Driver (Gale) Daily Schedule (10 November 2001)

Job #	From Ware-house	Loading Time	Load' P'form	To Ware-house	Unload-ing Time	UnLoad' P'form	Items	Qty
100002	*Coventry*	*09:00-10:00*	*B*	*London*	*11:00-12:00*	*A*	*Bananas*	*1500*
200003	*London*	*12:30-13:00*	*A*	*Manchester*	*14:00-15:00*	*B*		
300001	*Manchester*	*15:30-16:00*	*C*	*Coventry*	*17:00-18:00*	*C*	*Computers*	*30*

APPENDIX D

Scripts for the Warehouse System

Script A: The Paper-Based Model for the Warehouse System

A.1 The DoNaLD and Scout Script for Figure 7.5 (in the Server Model)

```
%eden
OpenDisplay("door_screen", 600, 400);
proc display_screen : door_screen { DisplayScreen(&door_screen, "door_screen"); }

%scout
window warehousePict = {
   type: DONALD,
   pict: "WarehouseProcess",
   box: [{0, 0}, {600, 400}],
   xmin: 0, ymin: 400, xmax: 600, ymax: 0,
   bgcolor: "blue",
   fgcolor: "yellow",
   border: 1,
   sensitive: ON
};
display dispWarehouse;
dispWarehouse = < warehousePict >;
door_screen = dispWarehouse;

%donald
viewport WarehouseProcess
label wh1, fm1, ww1, fo1, off, dri
label wh2, fm2, ww2, fo2
line whl11, whl12, whl21, whl22
line l11, l12, l13, l21, l22, l23, l31, l32, l33, l41, l42, l43, l51, l52, l53
line l61, l62, l63, l71, l72, l73, l81, l82, l83, l91, l92, l93
label l14, l24, l34, l44, l54, l64, l74, l84, l94
real ptX, ptY

wh1 = label("Warehouse A",    {500,  50})
fm1 = label("(Foreman A)",    {500, 150})
ww1 = label("(W. Worker A)",  {500, 250})
fo1 = label("(Forklifter A)", {500, 350})
off = label("(Office)",       {300,  50})
dri = label("(Truck Driver)", {300, 250})
```

```
wh2 = label("Warehouse B",    {100,  50})
fm2 = label("(Foreman B)",    {100, 150})
ww2 = label("(W. Worker B)",  {100, 250})
fo2 = label("(Forklifter B)", {100, 350})

whl11 = [{450, 60}, {550, 60}]
whl12 = [{450, 62}, {550, 62}]
whl21 = [{ 50, 60}, {150, 60}]
whl22 = [{ 50, 62}, {150, 62}]

l11 = [{550, 150}, {580, 150}]
l12 = [{580, 150}, {580, 245}]
l13 = [{580, 245}, {550, 245}]
l14 = label("RF1,2,3,4", {560, 200})

l21 = [{550, 255}, {580, 255}]
l22 = [{580, 255}, {580, 350}]
l23 = [{580, 350}, {550, 350}]
l24 = label("RF4", {580, 300})

l31 = [{450, 350}, {420, 350}]
l32 = [{420, 350}, {420, 150}]
l33 = [{420, 150}, {450, 150}]
l34 = label("RF4", {420, 300})

l41 = [{450, 250}, {400, 250}]
l42 = [{400, 250}, {400,  45}]
l43 = [{400,  45}, {340,  45}]
l44 = label("RF1,2,3", {420, 120})

%eden
func changeWidth { para lineNo;
   execute("
   A_l"// lineNo //"1 = 'linewidth=5';\n
   A_l"// lineNo //"2 = 'linewidth=5';\n
   A_l"// lineNo //"3 = 'linewidth=5';\n
   ");
};

A_wh1 = "color=red";   A_fm1 = "color=green"; A_ww1 = "color=green";
A_fo1 = "color=green"; A_off = "color=white"; A_dri = "color=white";
A_wh2 = "color=red";   A_fm2 = "color=green"; A_ww2 = "color=green";
A_fo2 = "color=green";
A_whl11 = "color=white"; A_whl12 = "color=white";
A_whl21 = "color=white"; A_whl22 = "color=white";
A_l14 = "color=gray"; A_l24 = "color=gray"; A_l34 = "color=gray";
A_l44 = "color=gray"; A_l54 = "color=gray"; A_l64 = "color=gray";
A_l74 = "color=gray"; A_l84 = "color=gray"; A_l94 = "color=gray";
```

A.2 Script for Redistribution Form (RF) shown in Figure 7.10

```
%scout
window f = {
   type: TEXT
   frame: ([{0, 0}, {500, 700}])
};
window RF1 = {
   type : TEXT
   frame: ([{20, 25}, {55, 35}])
   string: "RF 1"
   border: 1
   alignment: CENTRE
   sensitive: ON
   fgcolor: "blue"
   bgcolor: "yellow"
};
window RF2 = {
   type : TEXT
   frame: ([{65, 25}, {100, 35}])
   string: "RF 2"
   border: 1
   alignment: CENTRE
   sensitive: ON
   fgcolor: "blue"
   bgcolor: "yellow"
};
window RFfill = {
   type : TEXT
   frame: ([{20, 410}, {210, 422}])
   string: "Redistribution Forms Fill"
   border: 1
   alignment: CENTRE
   sensitive: ON
   fgcolor: "blue"
   bgcolor: "yellow"
};
window RFtitle = {
   type : TEXT
   frame: ([{40, 60}, {200, 70}])
   string: "REDISTRIBUTION FORM"
   alignment: CENTRE
   fgcolor: "yellow"
};
window RFcopy = {
   type : TEXT
   frame: ([{220, 60}, {255, 72}])
   string: "Copy:"
```

```
   alignment: LEFT
   fgcolor: "yellow"
};
window copyText = {
   type : TEXTBOX
   frame: ([{260, 60}, {5, 1}])
   fgcolor: "yellow"
};
window RFwarehouse = {
   type : TEXT
   frame: ([{25, 90}, {97, 102}])
   string: "Warwhouse:"
   alignment: LEFT
};
window warehouseText = {
   type : TEXTBOX
   frame: ([{100, 90}, {20, 1}])
   border: 1
   sensitive: ON
};

display screenRF = <RFtitle/RFcopy/copyText/RFwarehouse/warehouseText/RFforeman/
   foremanText/RFjobNo/jobNoText/RFitem/itemText/RFquantity/quantityText/RFfromPlace/
   fromPlaceText/RFtoWarwhouse/toWarehouseText/RFdate/dateText/RFforemanSignature/
   foremanSignatureText/RFsignatureDate/signatureDateText/RFredistribution/
   redistributionText/RFsubtitle1/RFsubtitle2/RFmovePending/movePendingText/RFloadTime/
   loadTimeText/RFloadPlatform/loadPlatformText/RFdriver/driverText/RFarrivalTime/
   arrivalTimeText/RFunloadPlatform/unloadPlatformText/RFsubtitle3/RFsubtitle4/
   RFonTransport/onTransportText/RFtoPlace/toPlaceText/RF1frame/RF2frame/RF3frame/
   RF4frame/RF/RFfill>;

%eden
m = RFv;
proc RF1_click { renewObs("mainRForm"); showRF(1); m = RFv1; };
proc RF2_click { renewObs("mainRForm"); showRF(2); m = RFv2; };
proc RF3_click { renewObs("mainRForm"); showRF(3); m = RFv3; };
proc RF4_click { renewObs("mainRForm"); showRF(4); m = RFv4; };
proc RF5_click { renewObs("mainRForm"); showRF(5); m = RFv5; };

proc RFfill_click {
   auto i;
   for (i = 1; i <= m#; i++) {
      sendServer(""," 
      mainRForm[" //str(m[i])// "][1] =\"" //str(warehouseText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][2] =\"" //str(foremanText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][3] =\"" //str(jobNoText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][4] =\"" //str(itemText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][5] =\"" //str(quantityText_getText())// "\";\n
```

```
      mainRForm[" //str(m[i])// "][6] =\"" //str(fromPlaceText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][7] =\"" //str(toWarehouseText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][8] =\"" //str(dateText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][9] =\"" //str(foremanSignatureText_getText())//"\";\n
      mainRForm[" //str(m[i])// "][10]=\"" //str(signatureDateText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][11]=\"" //str(redistributionText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][12]=\"" //str(movePendingText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][13]=\"" //str(loadTimeText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][14]=\"" //str(loadPlatformText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][15]=\"" //str(driverText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][16]=\"" //str(arrivalTimeText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][17]=\"" //str(unloadPlatformText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][18]=\"" //str(onTransportText_getText())// "\";\n
      mainRForm[" //str(m[i])// "][19]=\"" //str(toPlaceText_getText())// "\";\n
      ");
   };
   sendServer(""," 
   for (k = 1; k <= 5; k++) {\n
      for (l = 1; l <= 19; l++) {\n
         if (mainRForm[k][l] != '@') {} else {mainRForm[k][l]='';};
   };\n };\n");
};

func showRF { para copyNo;
   RForm is mainRForm;
   copyText_setText("* "//str(copyNo)//" *");
   warehouseText_setText(RForm[copyNo][1]);
   foremanText_setText(RForm[copyNo][2]);
   jobNoText_setText(RForm[copyNo][3]);
   itemText_setText(RForm[copyNo][4]);
   quantityText_setText(RForm[copyNo][5]);
   fromPlaceText_setText(RForm[copyNo][6]);
   toWarehouseText_setText(RForm[copyNo][7]);
   dateText_setText(RForm[copyNo][8]);
   foremanSignatureText_setText(RForm[copyNo][9]);
   signatureDateText_setText(RForm[copyNo][10]);
   redistributionText_setText(RForm[copyNo][11]);
   movePendingText_setText(RForm[copyNo][12]);
   loadTimeText_setText(RForm[copyNo][13]);
   loadPlatformText_setText(RForm[copyNo][14]);
   driverText_setText(RForm[copyNo][15]);
   arrivalTimeText_setText(RForm[copyNo][16]);
   unloadPlatformText_setText(RForm[copyNo][17]);
   onTransportText_setText(RForm[copyNo][18]);
   toPlaceText_setText(RForm[copyNo][19]);
};
```

A.3 Script for the Foreman Window (in the Client Model)

```
%eden
OpenDisplay("foremanA_screen", 1050, 500);
proc display_screen : foremanA_screen { DisplayScreen(&foremanA_screen,
   "foremanA_screen"); }
RFv = [1, 2, 3, 4, 5]; RFv1 = [1, 2, 3, 4, 5];
RFv2 = []; RFv3 = []; RFv4 = []; RFv5 = [];
renewObs("mainRForm");
renewObs("mainITableA");

%eden
include("/dcs/emp/cheny/PUBLIC/D-warehouse/RFscreen.e");
include("/dcs/emp/cheny/PUBLIC/D-warehouse/ITscreen.e");
include("/dcs/emp/cheny/PUBLIC/D-warehouse/ITscreenA.e");
include("/dcs/emp/cheny/PUBLIC/D-warehouse/whPicture.e");

%scout
window PassRF1234 = {
   type : TEXT
   frame: ([{20, 5}, {240, 15}])
   string: "Pass RF1...4 -> W. Worker A"
   border: 1
   alignment: CENTRE
   sensitive: ON
   fgcolor: "red"
   bgcolor: "white"
};

foremanA_screen = <RF1/RF2/RF3/RF4/RF5/PassRF1234/ITbotton> & screenRF & screenIT & <f> & dispWH;

%eden
proc PassRF1234_click : PassRF1234_mouseClick {
   include("/dcs/emp/cheny/PUBLIC/D-warehouse/foremanAPassClick.e");
};

          /*** The following is the script of 'foremanAPassClick.e' ***/
%eden
sendClient("foremanA," %scout\n
   foremanA_screen = <RF5/ITbotton> & screenRF & screenIT & <f> & dispWH;\n
");

sendClient("workerA," %scout\n
   workerA_screen = <RF1/RF2/RF3/RF4/PassRF123/PassRF4/ITbotton/PTbotton> &
   screenRF & screenIT & screenPT & screemWorkerA & dispWH;\n
");

sendServer("","changeWidth('1');");
```

Script B: The EDDI Table Defined in the Server Model

```
%eddi
AllTable is (Coventry + London + Manchester);
```

Script C: The Foreman (in Warehouse Coventry) Script

C.1 The EDDI Table

```
%eddi
Coventry (FromHouse char, JobNo int, Foreman char, Items char, Qty int, FromPlace int, ToWarehouse char,
    FinishedBy char, R_Date char, L_Pform char, L_Time_From char, L_Time_To char, L_Forklift char,
    Driver char, UL_Pform char, To_Place char, UL_Time_From char, UL_Time_To char, UL_Forklift char,
    JobDone char);
Coventry << ["Coventry",100001,"Bill","Screws",550,4,"London","30/11/01","","","","",
   "","","","","","","",""];
Coventry << ["Coventry",100002,"Susan","Bananas",1500,2,"London","10/11/01","","","",
   "","","","","","","","",""];
Coventry << ["Coventry",100003,"Bill","Oil Drums",200,1,"Manchester","20/11/01","","",
   "","","","","","","","","",""];
Coventry << ["Coventry",100004,"Susan","Computers",120,5,"London","20/11/01","","","",
   "","","","","","","","",""];
```

C.2 The Window Interface and the Functions of Buttons Defined by EDEN

```
%eden
OpenDisplay("coventry_screen", 500, 500);
proc display_screen : coventry_screen { DisplayScreen(&coventry_screen, "coventry_screen"); }
include("/dcs/emp/cheny/PUBLIC/Warehouse-System/foremanWindow.e");
jobNumberC = 100005;

proc addButton_click : addButton_mouse_1 {
   if(addButton_mouse_1[2]==4) {
      changeData();
      execute(%eddi\n
         Coventry << [\"Coventry\","
         // str(jobNumberC) // ",\"" // c2 // "\",\""" // c3 // "\"," // c4 // ","
         // c5 // ",\"" // c6 // "\,\""" // c7
         // "\,\"\"",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\",\"\"];";
      jobNumberC++;
   };
};

proc viewData_click : viewData_mouse_1 {
   if(viewData_mouse_1[2]==4) {
      c8 = jobNoModifyText_getText();
```

```
      for (i = 2; i <= (Coventry#); i++) {
         if (Coventry[i][2] == int(c8)) {
            j = i;
            jobNoText_setText(str(Coventry[j][2]));
            foremanText_setText(str(Coventry[j][3]));
            itemText_setText(str(Coventry[j][4]));
            quantityText_setText(str(Coventry[j][5]));
            fromPlaceText_setText(str(Coventry[j][6]));
            toWarehouseText_setText(str(Coventry[j][7]));
            dateText_setText(str(Coventry[j][8]));
         };
      };
   };
};

proc modify_click : modify_mouse_1 {
   if(modify_mouse_1[2]==4) {
      c8 = jobNoModifyText_getText();
      for (i = 2; i <= (Coventry#); i++) {
         if (Coventry[i][2] == int(c8)) {
            j = str(i);
            changeData();
            Coventry[j][3] = c2;   Coventry[j][4] = c3;
            Coventry[j][5] = c4;   Coventry[j][6] = c5;
            Coventry[j][7] = c6;   Coventry[j][8] = c7;
            break;
         };
      };
   };
};

func changeData {
      c2 = foremanText_getText();     c3 = itemText_getText();
      c4 = quantityText_getText();    c5 = fromPlaceText_getText();
      c6 = toWarehouseText_getText(); c7 = dateText_getText();
};

proc cancel_click : cancel_mouse_1 {
   if(cancel_mouse_1[2] == 4) {
      jobNoText_setText(str(jobNumberC));
      clearTable();
   };
};

func clearTable {
   foremanText_setText("");      itemText_setText("");
   quantityText_setText("");     fromPlaceText_setText("");
   toWarehouseText_setText("");  dateText_setText("");
```

```
   jobNoModifyText_setText("");
};

proc viewTable1_click : viewTable1_mouse_1 {
   if(viewTable1_mouse_1[2]==4) {
      execute("%eddi\n
         ?Coventry %JobNo,Foreman,Items,Qty,FromPlace,ToWarehouse,FinishedBy;");
   };
};

proc viewTable2_click : viewTable2_mouse_1 {
   if(viewTable2_mouse_1[2]==4) {
      execute("%eddi\n  ?PlaceDetailsCoventry;");
   };
};
```

Script D: The Office Personnel Script

D.1 The EDDI Tables

```
%eddi
FromCoventry is AllTable: FromHouse=='Coventry' %JobNo,FromHouse,ToWarehouse,
   FinishedBy,R_Date,Driver,L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
FromLondon is AllTable: FromHouse=='London' %JobNo,FromHouse,ToWarehouse,
   FinishedBy,R_Date,Driver,L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
ToCoventry is AllTable: ToWarehouse=='Coventry' %JobNo,FromHouse,ToWarehouse,
   FinishedBy,R_Date,Driver,L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
ToLondon is AllTable: ToWarehouse=='London' %JobNo,FromHouse,ToWarehouse,
   FinishedBy,R_Date,Driver,L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
Adam is AllTable: Driver=="Adam" %JobNo,FromHouse,ToWarehouse,R_Date,Driver,
   L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
David is AllTable: Driver=="David" %JobNo,FromHouse,ToWarehouse,R_Date,Driver,
   L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
Gale is AllTable: Driver=="Gale" %JobNo,FromHouse,ToWarehouse,R_Date,Driver,
   L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;
```

D.2 The Window Interface and the Functions of Buttons Defined by EDEN

```
%eden
OpenDisplay("office_screen", 500, 500);
proc display_screen : office_screen { DisplayScreen(&office_screen, "office_screen"); }
include("/dcs/emp/cheny/PUBLIC/Warehouse-System/officeWindow.e");

proc viewData_click : viewData_mouse_1 {
   if(viewData_mouse_1[2]==4) {
      renewObs("AllTable");
```

```
      c8 = jobNoModifyText_getText();
      for (i = 2; i <= (AllTable#); i++) {
         if (AllTable[i][2] == int(c8)) {
            j = i;
            jobNoText_setText(str(AllTable[j][2]));
            foremanText_setText(str(AllTable[j][3]));
            fromPlaceText_setText(str(AllTableAllTable[j][1]));
            itemText_setText(str(AllTable[j][4]));
            quantityText_setText(str(AllTable[j][5]));
            toWarehouseText_setText(str(AllTable[j][7]));
            dateText_setText(str(AllTable[j][8]));
            OfficeRDateText_setText(str(AllTable[j][9]));
            OfficeLTFromText_setText(str(AllTable[j][11]));
            OfficeLTToText_setText(str(AllTable[j][12]));
            unloadTFromText_setText(str(AllTable[j][17]));
            unloadTToText_setText(str(AllTable[j][18]));
            driverText_setText(str(AllTable[j][14]));
         };
      };
   };
};

proc modify_click : modify_mouse_1 {
   if(modify_mouse_1[2]==4) {
      c8 = jobNoModifyText_getText();
      for (i = 2; i <= (AllTable#); i++) {
         if (AllTable[i][2] == int(c8)) {
            j = str(i);
            changeData();
            sendServer(""," 
               AllTable[" // j // "][9]  = \"" // str(c2) // "\";\n
               AllTable[" // j // "][11] = \"" // str(c3) // "\";\n
               AllTable[" // j // "][12] = \"" // str(c4) // "\";\n
               AllTable[" // j // "][17] = \"" // str(c5) // "\";\n
               AllTable[" // j // "][18] = \"" // str(c6) // "\";\n
               AllTable[" // j // "][14] = \"" // str(c7) // "\";
            ");
            break;
         };
      };
   };
};

func changeData {
      c2 = OfficeRDateText_getText();  c3 = OfficeLTFromText_getText();
      c4 = OfficeLTToText_getText();   c5 = unloadTFromText_getText();
      c6 = unloadTToText_getText();    c7 = driverText_getText();
};
```

```
proc clear_click : clear_mouse_1 {
   if (clear_mouse_1[2]==4) {
      OfficeRDateText_setText("");    OfficeLTFromText_setText("");
      OfficeLTToText_setText("");     unloadTFromText_setText("");
      unloadTToText_setText("");      driverText_setText("");
   };
};

proc viewCoventry1_click : viewCoventry1_mouse_1 {
   if(viewCoventry1_mouse_1[2]==4) {
      execute("%eddi\n  ?FromCoventry");
   };
};

proc viewLondon1_click : viewLondon1_mouse_1 {
   if(viewLondon1_mouse_1[2]==4) {
      execute("%eddi\n  ?FromLondon");
   };
};

proc viewCoventry2_click : viewCoventry2_mouse_1 {
   if(viewCoventry2_mouse_1[2]==4) {
      execute("%eddi\n  ?ToCoventry");
   };
};

proc viewLondon2_click : viewLondon2_mouse_1 {
   if(viewLondon2_mouse_1[2]==4) {
      execute("%eddi\n  ?ToLondon");
   };
};

proc go1_click : go1_mouse_1 {
   if(go1_mouse_1[2]==4) {
      renewObs("AllTable");
      c8 = finishByInViewText_getText();
      execute("%eddi\n  ?AllTable:FinishedBy==\"" // str(c8) // "\"%JobNo,FromHouse,
         ToWarehouse,FinishedBy,R_Date,Driver,L_Time_From,L_Time_To, UL_Time_From,
         UL_Time_To;");
   };
};

proc viewAdam1_click : viewAdam1_mouse_1 {
   if(viewAdam1_mouse_1[2]==4) {
      execute("%eddi\n  ?Adam");
   };
};
```

```
proc go2_click : go2_mouse_1 {
   if(go2_mouse_1[2]==4) {
      renewObs("AllTable");
      c8 = viewDriverInViewText_getText();
      execute("%eddi\n  ?AllTable:R_Date==\"" // str(c8) // "\"%JobNo,R_Date,Driver,
         FromHouse,ToWarehouse,L_Time_From,L_Time_To, UL_Time_From, UL_Time_To;");
   };
};
```

Script E: The Warehouse Worker (in Warehouse Coventry) Script

E.1 The EDDI Tables

```
%eddi
WorkerInCoventry is ((AllTable: FromHouse == "Coventry")
   + (AllTable: ToWarehouse == "Coventry")) - (AllTable: R_Date == "");
PlatformCoventry is ((AllTable: FromHouse == "Coventry")
   + (AllTable: ToWarehouse == "Coventry"))
   - (AllTable: R_Date == "").((AllTable: L_Pform != "")+(AllTable: UL_Pform != ""));
Lucy is (AllTable: L_Forklift == "Lucy") + (AllTable: UL_Forklift == "Lucy");
Tim is (AllTable: L_Forklift == "Tim") +(AllTable: UL_Forklift == "Tim");
```

E.2 The Window Interface and Functions of Buttons Defined by EDEN

```
%eden
OpenDisplay("coventryWorker_screen", 500, 500);
proc display_screen : coventryWorker_screen { DisplayScreen(&coventryWorker_screen,
    "coventryWorker_screen"); }
include("/dcs/emp/cheny/PUBLIC/Warehouse-System/workerWindow.e");

proc viewData_click : viewData_mouse_1 {
   if (viewData_mouse_1[2]==4) {
      c8 = jobNoModifyText_getText();
      for (i = 2; i <= (WorkerInCoventry#); i++) {
         if (WorkerInCoventry[i][2] == int(c8)) {
            j = i;
            jobNoText_setText(str(WorkerInCoventry[j][2]));
            foremanText_setText(str(WorkerInCoventry[j][3]));
            itemText_setText(str(WorkerInCoventry[j][4]));
            quantityText_setText(str(WorkerInCoventry[j][5]));
            fromPlaceText_setText(str(WorkerInCoventry[j][1]));
            rDateText_setText(str(WorkerInCoventry[j][9]));
            LTFromText_setText(str(WorkerInCoventry[j][11]));
            LTToText_setText(str(WorkerInCoventry[j][12]));
            ULTFromText_setText(str(WorkerInCoventry[j][17]));
            ULTToText_setText(str(WorkerInCoventry[j][18]));
```

```
            loadPlatformText_setText(str(WorkerInCoventry[j][10]));
            loadForkliftText_setText(str(WorkerInCoventry[j][13]));
            unloadPlatformText_setText(str(WorkerInCoventry[j][15]));
            unloadForkliftText_setText(str(WorkerInCoventry[j][19]));
            toPlaceText_setText(str(WorkerInCoventry[j][16]));
         };
      };
   };
};

proc modify_click : modify_mouse_1 {
   if (modify_mouse_1[2]==4) {
      c8 = jobNoModifyText_getText();
      for (i = 2; i <= (AllTable#); i++) {
         if (AllTable[i][2] == int(c8)) {
            j = str(i);
            changeData();
            sendServer(""," 
               AllTable[" // j // "][10] = \"" // str(c2) // "\";\n
               AllTable[" // j // "][13] = \"" // str(c3) // "\";\n
               AllTable[" // j // "][15] = \"" // str(c4) // "\";\n
               AllTable[" // j // "][19] = \"" // str(c5) // "\";\n
               AllTable[" // j // "][16] = \"" // str(c6) // "\";
            ");
            break;
         };
      };
   };
};

func changeData {
      c2 = loadPlatformText_getText();     c3 = loadForkliftText_getText();
      c4 = unloadPlatformText_getText();   c5 = unloadForkliftText_getText();
      c6 = toPlaceText_getText();
};

proc cancel_click : cancel_mouse_1 {
   if(cancel_mouse_1[2] == 4) {
      loadPlatformText_setText("");     loadForkliftText_setText("");
      unloadPlatformText_setText("");   unloadForkliftText_setText("");
      toPlaceText_setText("");
   };
};

proc viewAll_click : viewAll_mouse_1 {
   if(viewAll_mouse_1[2]==4) {
      execute("%eddi\n  ?WorkerInCoventry:FromHouse == 'Coventry' %JobNo,R_Date,
         FromHouse,L_Time_From,L_Time_To;");
```

```
      execute("%eddi\n  ?WorkerInCoventry:ToWarehouse == 'Coventry'
         %JobNo,R_Date,ToWarehouse,UL_Time_From,UL_Time_To;");
   };
};

proc viewLucy1_click : viewLucy1_mouse_1 {
   if(viewLucy1_mouse_1[2]==4) {
      execute("%eddi\n  ?Lucy: L_Forklift == 'Lucy'
         %JobNo,R_Date,L_Forklift,L_Time_From,L_Time_To;");
      execute("%eddi\n  ?Lucy: UL_Forklift == 'Lucy'
         %JobNo,R_Date,UL_Forklift,UL_Time_From,UL_Time_To;");
   };
};

proc viewTim1_click : viewTim1_mouse_1 {
   if(viewTim1_mouse_1[2]==4) {
      execute("%eddi\n  ?Tim:L_Forklift == 'Tim'
         %JobNo,R_Date,L_Forklift,L_Time_From,L_Time_To;");
      execute("%eddi\n  ?Tim:UL_Forklift == 'Tim'
         %JobNo,R_Date,UL_Forklift,UL_Time_From,UL_Time_To;");
   };
};

proc go1_click : go1_mouse_1 {
   if(go1_mouse_1[2]==4) {
      c8 = byDateInViewText_getText();
      execute("%eddi\n  ?PlatformCoventry:R_Date==\"" // str(c8) // "\"%JobNo,R_Date,
         Driver,L_Time_From,L_Time_To,L_Pform,UL_Time_From,UL_Time_To,UL_Pform;");
   };
};

proc viewItems_click : viewItems_mouse_1 {
   if(viewItems_mouse_1[2]==4) {
      execute("%eddi\n  ?PlaceDetailsCoventry;");
   };
};
```