

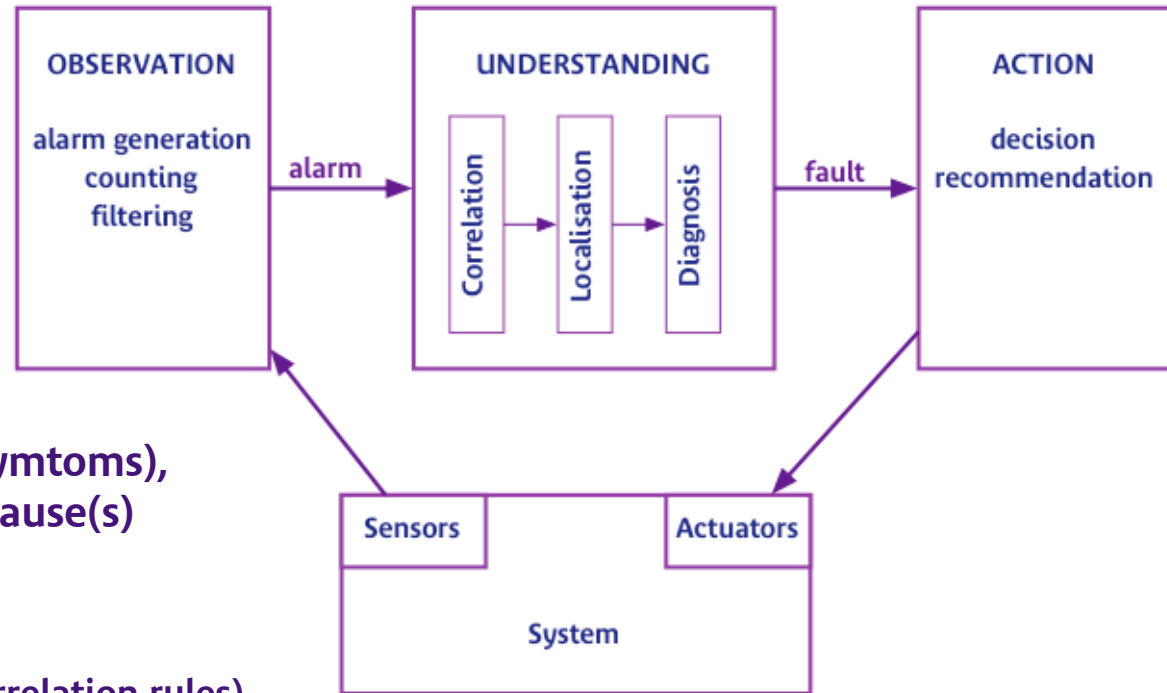


Diagnosis for component-based system

Christophe Dousson — Irène Grosclaude
FTR&D/DTL/TIC/ABC

Le présent document contient des informations qui sont la propriété de France Télécom R&D. L'acceptation de ce document par son destinataire implique, de la part de ce dernier, la reconnaissance du caractère confidentiel de son contenu et l'engagement de n'en faire aucune reproduction, aucune transmission à des tiers, aucune divulgation et aucune utilisation commerciale sans l'accord préalable écrit de France Télécom R&D

The diagnosing problem



- ➔ From the observations (the symptoms), to find the possible primary cause(s)
- ➔ Two families of approaches:
 - expert-based (deduction/correlation rules)
 - model-based (nominal or fault models)
- ➔ Two distinct objectives
 - maintenance (off-line, accuracy)
 - supervision/monitoring (real-time, QoS)

Magda: a distributed-diagnosis experiment



➔ Knowledge modeling:

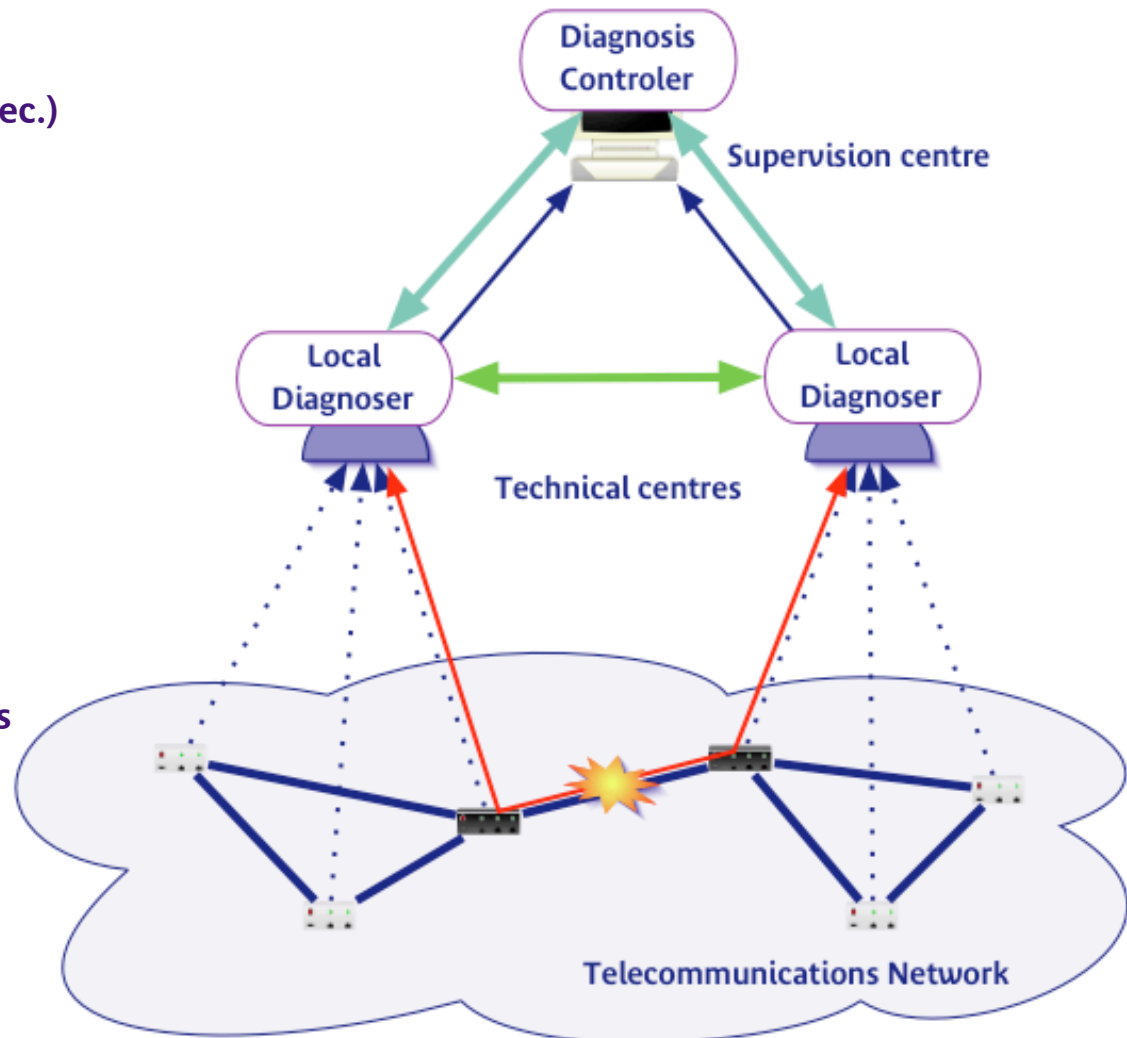
- elementary component (ITU rec.)
- component behavior (alarm propagation model)
- network topology
- logical connections

➔ Symptoms:

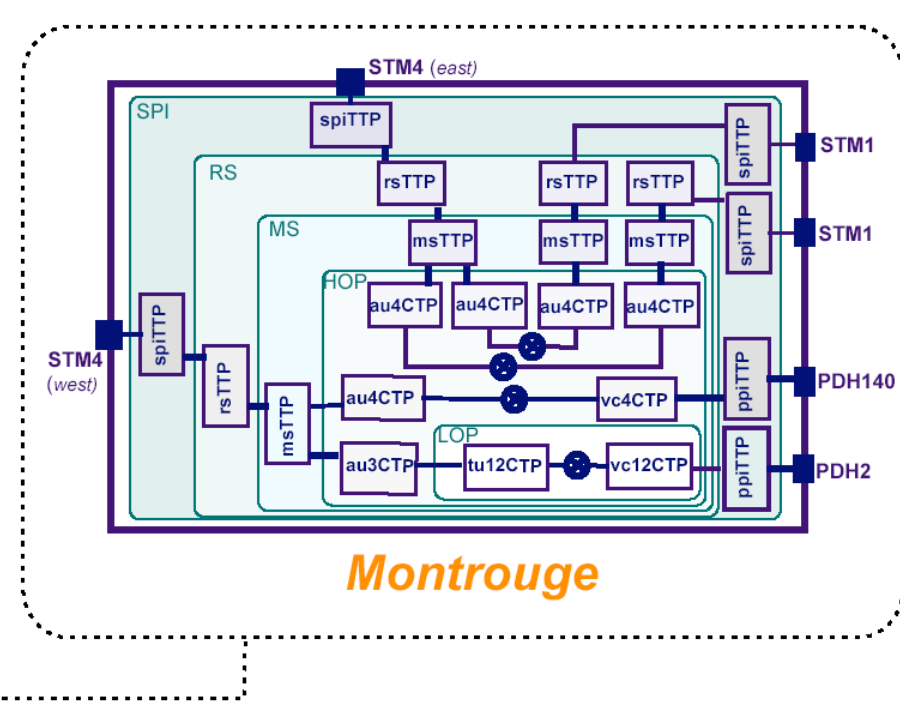
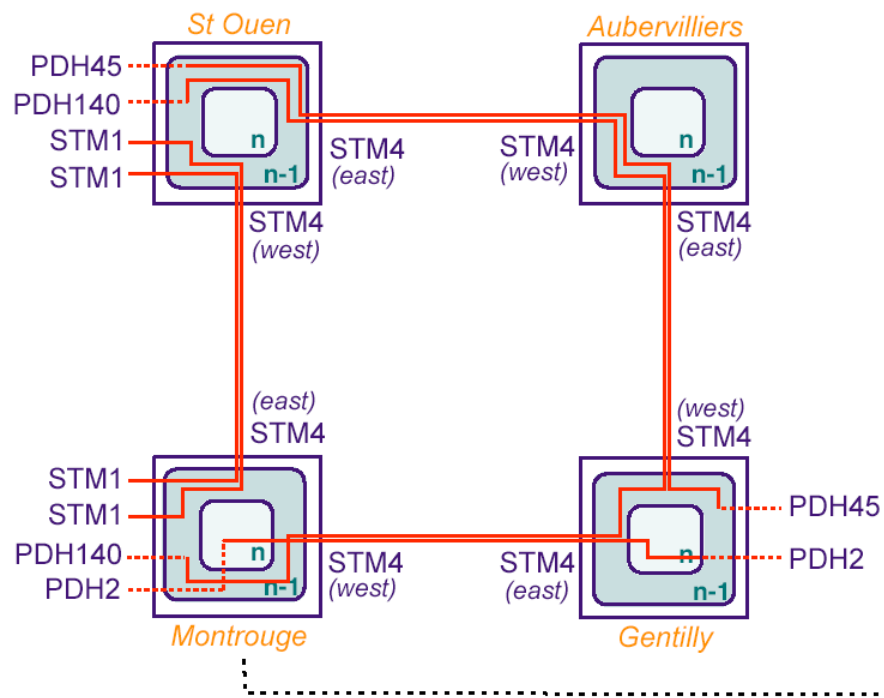
- alarms

➔ Diagnosis:

- histories of components states that explain the alarms
- causalities between them



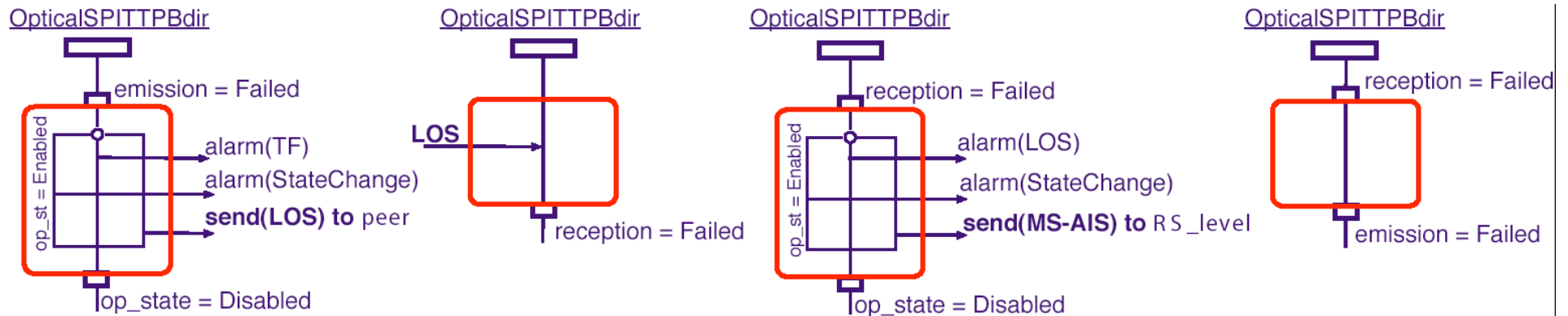
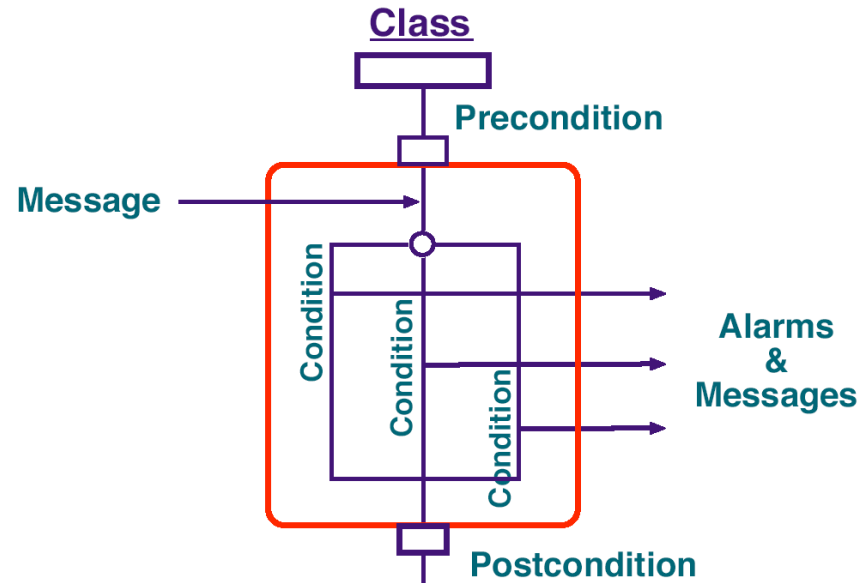
Magda modeling: component topology/connectivity



Magda modeling: component behaviour



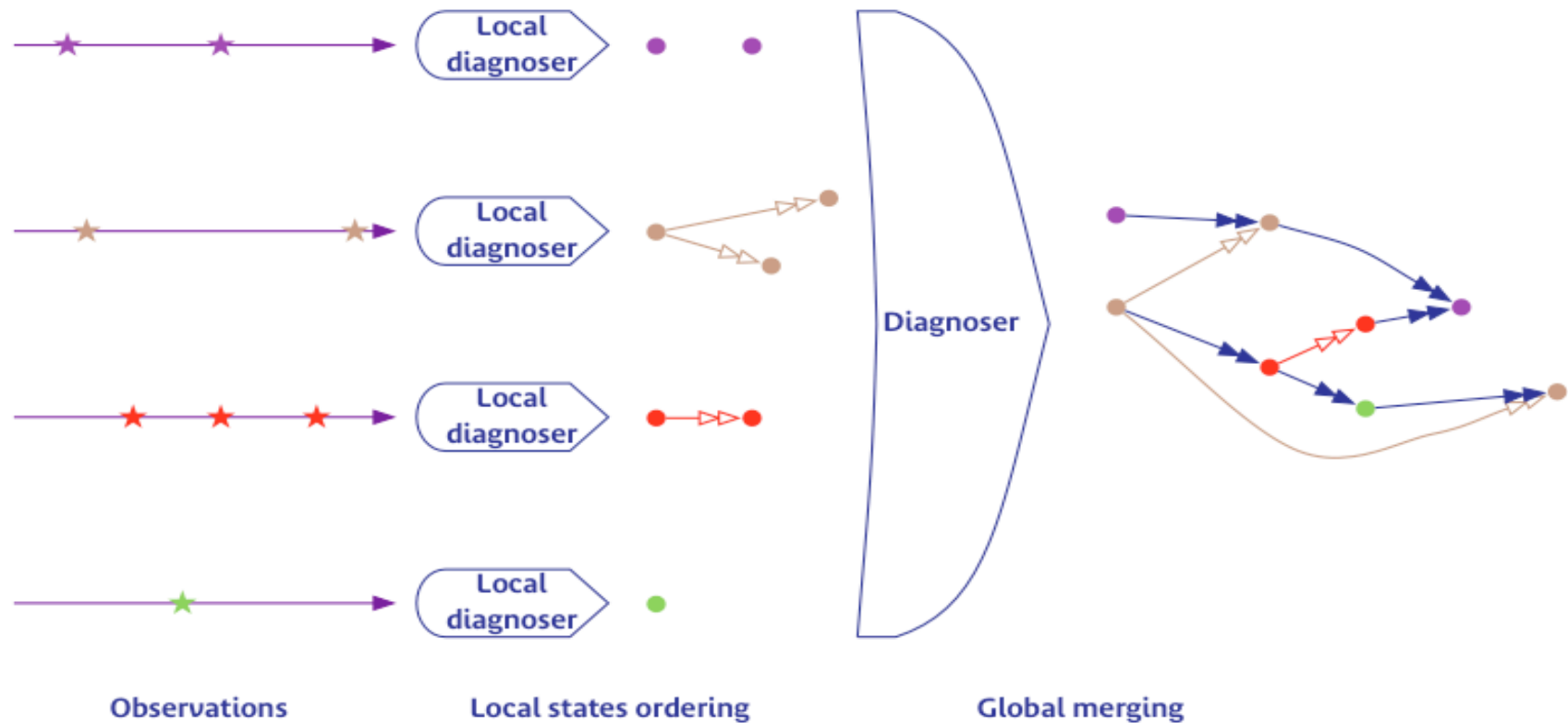
- ➔ Tiles: local alarm propagation model
 - one or zero input message
 - pre/post-conditions
 - produces messages to other components
 - produces alarms (symptoms)
- ➔ Diagnosis: connecting the tiles
 - with respect of the topology



Distributed Magda Diagnoser



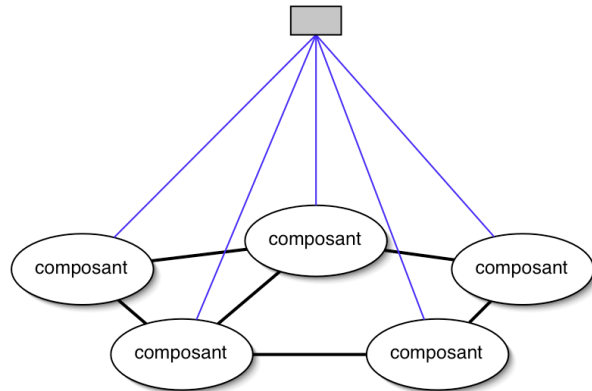
- ➔ Observations orders states with causal relationships
 - Local ordering then global merging (ambiguity / observation losses are allowed)
 - Root of the graph = primary causes



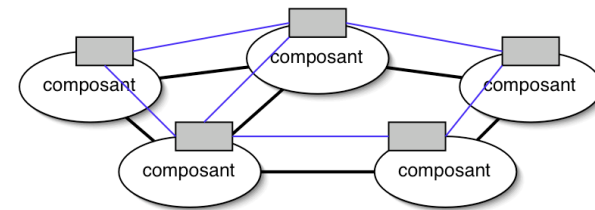
Distributed Diagnoser Architectures



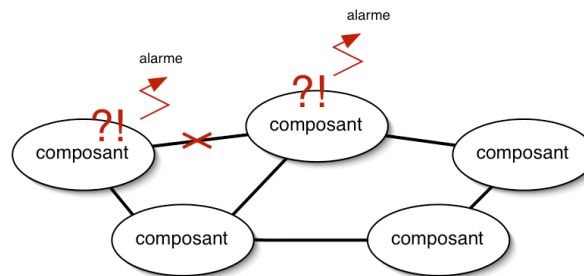
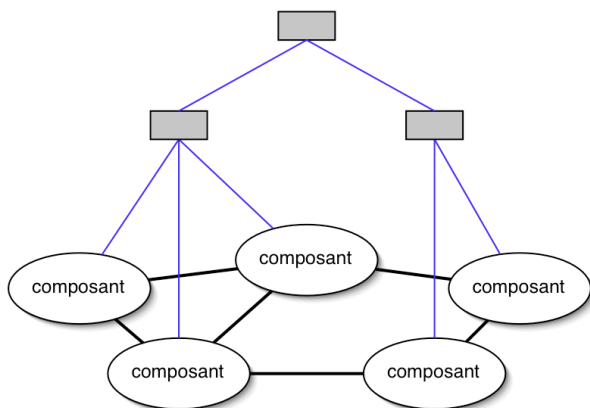
(a) Diagnostic centralisé



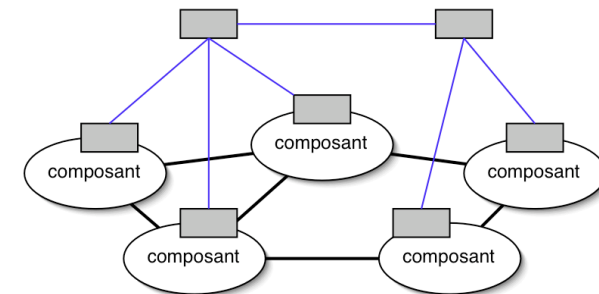
(c) Diagnostic distribué



(b) Diagnostic hiérarchique



(d) Diagnostic hiérarchique distribué





Fractal diagnosis problem

➔ Objective

- Platform administration (must work also with OTS Fractalized-component)
- THIS IS NOT component debugging

➔ Model: nominal behavior

- life-cycle automata (local behavior)
- additional knowledge
 - software failure mode and effect analysis

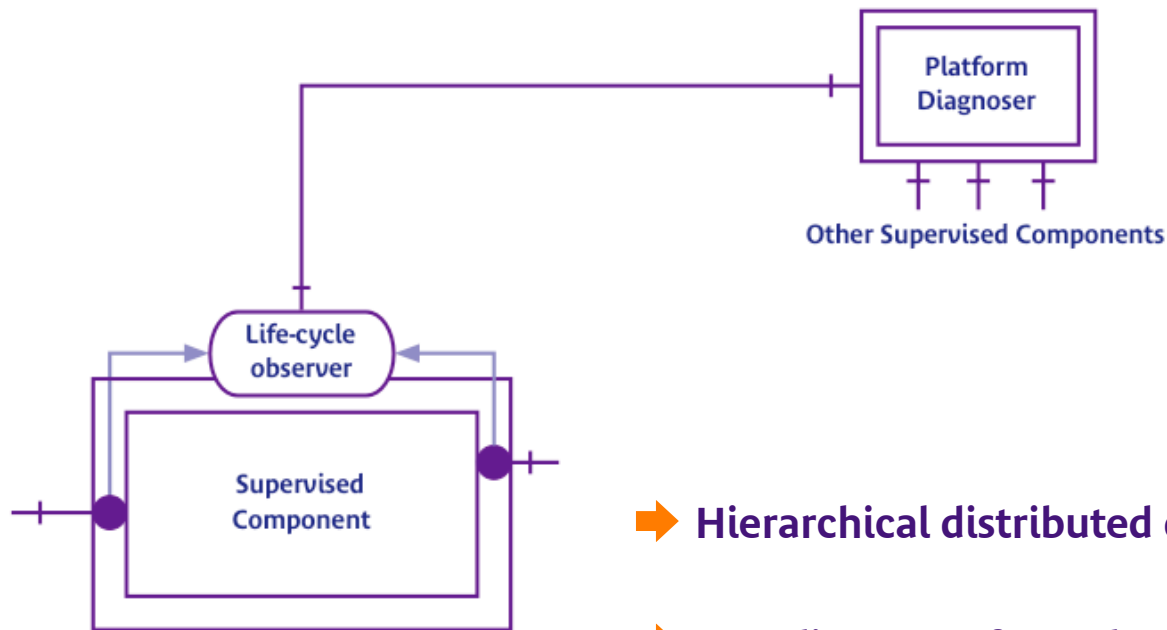
➔ Faults: discrepancy from nominal behavior

- Locked Component
- Bad component use (wrong inputs)
- Bad response (wrong outputs, exception)

➔ Observations

- exchanged messages between components
- user interaction
- additional system observations ? (cpu, ...)

Component diagnosis architecture



- ➔ Hierarchical distributed diagnosis
- ➔ One diagnoser for a platform
 - the diagnoser is a component
- ➔ One observer (the life-cycle controller) for each component
 - observations could be made through the interceptors



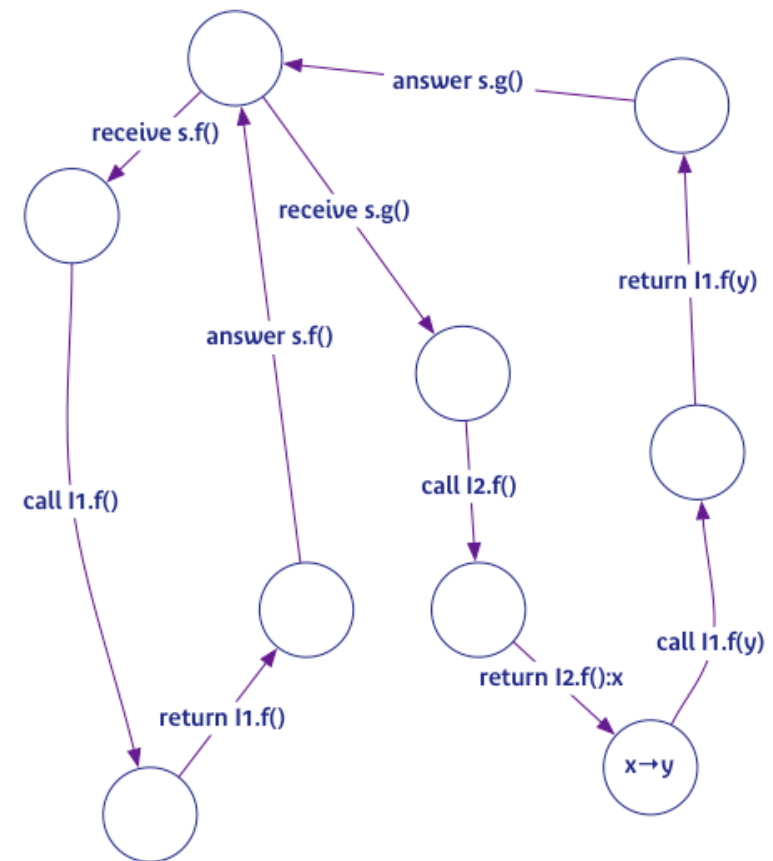
Component observer

- ➔ **Life-cycle observer**
 - implemented by the way of interceptors

- ➔ **Local diagnoser**
 - Build local histories

- ➔ **Life-cycle based-model**
 - transition:
 - method calls and returns
 - exceptions
 - state:
 - component state
 - faulty state? (will ask a global diagnosis ?)

 - additionnal knowledge on parameter flows





Additional knowledge for the diagnosis

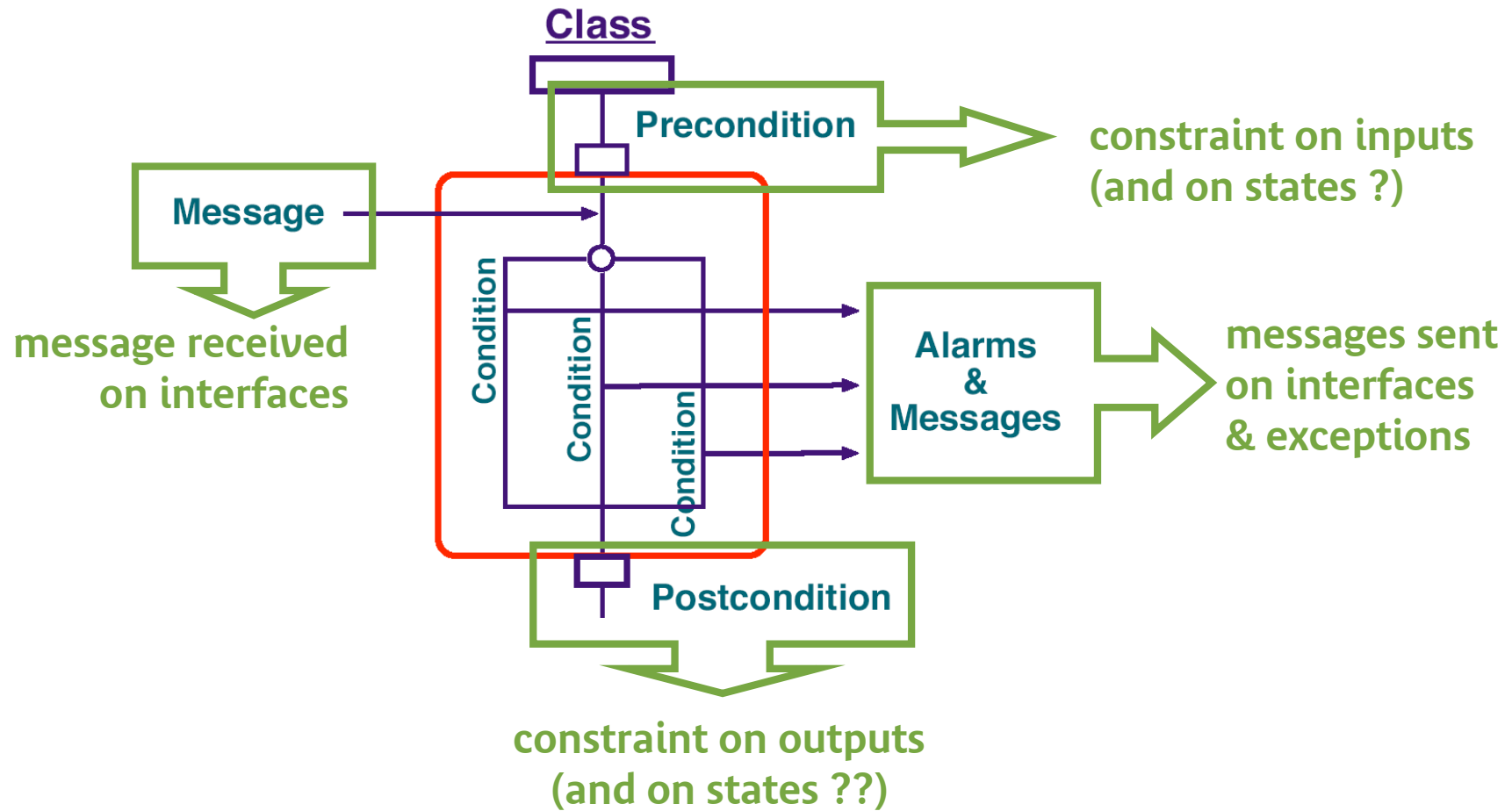
- ➔ **Must be easy to collect**
 - based on knowledge already integrated in the dvpt process

- ➔ **Can not be complete**
 - Use of ambiguity/observations losses process capabilities of the diagnoser

- ➔ **Proposed formalism: rules between the inputs and the outputs**
 - e.g. `c1.concat(string s1, int n): string`
 - if `s1` is empty or `n=0`, returned string is empty
 - if `s1` is unbounded or `n < 0`, throw a exception
 - ...

 - e.g. `c2. concat(string s1, int n): string`
 - if `s1` is unbounded or `n < 0`, returned string is empty
 - ...

Comparison with Magda tiles



Just do it...



- ➔ Find a (real) component-based application
- ➔ Exhibit some faulty states (?)
- ➔ Define some in/out properties
- ➔ Adapt the diagnosis algorithm

Prospectives



- ➔ **Diagnosability of the system**
 - Notion of n-diagnosability
 - Identification of critical states

- ➔ **Minimum observers capabilities**
 - Ensure no ambiguity between critical states

- ➔ **Minimize the overhead of the diagnoser**
 - Optimisation of interceptors usage