
Support: Domain Engineering: Mgt. and Org.

Management and Organisation

- In this appendix we illustrate three aspects of the management and organisation facet.
 - ★ Two models suggest how “layers” of management collaborate and in two different styles and at two different levels of detail:
 - ◇ as a simple functional model, Slides 882–895, and
 - ◇ as a not quite so simple model based on communicating sequential processes Slides 896–928.
 - The two formal models
 - ◇ share the same basic narrative.
 - ◇ That narrative is given on Slides 882–885.
 - ★ One model illustrates organisational aspects (Slides 934–951).

A Simple, Functional Description of Management

- By a functional description we mean
 - ★ a description which focuses on functions,
 - ★ that is, which explains things in terms of functions.
- By a simple description we mean
 - ★ a description which is is short.

[A Simple, Functional Description of Management]

A Base Narrative

- We think of
 - ★ (i) strategic, (ii) tactic, and (iii) operational managers
 - ★ as well as (iv) supervisors, (v) team leaders and the rest of the (vi) staff (i.e., workers)of a domain enterprise as functions.
- To make the description simple
 - ★ we think of each of the six categories (i–vi) of personnel and staff
 - ★ as functions, that is, there are six major domain functions
 - ★ related to management.

[A Simple, Functional Description of Management, A Base Narrative]

- Each category of staff, i.e., each function,
 - ★ works in state and
 - ★ updates that state
 - ★ according to schedules and resource allocations —
 - ★ which are considered part of the state.
- To make the description simple
 - ★ we do not detail the state
 - ★ other than saying that each category
 - ★ works on an “instantaneous copy” of “the” state.

[A Simple, Functional Description of Management, A Base Narrative]

- Now think of six staff category activities,
 - ★ strategic managers,
 - ★ tactical managers,
 - ★ operational managers,
 - ★ supervisors,
 - ★ team leaders and
 - ★ workers
- as six simultaneous sets of actions.
- Each function defines a step of collective (i.e., group)
 - ★ (strategic,
 - ★ tactical,
 - ★ operational) management,
 - ★ supervisor,
 - ★ team leader and
 - ★ workerwork.
- Each step is considered “atomic”.

[A Simple, Functional Description of Management, A Base Narrative]

- Now think of an enterprise as the “repeated” step-wise simultaneous performance of these category activities.
 - ★ Six “next” states arise.
 - ★ These are, in the reality of the domain, ameliorated, that is reconciled into one state.
 - ★ however with the next iteration, i.e., step, of work
 - ◇ having each category apply its work to a reconciled version of the state
 - ◇ resulting from that category’s previously yielded state and the mediated “global” state.
- We refer to Slide 348 for a caveat:
 - ★ The doubly recursive definition of the **enterprise** function is a pseudo-definition.
 - ★ It is not a mathematically proper definition.

[A Simple, Functional Description of Management]

A Formalisation

type

0. Σ

value

1. str, tac, opr, sup, tea, wrk: $\Sigma \rightarrow \Sigma$
 2. ame_s, ame_t, ame_o, ame_u, ame_e, ame_w: $\Sigma \rightarrow \Sigma^5 \rightarrow \Sigma$
 3. objective: $\Sigma^6 \rightarrow \mathbf{Bool}$
 4. enterprise, ameliorate: $\Sigma^6 \rightarrow \Sigma$
 5. enterprise($\langle \sigma_s, \sigma_t, \sigma_o, \sigma_u, \sigma_e, \sigma_w \rangle$) \equiv
 6. **let** $\sigma'_s = \text{ame_s}(\text{str}(\sigma_s))(\langle \sigma'_t, \sigma'_o, \sigma'_u, \sigma'_e, \sigma'_w \rangle)$,
 7. $\sigma'_t = \text{ame_t}(\text{tac}(\sigma_t))(\langle \sigma'_s, \sigma'_o, \sigma'_u, \sigma'_e, \sigma'_w \rangle)$,
 8. $\sigma'_o = \text{ame_o}(\text{opr}(\sigma_o))(\langle \sigma'_s, \sigma'_t, \sigma'_u, \sigma'_e, \sigma'_w \rangle)$,
 9. $\sigma'_u = \text{ame_u}(\text{sup}(\sigma_u))(\langle \sigma'_s, \sigma'_t, \sigma'_o, \sigma'_e, \sigma'_w \rangle)$,
 10. $\sigma'_e = \text{ame_e}(\text{tea}(\sigma_e))(\langle \sigma'_s, \sigma'_t, \sigma'_o, \sigma'_u, \sigma'_w \rangle)$,
 11. $\sigma'_w = \text{ame_w}(\text{wrk}(\sigma_w))(\langle \sigma'_s, \sigma'_t, \sigma'_o, \sigma'_u, \sigma'_e \rangle)$ **in**
 12. **if** objective($\langle \sigma'_s, \sigma'_t, \sigma'_o, \sigma'_u, \sigma'_e, \sigma'_w \rangle$)
 13. **then** ameliorate($\langle \sigma'_s, \sigma'_t, \sigma'_o, \sigma'_u, \sigma'_e, \sigma'_w \rangle$)
 14. **else** enterprise($\langle \sigma'_s, \sigma'_t, \sigma'_o, \sigma'_u, \sigma'_e, \sigma'_w \rangle$)
- end end

[A Simple, Functional Description of Management]

A Discussion of The Formal Model

A Re-Narration

0. Σ is a further undefined and unexplained enterprise state space.
The various enterprise players view this state in their own way.
1. Six staff group operations, **str**, **tac**, **opr**, **sup**, **tea** and **wrk**, each act in the enterprise state such as conceived by respective groups to effect a resulting enterprise state such as achieved by respective groups.
2. Six staff group state amelioration functions, **ame_s**, **ame_t**, **ame_o**, **ame_u**, **ame_e** and **ame_w**, each apply to the resulting enterprise states such as achieved by respective groups to yield a result state such as achieved by that group.
3. An overall **objective** function tests whether a state summary reflects that the objectives of the enterprise has been achieved or not.

[[A Simple, Functional Description of Management](#), [A Discussion of The Formal Model](#), [A Re-Narration](#)]

4. The **enterprise** function applies to the tuple of six group-biased (i.e., ameliorated) states. Initially these may all be the same state. The result is an ameliorated state.
5. An iteration, that is, a step of enterprise activities, lines 5.–13. proceeds as follows:
6. strategic management operates
 - in its state space, $\sigma_s : \Sigma$;
 - effects a next (un-ameliorated strategic management) state σ'_s ;
 - and ameliorates this latter state in the context of all the other player's ameliorated result states.

[A Simple, Functional Description of Management, A Discussion of The Formal Model, A Re-Narration]

- 7.–11. The same actions take place, simultaneously for the other players:
tac, opr, sup, tea and wrk.
12. A test, *has objectives been met*, is made on the six ameliorated states.
13. If test is successful, then the enterprise terminates in an ameliorated state.
14. Otherwise the enterprise recurses, that is, “repeats” itself in new states.

[A Simple, Functional Description of Management, A Discussion of The Formal Model]

On The Environment &c.

- The model does not explicitly cover interaction with
 - ★ the enterprise customers, suppliers, etc.,
 - ★ the enterprise board
 - ★ and other such external domain “players”.
- Either we can
 - ★ include those “players” as an additional number of actions
 - ★ like those of **str**, **tac**,, **wrk**, each with their states,
- or we can
 - ★ think of their states and hence their state changes
 - ★ and interaction (communication) with the enterprise
 - ★ being integrated into the enterprise state.

[A Simple, Functional Description of Management, A Discussion of The Formal Model, On The Environment &c.]

- Thus the omission of the environment is not serious:
 - ★ its modelling is just a simple extension to the given model.

[A Simple, Functional Description of Management, A Discussion of The Formal Model]

On Intra-communication

- The model does not explicitly cover communication
 - ★ between different enterprise staff group members or
 - ★ between these and the environment.
- We claim now that
 - ★ these forms of communication
 - ★ are modelled by the enterprise state:
 - ◇ in each atomic action step
 - ◇ such intended communications
 - ◇ are reflected in “messages” of the resulting state
 - ◇ where these messages are, or are not handled
 - ◇ by appropriate other enterprise staff groups
 - ◇ in some next atomic step.

[A Simple, Functional Description of Management, A Discussion of The Formal Model]

On Recursive Next-state Definitions

- Above, in Items 1.–14., we gave an intuition of the enterprise operating modes.
- But we have left un-explained the non-traditional recursive definition and use of mediated states of formula lines 6.–11.
- We now explain this unconventional recursion.
- Let us consider just two such group activities:

...

$$\sigma'_\alpha = \text{ame}_\alpha(\text{alpha}(\sigma_\alpha))(\langle \sigma'_\beta, \sigma'_\gamma, \sigma'_\delta, \sigma'_\epsilon, \sigma'_\zeta \rangle)$$

$$\sigma'_\beta = \text{ame}_\beta(\text{beta}(\sigma_\beta))(\langle \sigma'_\alpha, \sigma'_\gamma, \sigma'_\delta, \sigma'_\epsilon, \sigma'_\zeta \rangle)$$

...

[**A Simple, Functional Description of Management**, **A Discussion of The Formal Model**, **On Recursive Next-state Definitions**]

- We observe that the values σ'_α and σ'_β depend on each other.
- Thus formula lines 6.–11. recursively defines six values.
- Mathematically such recursive definitions may have solutions.
- If so, then such solutions are said to be fix points of the equations.
- In conventional computer science one normally seeks what is called least fixed point solutions.
- Such demands are not necessary in the domain.
- Mathematically one can explain a process that converges towards a solution to the set of recursive equations as an iterative process.
- If some solution exists then the process converges and terminates in one atomic step.
- If it does not exist then the process does not terminate —
 - ★ the enterprise is badly managed
 - ★ and goes bankrupt !

[A Simple, Functional Description of Management, A Discussion of The Formal Model]

Summary

- We have sketched a formal model.
- It captures some aspects of enterprise management and work.
- It abstracts most of this management and work:
 - ★ there is no hint at the nature of
 - ★ and differences between
 - ★ strategic, tactic, etc., work.
- That is,
 - ★ we have neither narrate-described such work
 - ★ to a sufficiently concrete level
 - ★ nor, obviously formalised it.

A Simple, Process Description of Management

An Enterprise System

- In this model we view
 - ★ the six “kinds” of manager and worker behaviours
 - ★ as six “kinds” of processes
 - ★ centered around a shared state process.
 - ★ There are any number,
 - ◇ CARD Strldx, of strategic,
 - ◇ CARD Supldx, of supervisors,
 - ◇ CARD Tacldx, of tactic and
 - ◇ CARD Tealdx, of team leaders and
 - ◇ CARD Opeldx, of operations managers,
 - ◇ CARD Wrkldx, of workers
 - ★ CARD Nameldx expresses the **card**inality of the set of further undefined indexes in **Nameldx**.
 - ★ The staff index sets, **Strldx**, **Tacldx**, **Opeldx**, **Supldx**, **Tealdx** and **Wrkldx** are pairwise disjoint.
- The single state process operates concurrently with all the concurrently operating manager, supervisor, team leader and worker behaviours.

[A Simple, Process Description of Management]

States and The System Composition

type

Ω , $\text{Idx}\Omega = \text{Idx} \xrightarrow{m} \Omega$, **value** $\text{idx}\omega:\text{Idx}\Omega$,

Σ , **value** $\sigma:\Sigma$

value

enterprise: **Unit** \rightarrow **Unit**

enterprise() \equiv shared_state(σ) ||

|| {strateg_process(i)(idx ω (i))|i:StrIdx} || {tactic_process(i)(idx ω (i))|i:TacIdx}

|| {operat_process(i)(idx ω (i))|i:OpeIdx} || {superv_process(i)(idx ω (i))|i:SupIdx}

|| {teamld_process(i)(idx ω (i))|i:TeaIdx} || {worker_process(i)(idx ω (i))|i:WrkIdx}

- The signature of these seven functions will be given shortly.

Channels and Messages

- Staff interaction with one another is modelled by messages sent over channels.
- Staff obtains current state from and “delivers” updated states to the state process, also via channels, one for each staff process.

[A Simple, Process Description of Management, Channels and Messages]

- We postulate a linear ordering, $<$, on indexes.
- Channels are bidirectional — so there is only a need for $n \times (n - 1)$ channels to serve n staff behaviours.

type

$$\text{Idx} = \text{StrIdx} \mid \text{TacIdx} \mid \text{OpeIdx} \mid \text{SupIdx} \mid \text{TeaIdx} \mid \text{WrkIdx}$$

$$[\text{axiom} : \text{pairwise disjoint}]$$

$$[\text{StrIdx} \cap (\text{TacIdx} \cup \text{OpeIdx} \cup \text{SupIdx} \cup \text{TeaIdx} \cup \text{WrkIdx}) = \{\}]$$

$$[\text{TacIdx} \cap (\text{OpeIdx} \cup \text{SupIdx} \cup \text{TeaIdx} \cup \text{WrkIdx}) = \{\}]$$

$$[\text{OpeIdx} \cap (\text{SupIdx} \cup \text{TeaIdx} \cup \text{WrkIdx}) = \{\}]$$

$$[\text{SupIdx} \cap (\text{TeaIdx} \cup \text{WrkIdx}) = \{\}]$$

$$[\text{TeaIdx} \cap \text{WrkIdx} = \{\}]$$
channel

$$\sigma_{\text{ch}}[i|i:\text{ChIdx}]: (\text{get}_{\Sigma}|\Sigma),$$

$$\text{staff}_{\text{ch}}[i,j|i,j:\text{ChIdx} \cdot j < i]: \text{Msg}$$
type

$$\text{Msg}, \text{get}_{\Sigma}$$

[A Simple, Process Description of Management]

Process Signatures

valueshared_state: $\Sigma \rightarrow \mathbf{in, out} \sigma_ch[j:Idx] \mathbf{Unit}$

strateg_process:

 $j:StrIdx \rightarrow \Omega \rightarrow \mathbf{in, out} \sigma_ch[j], \mathbf{staff_ch}[j,i|i:Idx \cdot i < j] \mathbf{Unit}$

tactic_process:

 $j:TacIdx \rightarrow \Omega \rightarrow \mathbf{in, out} \sigma_ch[j], \mathbf{staff_ch}[j,i|i:Idx \cdot i < j] \mathbf{Unit}$

operat_process:

 $j:OpeIdx \rightarrow \Omega \rightarrow \mathbf{in, out} \sigma_ch[j], \mathbf{staff_ch}[j,i|i:Idx \cdot i < j] \mathbf{Unit}$

superv_process:

 $j:SupIdx \rightarrow \Omega \rightarrow \mathbf{in, out} \sigma_ch[j], \mathbf{staff_ch}[j,i|i:Idx \cdot i < j] \mathbf{Unit}$

teamld_process:

 $j:TeaIdx \rightarrow \Omega \rightarrow \mathbf{in, out} \sigma_ch[j], \mathbf{staff_ch}[j,i|i:Idx \cdot i < j] \mathbf{Unit}$

worker_process:

 $j:WrkIdx \rightarrow \Omega \rightarrow \mathbf{in, out} \sigma_ch[j], \mathbf{staff_ch}[j,i|i:Idx \cdot i < j] \mathbf{Unit}$

[A Simple, Process Description of Management]

The Shared State Process

- The **shared_state** process recurses around the following triplet of actions:
 - ★ waiting for a **get state** (get_Σ) message from any staff process, j ;
 - ★ forwarding the state, σ , to that staff process;
 - ★ waiting for an updated state to be returned from staff process j .

value

$\text{shared_state}: \Sigma \rightarrow \mathbf{in}, \mathbf{out} \ \sigma_ch[j:\text{Idx}] \ \mathbf{Unit}$

$\text{shared_state}(\sigma) \equiv$

$\square \ \{\mathbf{let} \ \text{msg} = \sigma_ch[j]? \ \mathbf{in}$

$\mathbf{case} \ \text{msg} \ \mathbf{of}$

$\text{req}_\Sigma \rightarrow (\sigma_ch[j]!\sigma ; \text{shared_state}(\sigma_ch[j]?)),$

$_ \rightarrow \text{shared_state}(\sigma_ch[j]?) \ \mathbf{end} \ \mathbf{end} \ | \ j:\text{Idx}\}$

- From the definition of the **enterprise** and the staff processes
- one can prove that the message, **msg**, is either the req_Σ token or a state.

[A Simple, Process Description of Management]

Staff Processes

- There are six different kinds of staff processes:
 - ★ `strateg_process`,
 - ★ `operat_process`,
 - ★ `teamId_process` and
 - ★ `tactic_process`,
 - ★ `superv_process`,
 - ★ `worker_process`.
- We define a process, `staff_process`, to generically model any of these six processes.

A Generic Staff Behaviour

- We narrate the staff behaviour:
 - ★ (0.) We can model staff members as having three “alternative” behaviours;
 - ◇ (2.–4.) doing their **own** work;
 - ◇ (5.–9.) taking an initiative to act with other staff (**i**); and
 - ◇ (10.–13.) being prompted by other staff (**i**) to react.

[A Simple, Process Description of Management, A “Staff” Process]

- ★ (0.) Each staff behaviour **selects** whether to “do own work”, to act, or to react.
- ★ (2.) Doing own work means to work on an “own state” (ω').
- ★ (3.) The result, ω'' ,
 - ◇ is “merged” into the global state
 - ◇ which is request-obtained from the shared state.

[A Simple, Process Description of Management, A “Staff” Process]

- ★ (5.) Acting means to act on the global state (σ);
- ★ (6.) by performing some “local” operations ($\mathbf{staff_act}_j(\omega, \sigma)$)
- ★ (7.) which result in
 - ◇ local and global state changes (ω', σ') and
 - ◇ the identification of another staff member from whom to request some action (\mathbf{req})
 - ◇ which then result in some new global state (σ'') and a “local” result (\mathbf{res}).
- ★ (8.) The new global state is updated “locally”
- ★ (9.) as is the local state.

[A Simple, Process Description of Management, A “Staff” Process]

- ★ (10.) Reacting means to accept a request (**req**) from some other staff member (**i**);
- ★ (11.) to then perform some “local” operation ($\text{staff_react}_j(\text{req}, \omega, \sigma)$) which result in
 - ◇ local and global state changes (ω', σ'')
 - ◇ and some result (**res**)
- ★ (12.) The new global state is updated “locally”
- ★ (13.) as is the local state.

- ★ (4., 9., 13.) The staff process iterates (by “tail-recursion”).

[A Simple, Process Description of Management, A "Staff" Process]

value

```

0. staff_process(j)( $\omega$ )  $\equiv$  let ( $\omega'$ ,wtd) = select_wtd( $\omega$ ) in
1.   case wtd of
2.     own  $\rightarrow$  let  $\omega''$  = own_work( $\omega'$ ) in
3.       ( $\sigma$ _ch[j]! $\sigma$ _updatej( $\omega''$ ,( $\sigma$ _ch[j]!get_ $\Sigma$  ;  $\sigma$ _ch[j]?)) ||
4.       staff_process(j)( $\omega''$ )) end
5.     act  $\rightarrow$  let  $\sigma$  = ( $\sigma$ _ch[j]!get_ $\Sigma$  ;  $\sigma$ _ch[j]?) in
6.       let (i,req, $\omega''$ , $\sigma'$ ) = staff_actj( $\omega'$ , $\sigma$ ) in
7.       let (res, $\sigma''$ ) = (staff_ch[j,i]!(req, $\sigma'$ ) ; staff_ch[j,i]?) in
8.       ( $\sigma$ _ch[j]! $\sigma$ _updatej(req,res, $\omega''$ , $\sigma''$ ) ||
9.       staff_process(j)( $\omega$ _updatej(req,res, $\omega''$ ))) end end end
10.    react  $\rightarrow$  let (i,req, $\sigma'$ ) =  $\prod$ {staff_ch[j,i]?|i:Idx} in
11.      let (res, $\omega''$ , $\sigma''$ ) = staff_reactj(req, $\omega'$ , $\sigma'$ ) in
12.      (staff_ch[j,i]!(res, $\sigma$ _updatej(req,res, $\omega''$ , $\sigma''$ )) ||
13.      staff_process(j)( $\omega$ _updatej(req,res, $\omega''$ ))) end end end end

```

[A Simple, Process Description of Management, A "Staff" Process]

A Diagrammatic Rendition

- Let us consider Fig. H.12 on the facing page:
 - ★ The digits of Fig. H.12 on the next page refer to the line numbers of the `staff_process` definition (Slide 905).
 - ★ The figure intends to show the trace of three processes:
 - ◇ the `shared state` process,
 - ◇ `staff` process `j`
 - serving in the own work work mode
 - as well as in the active work mode,
 - and
 - ◇ `staff` process `i` (serving in the reactive work mode).

[A Simple, Process Description of Management, A "Staff" Process, A Diagrammatic Rendition]

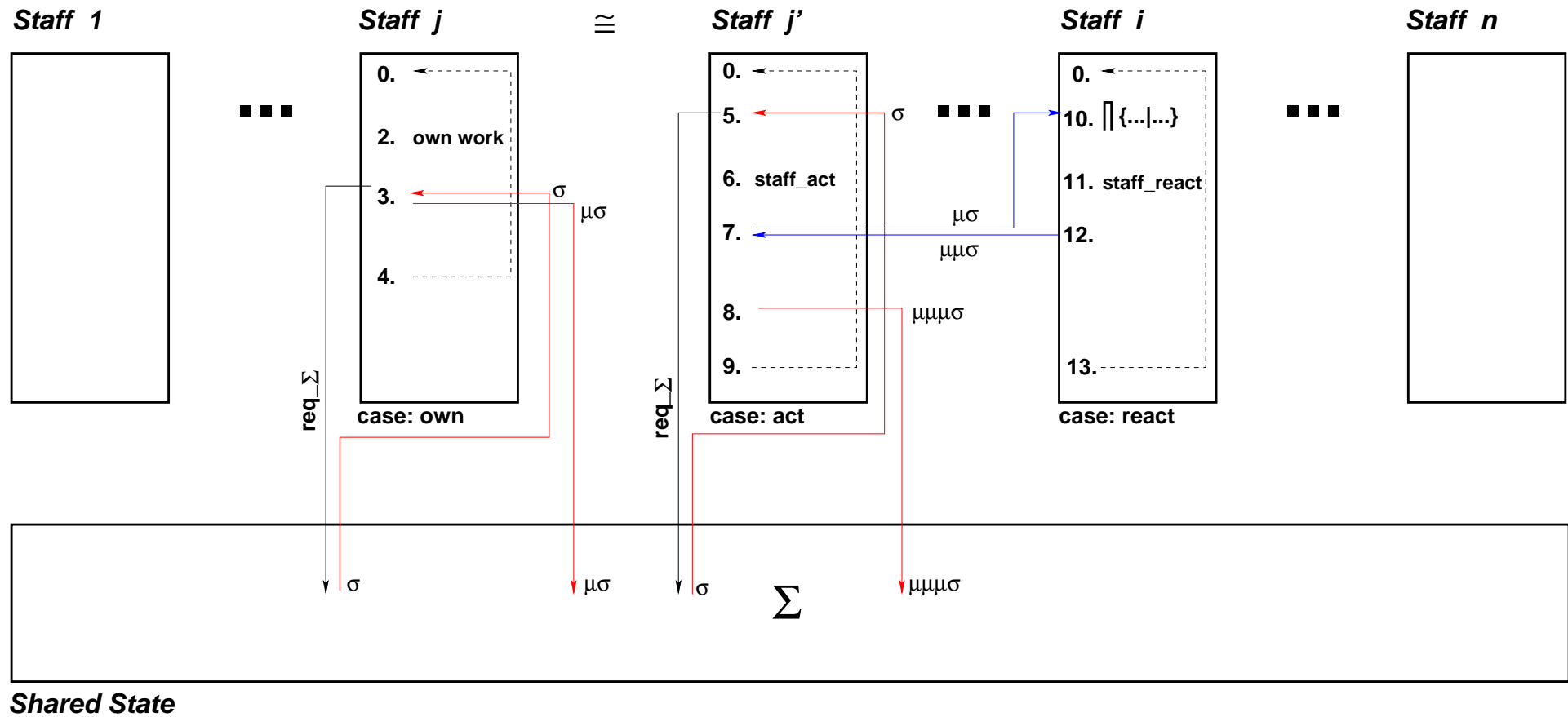


Figure H.12: Own and 'action'-'reaction' process traces: $\mu\sigma = \sigma'$, $\mu\mu\sigma = \sigma''$, $\mu\mu\mu\sigma = \sigma'''$

[A Simple, Process Description of Management, A “Staff” Process]

Auxiliary Functions

- A number of auxiliary functions have been used.
 - ★ The `select_wtd` (“select what to do”) clause
 - type**
WhatToDo = own | act | react
 - value**
 $\text{select_wtd}: \Omega \rightarrow \Omega \times \text{WhatToDo}$
 - ◇ internally, non-deterministically
 - chooses one of the three **WhatToDo** alternatives
 - as also shown in Lines 2., 6. and 11 Slide 905.
 - ◇ The choice is based on the local state (ω).
 - The outcome of the choice, whether **own**, **act** or **react**,
 - reflects, we could claim, a priority need for either of these alternatives,
 - or just reflects human vagary!
 - ◇ The choice is recorded in an update local state (ω').

[A Simple, Process Description of Management, A “Staff” Process, Auxiliary Functions]

★ The `own_work` function

value `own_work`: $\Omega \rightarrow \Omega$

- ◇ is “own” as it applies only to the local state (ω).
- ◇ The modelling idea is the following:
 - The ‘own work’, by any staff member,
 - is modelled as taking place, not on the global state,
 - but the result is eventually (see Line 3. Slide 905 and, next, the σ_update_j) into the global state.
 - This models, we claim, that staff works locally on “copies” of the global state.

[A Simple, Process Description of Management, A “Staff” Process, Auxiliary Functions]

★ The σ_update_j function

value $\sigma_update_j: \Omega \times \Sigma \rightarrow \Sigma$

- ◇ models the “merging” of a local state (ω) into the global state (σ).
 - We do not describe this ‘merging’.
 - But such a description should be made,
 - if need be, separately, for each case of the
 - **own_work** (Line 3.), **staff_act** (Line 8.) and **staff_react** (Line 12., Slide 905)
 - alternatives.

[A Simple, Process Description of Management, A "Staff" Process, Auxiliary Functions]

★ The `staff_act` function

type

Request = Req₁ | Req₂ | ... | Req_m

value

`staff_act`: $\Omega \times \Sigma \rightarrow \text{Idx} \times \text{Request} \times \Omega \times \Sigma$

- ◇ applies to both the local and the global state.
- ◇ The purpose of the `staff_act` query is (line 6., Slide 905)
 - to determine with which other staff (`i:Idx`) the current staff (`j`) need interact
 - and for what purposes (`req`),
 - The `staff_act` query updates both the local and the global states (ω'' , σ').
- ◇ The `staff_act` query is assumed to take very little time.

[A Simple, Process Description of Management, A "Staff" Process, Auxiliary Functions]

★ The ω_{update} function (Lines 9. and 13., Slide 905)

value ω_{update} : Request \times Result $\times \Omega \rightarrow \Omega$

- ◇ updates the local state with the action request and result (**req, res**) being made to and yielded by staff i .
- ◇ We do not describe this 'merging'.
 - But such a description should be made,
 - if need be, separately, for each case of the
 - **staff_act** (Line 9.) and **staff_react** (Line 13., Slide 905)
 - alternatives.

[A Simple, Process Description of Management, A “Staff” Process, Auxiliary Functions]

★ The staff_react_j function is, for each j , a (usually large) set of functions:

type

Result = Res₁ | Res₂ | ... | Res_m

value

$\text{staff_react}_j: \text{Request} \times \Omega \times \Sigma \rightarrow \text{Result} \times \Omega \times \Sigma$

$\text{staff_react}_j(\text{req}, \omega, \sigma) \equiv$

(**case** req **of** $r_1 \rightarrow \text{op}_{j_1}(r_1), r_2 \rightarrow \text{op}_{j_2}(r_2), \dots, r_m \rightarrow \text{op}_{j_m}(r_m)$ **end**)(ω, σ)

$\text{op}_{j_i}: \text{Req}_i \rightarrow \Omega \times \Sigma \rightarrow \text{Res}_i \times \Omega \times \Sigma$

◇ Each of these operations are assumed to be of the kind:

- prepare for this operation to be carried out
- when doing ‘own work’.

◇ We shall comment on these operations below [Slides 917–928].

[A Simple, Process Description of Management, A “Staff” Process]

Assumptions

- A number of assumptions have been made in expressing the staff process:
 - ★ The time-duration of
 - ◇ the inter-process communications and
 - ◇ the ω and σ updates are zero;and the time-durations of
 - ◇ $\text{staff_act}_j(\omega, \sigma)$ and
 - ◇ $\text{staff_react}_j(\text{req}, \omega, \sigma)$
 - ◇ operations are “near”-zero.

[A Simple, Process Description of Management, A “Staff” Process, Assumptions]

- ★ These two functions
 - ◇ do not cause any interaction with neither the shared state nor other staff processes.
- ★ The time-duration of the `own_work(ω)` operation may be any length of time.
- ★ (The real work is done
 - ◇ during the local state change `own_work(ω)` operation
 - ◇ and the result of this work is eventually “fed back into the global state”!)

[A Simple, Process Description of Management, A "Staff" Process, Assumptions]

- When offering to act or react the designated partner staff behaviour will accept the offer and within a reasonably realistic time interval.
- With these assumptions fulfilled
 - ★ it is acceptable to model
 - ★ global state changes as non-interleaved.

[A Simple Process Description of Management]

Management Operations

- So, which are the functions $\text{op}_{j_i}(\omega, \sigma)$?
 - ★ As is obvious from Slides 319–343 there are zillions of management operations.
 - ★ They are loosely suggested on Slides 319–343.
- Thus we shall not further define these here.
- But some comments are in order.

[A Simple Process Description of Management, Management Operations]

● Focus on Management ●

- We focus on management rather than on “workers”.
- Operations by workers, say in a railway transport system, deal with:
 - ★ selling and cancelling tickets,
 - ★ starting, driving and stopping trains,
 - ★ setting signals,
 - ★ laying down new and maintaining rails, etcetera.
- Management operations are of two kinds:
 - ★ Own, preparatory ‘work’ — that may take hours and days; and
 - ★ dispensing or receiving order — that may take “down to” fractions of minutes.
- This view of ‘mgt. operations’ partly justifies our staff model.

[A Simple Process Description of Management, Management Operations]

● Own and Global States ●

- Workers spend most, and managers some of their time on ‘own work’ —
 - ★ and then apply the operations of that ‘own work’ to local states.
- Worker local states
 - ★ are usually very clearly delineated “copies” of the global states
 - ★ somehow made inaccessible to other staff while subject to ‘own work’.
- Manager local states are usually not so clearly delineated.
- ‘Own work’ is “reflected back into”, that is, updates, the global state (cf. Line 3. Slide 905).

[A Simple Process Description of Management, Management Operations, Own and Global States]

● State Classification ●

- We have presented a notion of local ($\omega : \Omega$) global states ($\sigma : \Sigma$) without really saying much about these.
- We may also have given the impression that these states were inert,
 - ★ that is, changed only when operated upon by the staff.
- We now redress this impression,
 - ★ that is, we now make it clear
 - ★ that states may have several components,
 - ★ and that some individual state components may be
 - ◇ (i) inert dynamic,
 - ◇ (ii) active dynamic comprising:
 - (ii.1) autonomous active dynamic,
 - (ii.2) biddable active dynamic and
 - ◇ (iii) reactive dynamic.
 - (ii.3) programmable active dynamic
 - and

[A Simple Process Description of Management, Management Operations]

Transport System States

- In a transport system these are some of the state components.

★ *Transport Net State Changes* ★

- (i) the transport net which changes state due to
 - ★ (i.1) wear and tear of the net,
 - ★ (i.2) setting and resetting of signals,
 - ★ (i.3) insertion and removal of links,
 - ★ etcetera.

[A Simple Process Description of Management, Management Operations, Transport System States]

★ *Net Traffic State Changes* ★

- (ii) the net traffic which changes state due to
 - ★ (ii.1) vehicles entering, moving around, and leaving the net,
 - ★ (ii.2) vehicles accidents, road/bridge/tunnel breakdowns,
 - ★ (ii.3) the state changes of the underlying net,
 - ★ etcetera.

[A Simple Process Description of Management, Management Operations, Transport System States]

★ *Managed State Changes* ★

- (iii) management state changes due to
 - ★ (iii.1) changed transport vehicle timetables being inserted,
 - ★ (iii.2) changed toll road fee schedules being enacted,
 - ★ (iii.3) changed speed limits,
 - ★ (iii.4) changed signalling rules,
 - ★ (iii.5) changed resource (incl. public vehicle) allocation,
 - ★ etcetera.

[A Simple Process Description of Management]

The Overall Managed System

- One can speak of an *overall managed system* which we consider as consisting of
 - ★ all staff,
 - ★ all explicitly shared state components, and
 - ★ the more implicitly observable state components of the environment.
- Figure H.13 on the facing page tries to conceptually “picture” such an overall managed system

[A Simple Process Description of Management, Management Operations, The Overall Managed System]

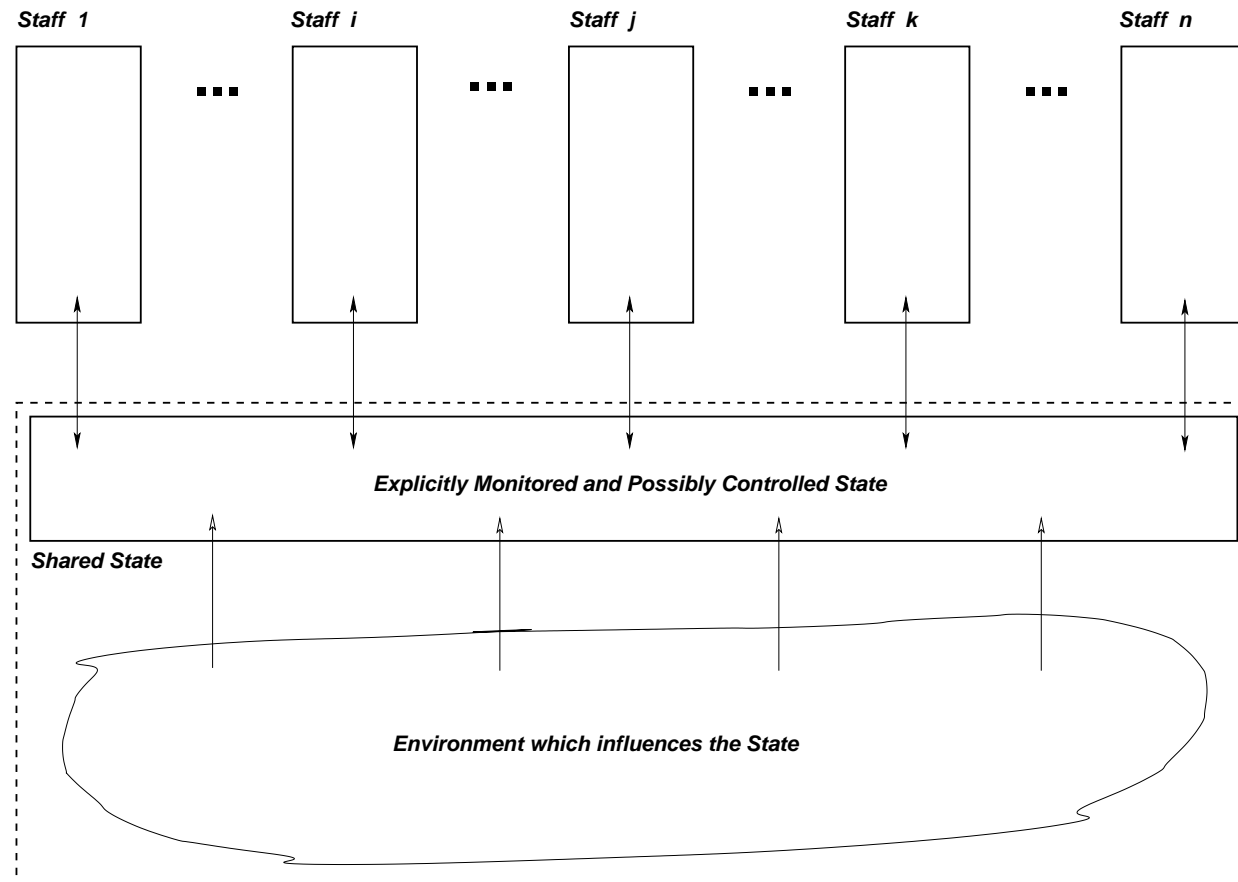


Figure H.13: Staff and explicit (shared) and implicit states

[A Simple Process Description of Management, Management Operations, The Overall Managed System]

- In a domain model we try, to our best, to describe
 - ★ the n staff behaviours: executive, strategic, tactic and operations managers, supervisors, team leader and workers, and
 - ★ the various explicitly known state components of the domain,
 - ◇ whether only observable (that is: monitorable)
 - ◇ or also controllable (that is: generable).
 - ★ In a domain model we may have,
 - ◇ through the modelling of the state components,
 - ◇ to thus implicitly model environment state concepts.

[A Simple Process Description of Management]

Discussion**Management Operations**

- We leave it to the student to draw the necessary conclusions:
 - ★ (i) only some state components are of concern to management,
 - ★ (ii) not all such state components can be controlled by management,
 - ★ (iii) to model the all the system state changes is thus not a concern of modelling management (and organisation), and
 - ★ (iv) to model all management operations is not feasible.
- We have, in other sections of this domain model development, i.e., Slides 691–1101,
 - ★ while covering other domain facets,
 - ★ illustrated many management operations.

[A Simple Process Description of Management, Discussion]

Managed States

-
-
-
-

Discussion of First Two Management Models

Generic Management Models

- The first two management models, Slides 881–928,
 - ★ the functional, Slides 881–895, and
 - ★ the process descriptions, Slides 896–928,really did not show any management aspects of transportation systems.
- The two models can be claimed to be generic.
 - ★ As such they apply to a wide variety of domain management.

[Discussion of First Two Management Models, Generic Management Models]

- The two models can also be claimed to be one another's "inverse".
 - ★ The process description can be claimed to "implement" the functional description;
 - ★ each "step" of the staff processes can be claimed to correspond to an "iteration" of the "solving" of the recursive equations of Lines 6.–11., Slide 886.
 - ★ We leave it as a research challenge for the student
 - ◇ "clean-up" the two formal definitions, that is, express them in a formalism,
 - ◇ such that a theorem expressing that the process model "implements" the functional, doubly recursive model.
- Such a 'clean-up' might possibly involve rewriting the functional, doubly recursive model into an imperative tail-recursive model.

[Discussion of First Two Management Models]

Management as Scripts

- It was said, above, that
 - ★ management is manifested in “zillions” of actions,
 - ★ some occurring concurrently,
 - ★ some occurring in strict sequence,
 - ★ etcetera.
- But nothing was then said, above, of the order, if any, of these actions.
- Some actions cannot be meaningfully applied before others have been applied.
 - ★ (i) The management decision to remove a specific link, ℓ ,
 - ◇ must occur some time after a (thus previous) management decision
 - ◇ to insert that specific link, ℓ .

[Discussion of First Two Management Models, Management as Scripts]

- ★ (ii) The management decision to construct a (new) train timetable
 - ◇ must not occur before reasonable completion dates for
 - the construction of the underlying rail net,
 - the purchase of required rolling stock, and
 - the hiring of required net and train operation staffhave all been established.
- ★ (iii) the management go-ahead for the start of train traffic according to a new timetable
 - ◇ must not occur before the completion of
 - the construction of a train (staff) rostering plan,
 - the construction of a train maintenance plan, and
 - the rail net construction,
 - ◇ etcetera.

[Discussion of First Two Management Models, Management as Scripts]

- One — not so extreme — interpretation of the above is
 - ★ that we cannot meaningfully describe specific concurrent and sequential
 - ★ sets of management actions
 - ★ but must basically, in most cases of systems,
 - ★ accept any such patterns of actions.
- Another — slightly less extreme — interpretation of the above is
 - ★ that we can in some cases describe
 - ◇ what we shall later define as
 - ◇ a family of management scripts
- Let that suffice for the time being.

Transport Enterprise Organisation

- Transportation is “home” to many different kinds of enterprises.
- Each of these enterprises is concerned with relatively distinct and reasonably non-overlapping issues:
 - ★ road (bridge, etc.) building, \mathcal{E}_{rd_b} ,
 - ★ road (etc.) maintenance, \mathcal{E}_{rd_m} ,
 - ★ road & car traffic signaling, \mathcal{E}_{rd_s} ,
 - ★ bus services (i), \mathcal{E}_{bs_i} ,
 - ★ fire brigade, \mathcal{E}_{fi} ,
 - ★ police, \mathcal{E}_{po} ,
 - ★ rail building, \mathcal{E}_{rl_b} ,
 - ★ rail maintenance, \mathcal{E}_{rl_m} ,
 - ★ rail & train signaling, \mathcal{E}_{rl_s} ,
 - ★ train services (j), \mathcal{E}_{tp_j} ,
 - ★ map making (k) \mathcal{E}_{mm_k} ,
 - ★ etcetera.
- But they “share” the transportation net.

[Transport Enterprise Organisation]

Transport Organisations

- Each kind of transportation enterprise, say \mathcal{E}_i ,
 - ★ covers a subset, say $\mathcal{NET}_{\mathcal{E}_i}$, of the net, that is,
 - ★ not necessarily the entire net.
- For two or more $i, j, i \neq j$, it may be that $\mathcal{NET}_{\mathcal{E}_i} \cap \mathcal{NET}_{\mathcal{E}_j} \neq \{\}$.
- Each transportation enterprise
 - ★ has its own distinct staff, that is, sets of
 - ◇ strategies mgrs., $STR_{\mathcal{E}_i}$, ◇ operations mgrs., $OPS_{\mathcal{E}_i}$, ◇ team leaders, $TLD_{\mathcal{E}_i}$, and
 - ◇ tactics mgrs., $TAC_{\mathcal{E}_i}$, ◇ supervisors, $SUP_{\mathcal{E}_i}$, ◇ workers, $WRK_{\mathcal{E}_i}$,
- For some managers, supervisors, team leaders and workers
 - ★ the areas of the net for which they are responsible
 - ★ are proper, disjoint subsets of the net.

[Transport Enterprise Organisation]

Analysis

- To describe the transportation domain one has to model
 - ★ for each transportation enterprise, \mathcal{E}_i , separate subsets of the net
 - ◇ $\mathcal{NET}_{\mathcal{E}_{rd_b}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{rd_m}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{rd_s}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{bp_i}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{rl_b}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{rl_m}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{rl_s}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{tp_t}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{fi}}$,
 - ◇ $\mathcal{NET}_{\mathcal{E}_{po}}$,
 - ◇ and $\mathcal{NET}_{\mathcal{E}_{mm_k}}$; and,
 - ★ for each such transportation enterprise, one has to model a number of separate enterprise structures:
 - ◇ \mathcal{E}_{rd_b} ,
 - ◇ \mathcal{E}_{rd_m} ,
 - ◇ \mathcal{E}_{rd_s} ,
 - ◇ \mathcal{E}_{bp_i} ,
 - ◇ \mathcal{E}_{rl_b} ,
 - ◇ \mathcal{E}_{rl_m} ,
 - ◇ \mathcal{E}_{rl_s} ,
 - ◇ \mathcal{E}_{tp_t} ,
 - ◇ \mathcal{E}_{fi} ,
 - ◇ \mathcal{E}_{po} ,
 - ◇ and \mathcal{E}_{mm_k} .

[Transport Enterprise Organisation]

Modelling Concepts

Net Kinds

- In order to model the various nets, $\mathcal{NET}_{\varepsilon_i}$,
 - ★ given that we have a base model \mathbf{N} ,
 - ★ we introduce a notion of ‘net kind’:
 - ◇ one for each of the nets $\mathcal{NET}_{\varepsilon_{rd_b}}$, $\mathcal{NET}_{\varepsilon_{rd_m}}$, $\mathcal{NET}_{\varepsilon_{rd_s}}$, $\mathcal{NET}_{\varepsilon_{bp_i}}$, $\mathcal{NET}_{\varepsilon_{rl_b}}$, $\mathcal{NET}_{\varepsilon_{rl_m}}$, $\mathcal{NET}_{\varepsilon_{rl_s}}$, $\mathcal{NET}_{\varepsilon_{tp_t}}$, $\mathcal{NET}_{\varepsilon_{fi}}$, $\mathcal{NET}_{\varepsilon_{po}}$, $\mathcal{NET}_{\varepsilon_{mm_k}}$, etcetera.
 - ★ A ‘net kind’, $\mathbf{k}:\mathbf{K}$, is like a type designator

type

$\mathbf{K} = \text{SimP}|\text{BusK}|\text{TrainK}|\text{MapMK}$

$\text{SimK} == \text{road}_b|\text{road}_m|\text{road}_s|\text{rail}_b|\text{rail}_m|\text{rail}_s|\text{fireb}|\text{police}$

$\text{BusK} == \text{bus}_p1|\text{bus}_p2|\dots|\text{bus}_pm$

$\text{TrainK} == \text{train}_p1|\text{train}_p2|\dots|\text{train}_pn$

$\text{MapMK} == \text{map}_m1|\text{map}_m2|\dots|\text{map}_mo$

[Transport Enterprise Organisation, Modelling Concepts, Net Kinds]

- To each hub and link in a net we then associate zero, one or more net kinds.

- ★ We then postulate an observer function:

value

$$\text{obs}_K: (H|L) \rightarrow \mathbf{K\text{-set}}$$

- ★ Given a net kind we can then “extract” the net of hubs and links “carrying” that net kind:

value

$$\text{xtr}_N: N \times K \rightarrow N$$

- ◇ To guarantee that the extracted net is indeed a (well-formed) net
- ◇ we must make sure that any assignment of net kinds to hubs and links
- ◇ results in well-formed net kind nets:

value

$$\text{wf}_{NK}: N \rightarrow \mathbf{Bool}$$

- ★ To express wf_{NK} we define a function

value

$$\text{xtr}_{Ks}: N \rightarrow \mathbf{K\text{-set}}$$

- ◇ which collects all net kinds from all hubs and links of the net.

[Transport Enterprise Organisation, Modelling Concepts, Net Kinds]

value

$$\text{xtr_Ks}(hs, ls) \equiv \\ \cup \{ \text{obs_Ks}(h) \mid h:H \cdot h \in hs \} \cup \cup \{ \text{obs_Ks}(l) \mid l:L \cdot l \in ls \}$$

$$\text{wf_N}'(hs, ls) \equiv \\ \forall k:K \cdot k \in \text{xtr_Ks}(hs, ls) \Rightarrow \\ \text{wf_N}(\{h \mid h:H \cdot h \in hs \wedge k \in \text{obs_Ks}(h)\}, \{l \mid l:L \cdot l \in ls \wedge k \in \text{obs_Ks}(l)\})$$

- The predicate $\text{wf_N}'$ extends the predicate wf_N
- which corresponds to the satisfaction of all the axioms given on Slides 775–782.

$$\text{xtr_N}((hs, ls), k) \equiv \\ (\{h \mid h:H \cdot h \in hs \wedge k \in \text{obs_Ks}(h)\}, \{l \mid l:L \cdot l \in ls \wedge k \in \text{obs_Ks}(l)\}) \\ \text{pre } k \in \text{xtr_Ks}(hs, ls)$$

[Transport Enterprise Organisation, Modelling Concepts]

Enterprise Kinds

- In order to model the various enterprises, \mathcal{E}_i ,
 - ★ given that we have a base model for business staff, \mathbf{Sldx} ,
 - ★ we introduce a notion of ‘enterprise kind’:
 - ◇ one for each of the enterprise kind: \mathcal{E}_{rd_b} , \mathcal{E}_{rd_m} , \mathcal{E}_{rd_s} , \mathcal{E}_{bp_i} , \mathcal{E}_{rl_b} , \mathcal{E}_{rl_m} , \mathcal{E}_{rl_s} , \mathcal{E}_{tp_t} , \mathcal{E}_{fi} , \mathcal{E}_{po} , \mathcal{E}_{mm_k} , etcetera.
 - ★ A ‘enterprise kind’, $\mathbf{b:B}$, is like a type designator

type

$\mathbf{E} = \mathbf{SimE} | \mathbf{BusE} | \mathbf{TrainE} | \mathbf{MapMakE}$

$\mathbf{SimE} == \mathbf{road_b} | \mathbf{road_m} | \mathbf{road_s} | \mathbf{rail_b} | \mathbf{rail_m} | \mathbf{rail_s} | \mathbf{fireb} | \mathbf{police}$

$\mathbf{BusE} == \mathbf{bus_p1} | \mathbf{bus_p2} | \dots | \mathbf{bus_pm}$

$\mathbf{TrainE} == \mathbf{train_p1} | \mathbf{train_p2} | \dots | \mathbf{train_pn}$

$\mathbf{MapMakE} == \mathbf{map_m1} | \mathbf{map_m2} | \dots | \mathbf{map_mo}$

[Transport Enterprise Organisation, Modelling Concepts]

Staff Kinds

- To each staff we then associate an enterprise kind.

value

obs_B: SIdx \rightarrow B

- ★ We may further have to impose that interaction between staff, viz.:

staff_ch[i,j]

- ★ be subject to an “internal business constraint”:

obs_B(i) = obs_B(j)

[Transport Enterprise Organisation, Modelling Concepts]

Staff Kind Constraints

● Narrative ●

- The staff, $Staff_{\mathcal{E}_i}$, of each transportation enterprise, \mathcal{E}_i ,
- can be roughly categorised into:
 - ★ strategic managers, $STR_{\mathcal{E}_i}$,
 - ★ tactics managers, $TAC_{\mathcal{E}_i}$,
 - ★ operations managers, $OPS_{\mathcal{E}_i}$,
 - ★ supervisors, $SUP_{\mathcal{E}_i}$,
 - ★ team leaders, $TLD_{\mathcal{E}_i}$, and
 - ★ workers $WRK_{\mathcal{E}_i}$,as already mentioned (Slide 935).
- And each of these can be further sub-categorised.

[Transport Enterprise Organisation, Modelling Concepts, Staff Kind Constraints]

● Formalisation ●

- We suggest the “beginnings” of a formalisation.

type

StaffK == stra|tac|ope|sup|tld|wrk

value

obs_StaffK: SIdx → StaffK

- A more realistic model, for a given enterprise, would provide a far more detailed categorisation.

★ Typically there might be several “layers”

◇ of strategic and management. ◇ of tactic and ◇ of operations

★ Similarly a model might detail different kinds of

◇ supervisor, ◇ team leader and ◇ worker staff.

[Transport Enterprise Organisation, Modelling Concepts, Staff Kind Constraints]

● Hierarchical Staff Structures ●

- We refer to Fig. 2.2, Slide 354.
-
-
-

[Transport Enterprise Organisation, Modelling Concepts, Staff Kind Constraints]

• Matrix Staff Structures •

-
-
-
-

[Transport Enterprise Organisation, Modelling Concepts]

Net and Enterprise Kind Constraints

● Narrative ●

- A link (or a hub) is either a road or a rail link (hub).
 - ★ If a link is a road link then the two hubs that it connects are road hubs.
 - ★ If a hub is a road hub then all the links emanating from the hub are road links.
 - ★ If a link is a rail link then the two hubs that it connects are rail hubs.
 - ★ If a hub is a rail hub then all the links emanating from the hub are rail links.
- In consequence we may speak of disjoint road nets and rail nets.
- The enterprises associated with a road [rail] net must be road [rail] related (building, maintenance, signaling, bus [train] services, police, etc.).

[Transport Enterprise Organisation, Modelling Concepts, Net and Enterprise Kind Constraints]

• Formalisation •

type

NetK: road|rail|...

value

obs_NetK: (H|L) \rightarrow NetK

xtr_L: LI \rightarrow L-set \rightarrow L, xtr_H: HI \rightarrow H-set \rightarrow H

xtr_L(li)(ls) $\equiv \exists !l:L \cdot l \in ls \wedge \text{obs_LI}(l)=li$

xtr_H(hi)(hs) $\equiv \exists !h:H \cdot h \in hs \wedge \text{obs_HI}(h)=hi$

wf_N'': N \rightarrow **Bool**

wf_N''(hs,ls) \equiv

$\forall h:H \cdot h \in hs \Rightarrow$

$\forall li:LI \cdot li \in \text{obs_LIs}(h) \Rightarrow \text{obs_NetK}(xtr_L(li)(ls))=\text{obs_NetK}(h) \wedge$

$\forall l:L \cdot l \in ls \Rightarrow$

$\forall hi:HI \cdot hi \in \text{obs_HIs}(l) \Rightarrow \text{obs_NetK}(xtr_H(hi)(hs))=\text{obs_NetK}(l)$

- The predicate wf_N'' extends
- the predicate wf_N' given earlier (Slide 939).

[Transport Enterprise Organisation, Modelling Concepts, Net and Enterprise Kind Constraints, Formalisation]

value

netk: $N \rightarrow \text{NetK-set}$

$\text{netk}(hs,ls) \equiv \{\text{obs_NetK}(h) \mid h:H \cdot h \in hs\} \cup \{\text{obs_NetK}(l) \mid l:L \cdot l \in ls\}$

$\text{road_k}:\text{NetK-set} = \{\text{road_b}, \text{road_m}, \text{road_s}, \text{bus_p1}, \text{bus_p2}, \dots, \text{bus_pm}, \text{fireb}, \text{police}\},$

$\text{rail_k}:\text{NetK-set} = \{\text{rail_b}, \text{rail_m}, \text{rail_s}, \text{train_p1}, \text{train_p2}, \dots, \text{train_pn}, \text{fireb}, \text{police}\}$

$\text{wf_N}''': N \rightarrow \text{Bool}$

$\text{wf_N}'''(hs,ls) \equiv$

let nks = netk(hs,ls) **in**

case nks **of**

$\{\text{road}\} \rightarrow \text{xtr_Ks}(hs,ls) \subseteq \text{road_k}, \{\text{rail}\} \rightarrow \text{xtr_Ks}(hs,ls) \subseteq \text{rail_k}, _ \rightarrow \text{false}$

end end

- The predicate $\text{wf_N}'''$ extends $\text{wf_N}''$ given earlier (Slide 947).

[Transport Enterprise Organisation]

Net Signaling

- In the previous section on support technology we did not describe who or which “ordered” the change of hub states.
- We could claim that this might very well be a task for management.
 - ★ (We here look aside from such possibilities
 - ◇ that the domain being modelled has some further support technology which advises individual hub controllers as when to change signals and then into which states.
 - ◇ We are interested in finding an example of a management & organisation facet — and the upcoming one might do!)

[Transport Enterprise Organisation, Net Signaling]

Narrative

- So we think of a ‘net hub state management’ for a given net.
- That management is divided into a number of ‘sub-net hub state managements’ where the sub-nets form a partitioning of the whole net.
- For each sub-net management there are two kinds management interfaces:
 - ★ one to the overall hub state management, and
 - ★ one for each of interfacing sub-nets.
- What these managements do, what traffic state information they monitor, etcetera, you can yourself “dream” up.
- Our point is this:
 - ★ We have identified a management organisation.

[Transport Enterprise Organisation, Net Signaling]

Formalisation

type

$$\text{HIsLIs} = \text{HI-set} \times \text{LI-set}$$
$$\text{MgtNet}' = \text{HIsLIs} \times \mathbb{N}$$
$$\text{MgtNet} = \{ | \text{mgtnet} : \text{MgtNet}' \cdot \text{wf_MgtNet}(\text{mgtnet}) | \}$$
$$\text{Partitioning}' = \text{HIsLIs-set} \times \mathbb{N}$$
$$\text{Partitioning} = \{ | \text{partitioning} : \text{Partitioning}' \cdot \text{wf_Partitioning}(\text{partitioning}) | \}$$

value

$$\text{wf_MgtNet} : \text{MgtNet}' \rightarrow \mathbf{Bool}$$
$$\text{wf_MgtNet}((\text{his}, \text{lis}), n) \equiv$$

[The his component contains all the hub ids. of links identified in lis]

$$\text{wf_Partitioning} : \text{Partitioning}' \rightarrow \mathbf{Bool}$$
$$\text{wf_Partitioning}(\text{hisliss}, n) \equiv$$
$$\forall (\text{his}, \text{lis}) : \text{HIsLIs} \cdot (\text{his}, \text{lis}) \in \text{hisliss} \Rightarrow \text{wf_MgtNet}((\text{his}, \text{lis}), n) \wedge$$

[no sub-net overlap and together they "span" n]

Discussion

-
-
-
-

Dines Bjørner: 8th DRAFT: October 14, 2008

[Discussion]

-
-
-
-

[Discussion]

-
-
-
-

End of Support: Domain Engineering: Mgt. and Org.
