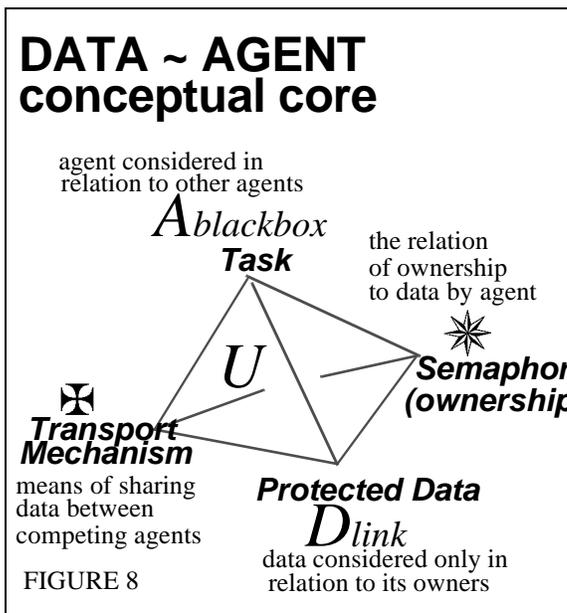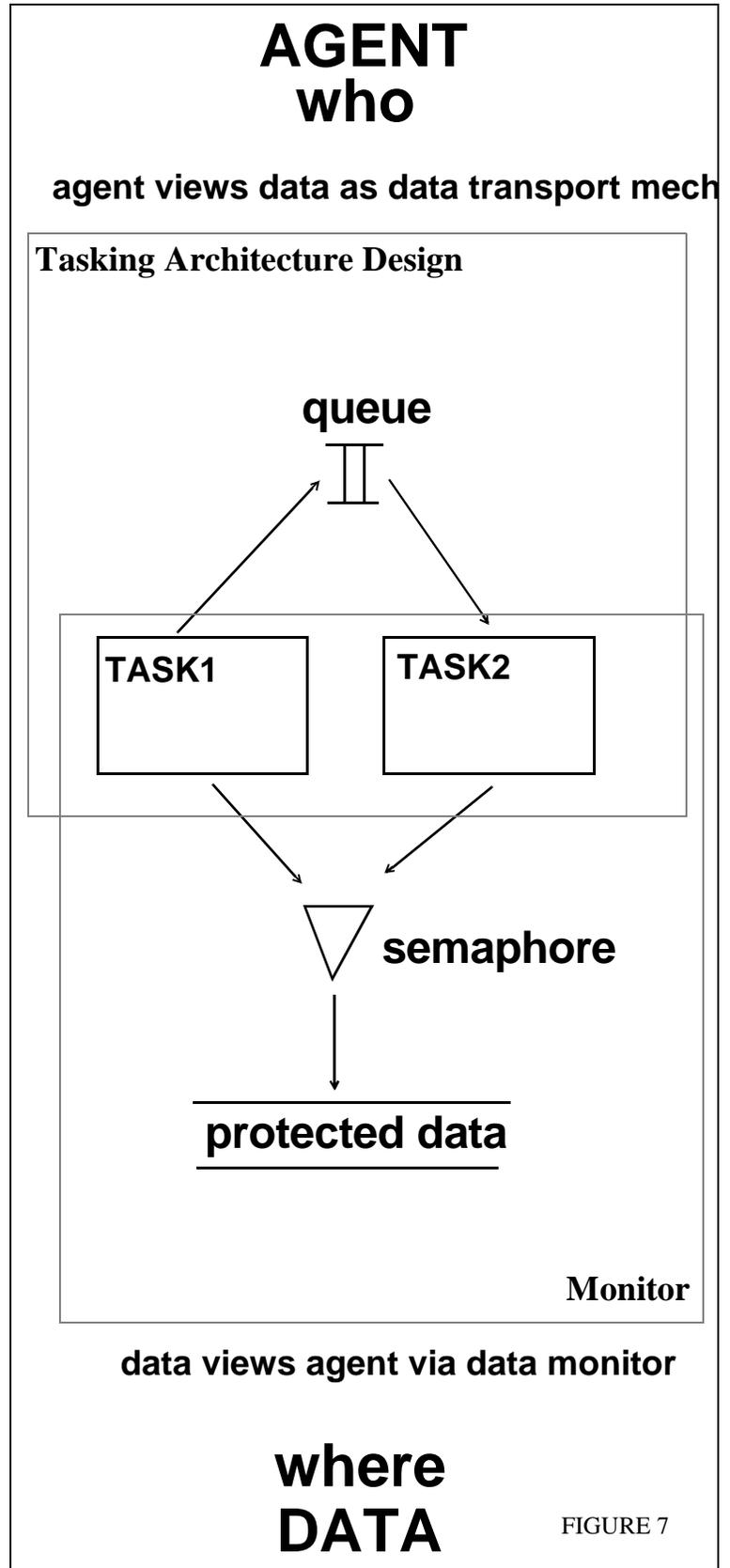# APPENDIX 1

Graphical Depiction of
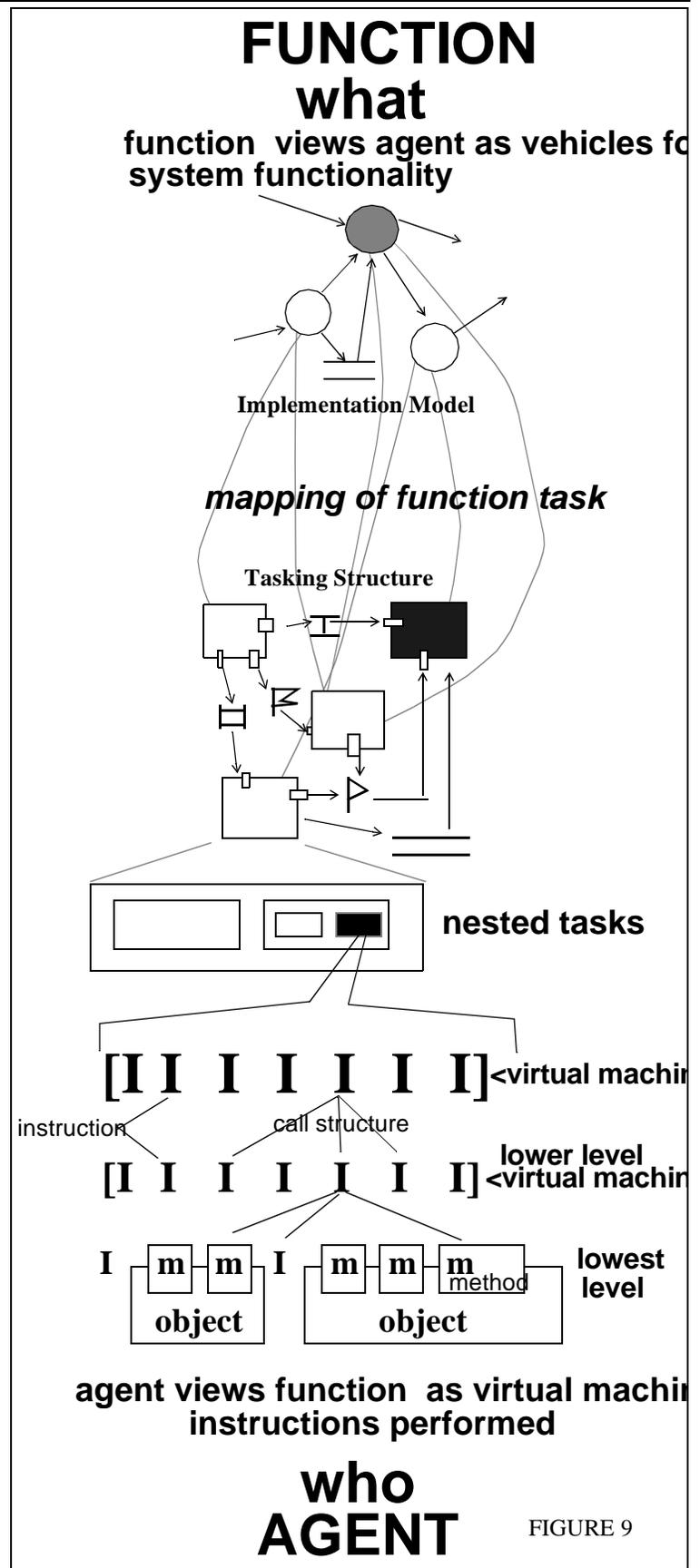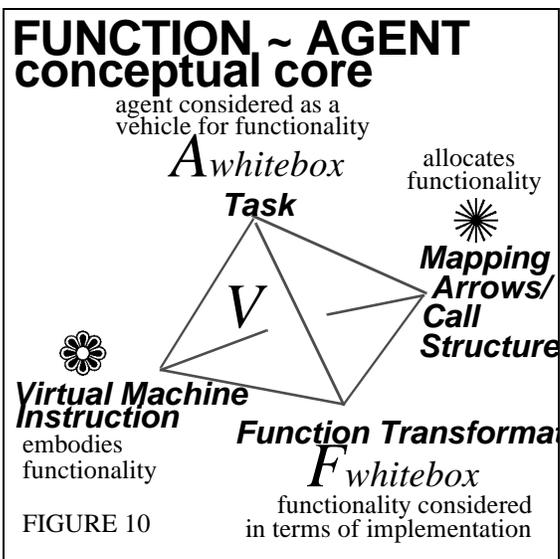
for Software Design Minimal Methods

Agent ~ Data minimal methods and conceptual core

o Tasking Architecture minimal
   method defined by Gomaa in
   DARTS and elaborated by
   SPC/Gomaa in ADARTS

o Monitor minimal method defined
   by Hoare

o These two minimal methods
   form a two way bridge.

AGENT
who

agent views data as data transport mech

Tasking Architecture Design

queue

TASK1   **TASK2**

semaphore

protected data

Monitor

data views agent via data monitor

where
DATA

FIGURE 7

DATA ~ AGENT
conceptual core

agent considered in
relation to other agents

$A_{blackbox}$
Task

the relation
of ownership
to data by agent

U

**Semaphor
(ownership)**

✠
**Transport
Mechanism**

means of sharing
data between
competing agents

Protected Data
D_{link}

data considered only in
relation to its owners

FIGURE 8

Function ~ Agent minimal methods and conceptual core

o Mapping minimal method defined by Mellor/Ward in STRUCTURED DEVELOPMENT FOR REAL-TIME SYSTEMS Volume 3

o Virtual Layered Machine minimal method defined by Nielsen/Shumate in DESIGNING LARGE REAL-TIME SYSTEMS WITH ADA. Method called VLM/OOD. [See also OOSD by Constantine/Wasserman.]

o Methods do not combine to form a two way bridge.

o Tasking structure mediates between these two minimal methods.

o Lowest level virtual machine instructions are methods of objects.

**FUNCTION
what**
function views agent as vehicles for system functionality

Implementation Model

mapping of function task

Tasking Structure

nested tasks

[I I   I   I   I   I   I]**<virtual machine**

instruction   call structure

[I I   I   I   I   I   I]**<virtual machine**

**lower level
<virtual machine**

I   m m   I   m   m   m method   **lowest
level**

object   object

agent views function as virtual machine instructions performed

**who
AGENT**   FIGURE 9

**FUNCTION ~ AGENT
conceptual core**
agent considered as a vehicle for functionality

$A_{whitebox}$ allocates functionality

Task

Mapping Arrows/ Call Structure

V

Virtual Machine Instruction
embodies functionality

Function Transformation
$F_{whitebox}$
functionality considered in terms of implementation

FIGURE 10

Function ~ Data minimal methods and conceptual core

o Dataflow minimal method
    defined by Yourdon/DeMarco

o Object transform minimal
    method defined by Guttag/
    Liskov

o Transform between these two
    methods demonstrated by
    Vaclav Rajlich

o Two minimal methods form a
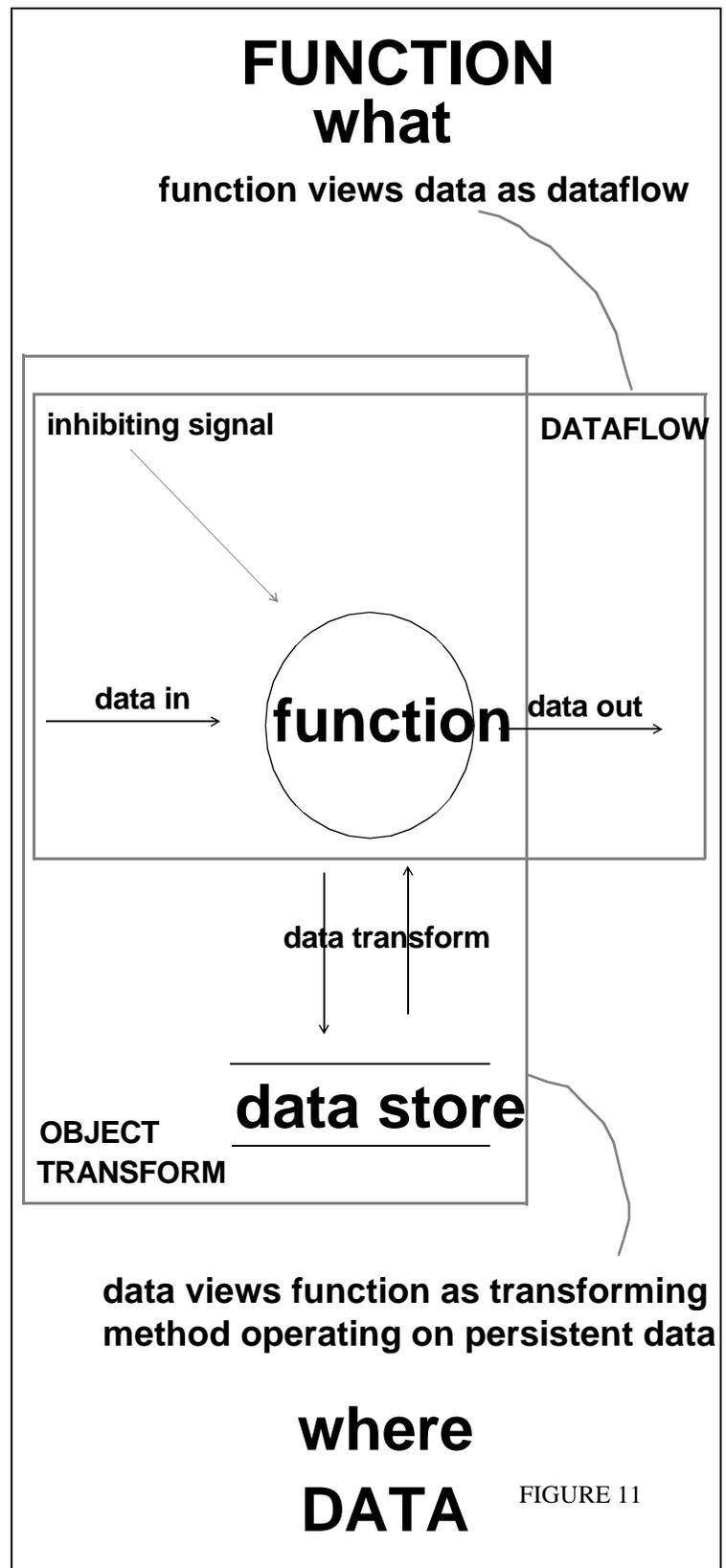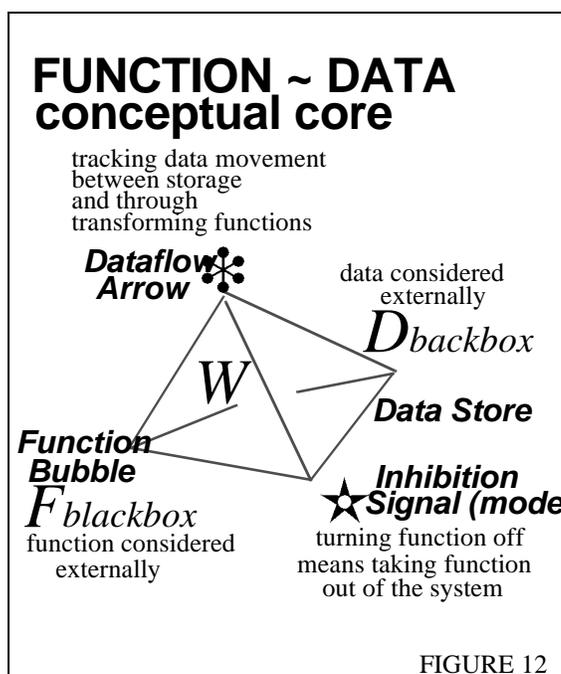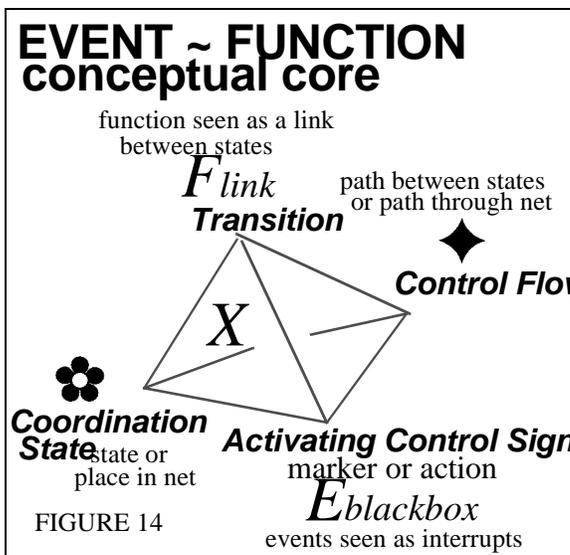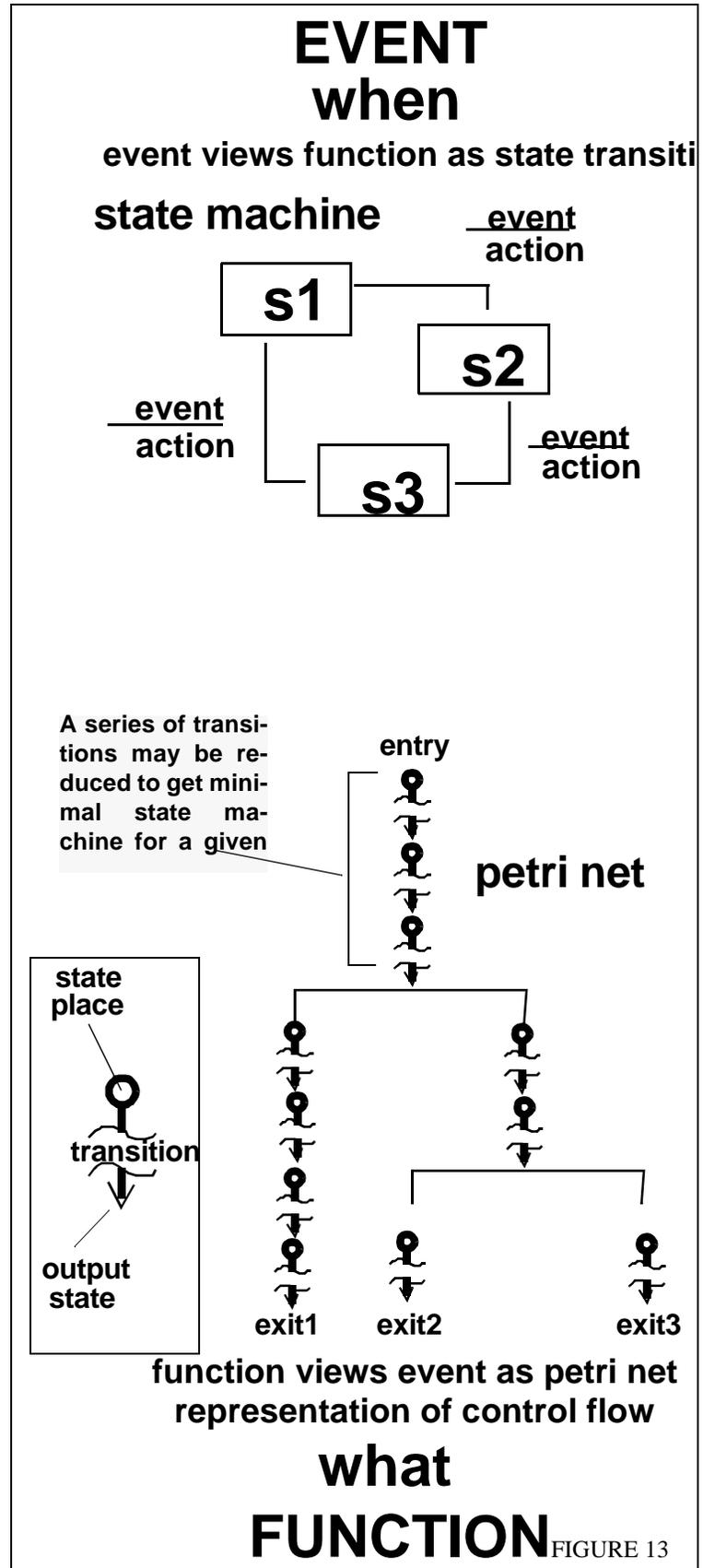   conceptual bridge.

FUNCTION
what

function views data as dataflow

inhibiting signal   **DATAFLOW**

data in   **function**   **data out**

data transform

data store

OBJECT
TRANSFORM

data views function as transforming
method operating on persistent data

where

DATA   FIGURE 11

FUNCTION ~ DATA
conceptual core

tracking data movement
between storage
and through
transforming functions

Dataflow
Arrow   data considered
                            externally

W $D_{backbox}$

                        Data Store

Function
Bubble

$F_{blackbox}$ ***Inhibition***
                            Signal (mode)
function considered   turning function off
    externally   means taking function
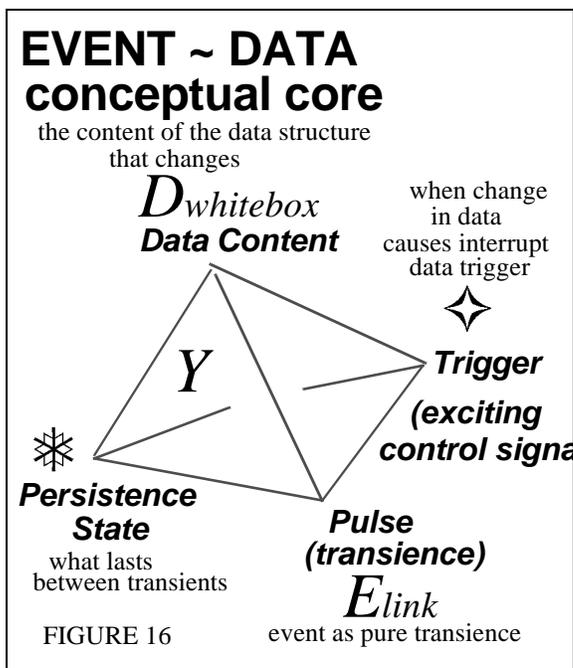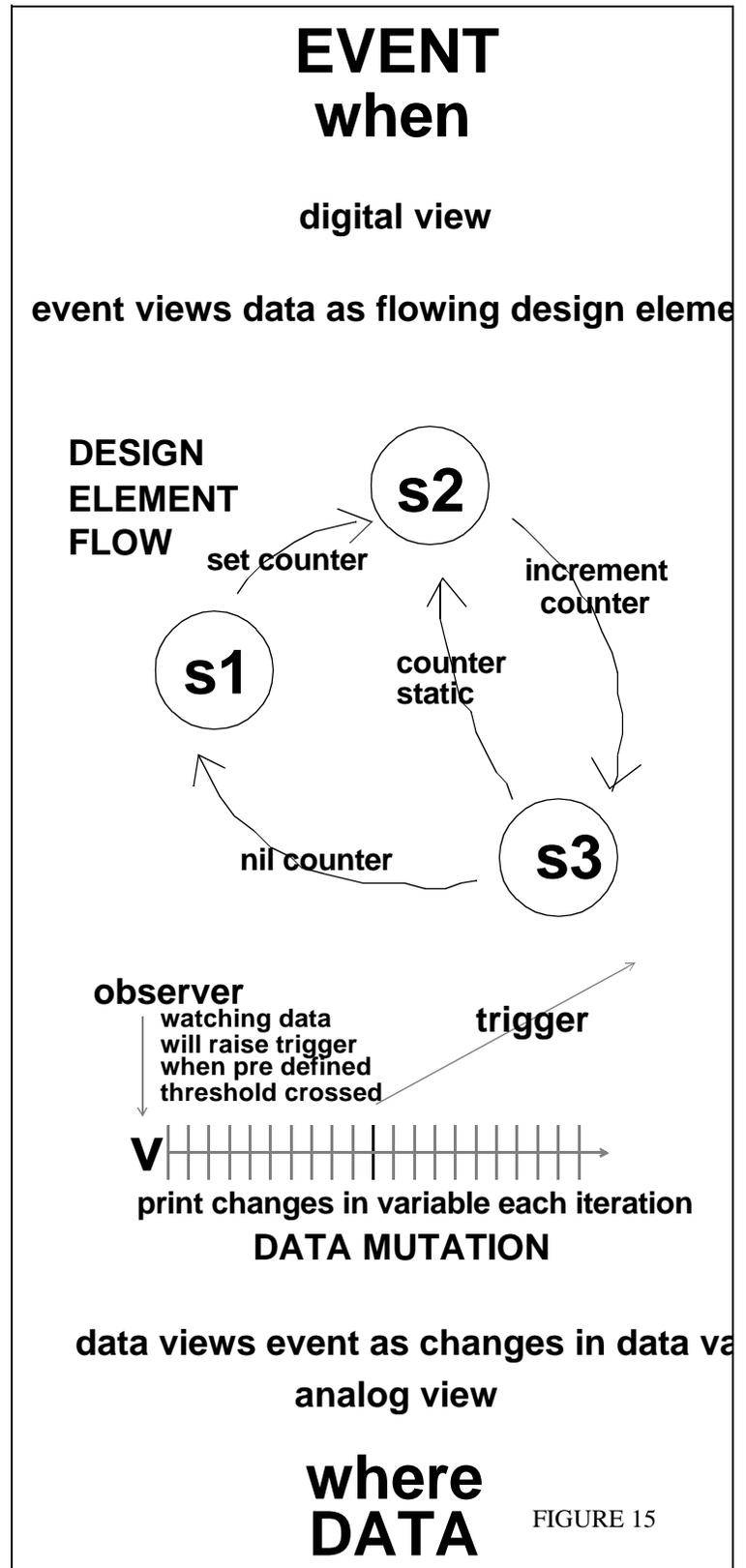   out of the system

FIGURE 12

Event ~Function minimal methods and conceptual core

o State machine minimal method defined by Ward/ Mellor and Hatley/ Pribhai

o PetriNet method defined by Petri and has been developed by many others.

o Two methods do not form a two way bridge, but they are closely related by the presence of states and actions in both methods.

EVENT
when

event views function as state transiti

state machine

event action

s1

s2

event action

event action

s3

A series of transi-tions may be re-duced to get mini-mal state ma-chine for a given

entry

petri net

state place

transition

output state

exit1   exit2   exit3

function views event as petri net representation of control flow

what

FUNCTION FIGURE 13

EVENT ~ FUNCTION
conceptual core

function seen as a link
between states

F_{link} path between states
or path through net

Transition

Control Flow

X

Coordination
State state or
place in net

Activating Control Sign
marker or action
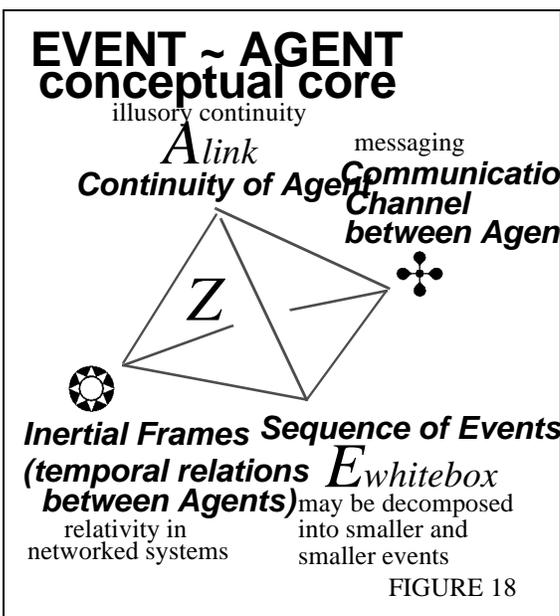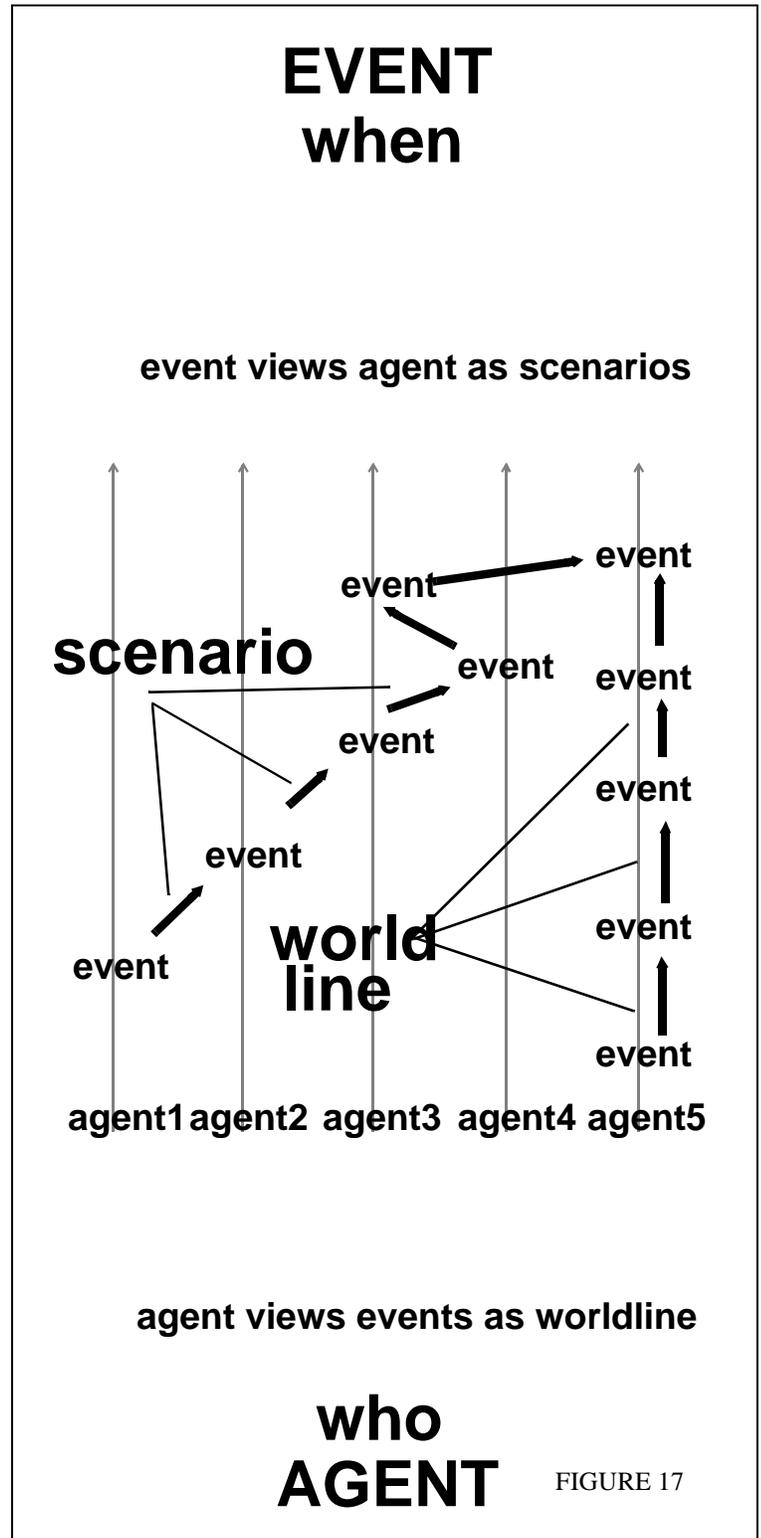
$E_{blackbox}$
events seen as interrupts

FIGURE 14

Event ~ Data minimal methods and conceptual core

o Design element flow diagram sees design elements as flowing through the states of the application instead of as a clock work mechanism. Not defined as a standard method. Allows coordination of state machines at different system levels.

o Data mutation sees the changes in individual data items as the program executes. This is the most basic method for all computer programming. Can either track information flow through the system or compare cross sections of different information flows.

o These minimal methods do not form a two way bridge.

EVENT
when

digital view

event views data as flowing design eleme

DESIGN ELEMENT FLOW

s1

s2

s3

set counter

increment counter

counter static

nil counter

observer
watching data
will raise trigger
when pre defined
threshold crossed

trigger

V

print changes in variable each iteration

DATA MUTATION

data views event as changes in data va

analog view

where DATA

FIGURE 15

EVENT ~ DATA conceptual core
the content of the data structure
that changes

$D_{whitebox}$
Data Content

when change
in data
causes interrupt
data trigger

Y

Trigger

(exciting control signa

❋
Persistence State
what lasts
between transients

Pulse (transience)

E_{link}
event as pure transience

FIGURE 16

Event ~ Agent minimal methods and conceptual core

o Worldline minimal method defined by Agha in ACTORS.

o Scenario minimal method used in practice but not explicitly defined as a formal method.

o These two minimal methods form a two way bridge.

EVENT
when

event views agent as scenarios

scenario

event

event

event

event

event

event

event

event

event

world
line

event

agent1 agent2 agent3 agent4 agent5

agent views events as worldline

who
AGENT

FIGURE 17

EVENT ~ AGENT
conceptual core

illusory continuity

A_{link}

Continuity of Agent

messaging

Communicatio
Channel
between Agen

Z

Inertial Frames
(temporal relations
between Agents)

Sequence of Events

$E_{whitebox}$

relativity in
networked systems

may be decomposed
into smaller and
smaller events

FIGURE 18

Apeiron Press

PO Box 4402
Garden Grove,
California 92842-4402

714-638-7376
714-638-1210
palmer@think.net
palmer@netcom.com
palmer@exo.com
Dataline 714-638-0876

This book was set using
Framemaker document
publishing software by the
author.

Electronic Version in Adobe
Acrobat PDF available at http://
server.snni.com:80/~palmer/
homepage.html

Keywords:

Software, Design Methods,
Ontology, Integral Software
Engineering Methodology,
Systems Theory

DRAFT