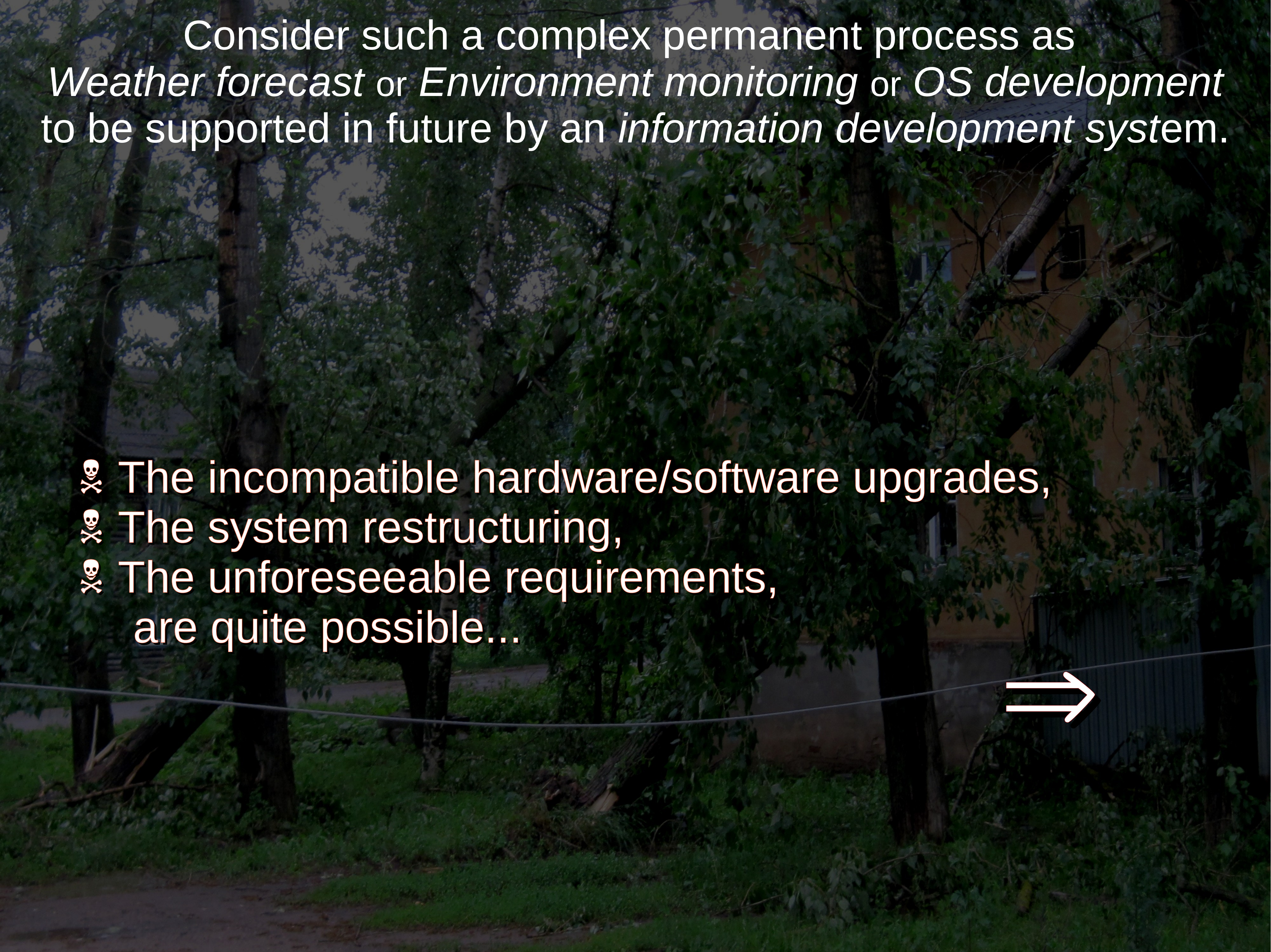
On the way to new information systems theory

Sergej Znamenskij

Ailamazyan Program system institute of RAS Pereslavl Zalesskij

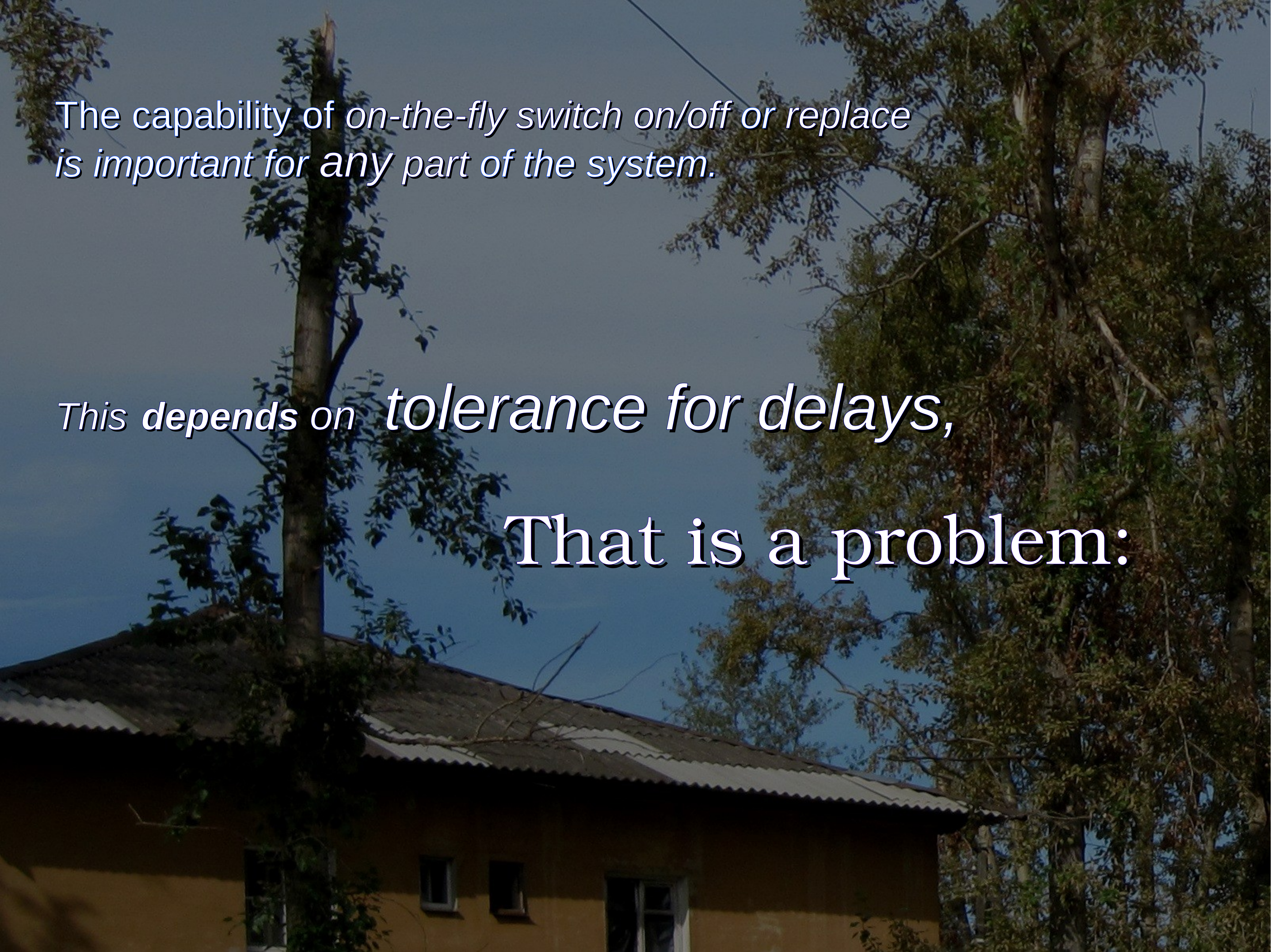
October 2012



Consider such a complex permanent process as
Weather forecast or *Environment monitoring* or *OS development*
to be supported in future by an *information development* system.

☠ The incompatible hardware/software upgrades,
☠ The system restructuring,
☠ The unforeseeable requirements,
are quite possible...

⇒



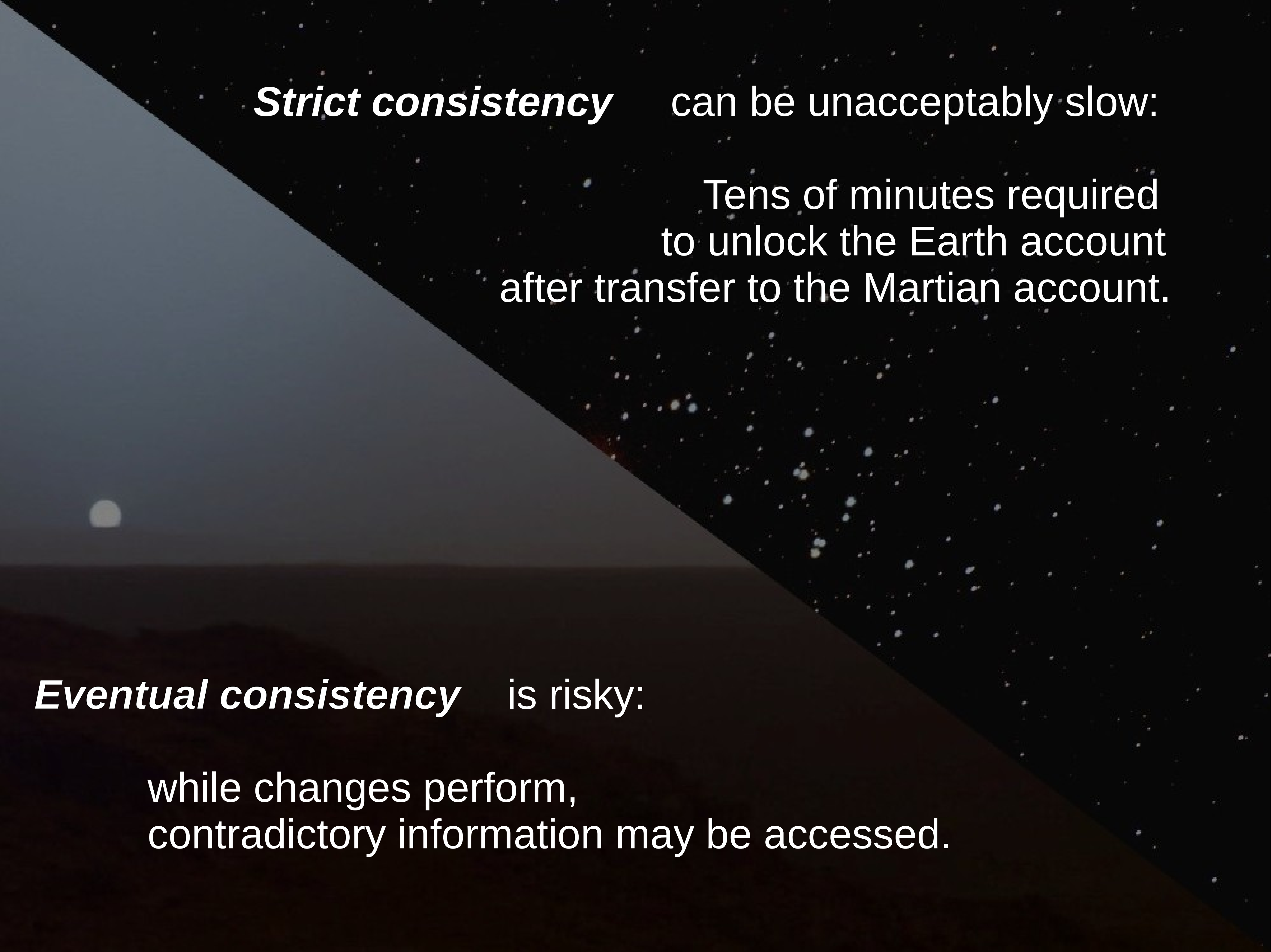
The capability of *on-the-fly switch on/off or replace is important for* any *part of the system.*

This ***depends*** *on* *tolerance for delays,*

That is a problem:

Strict consistency can be unacceptably slow:

Tens of minutes required
to unlock the Earth account
after transfer to the Martian account.

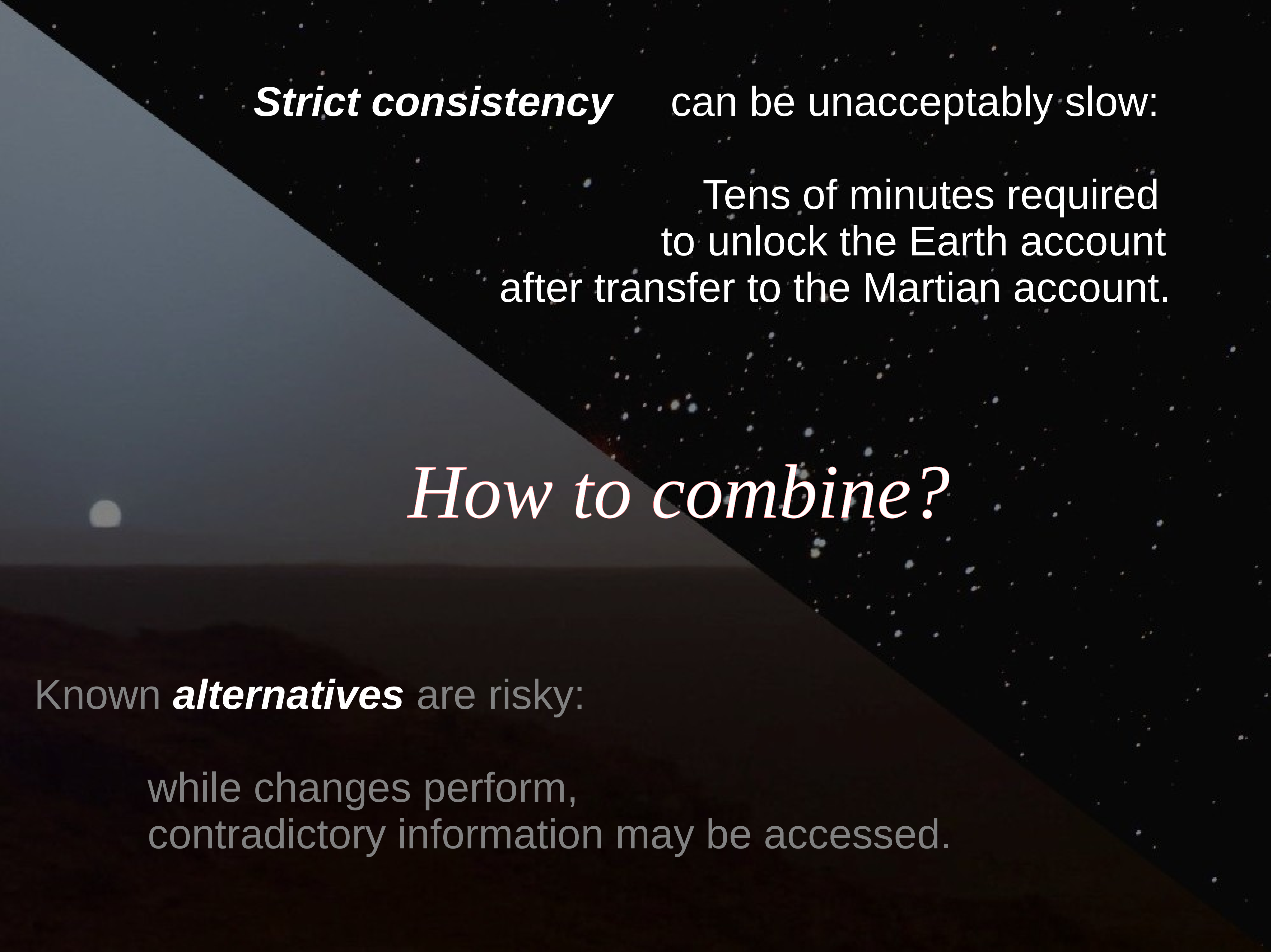


Strict consistency can be unacceptably slow:

Tens of minutes required
to unlock the Earth account
after transfer to the Martian account.

Eventual consistency is risky:

while changes perform,
contradictory information may be accessed.



Strict consistency can be unacceptably slow:

Tens of minutes required
to unlock the Earth account
after transfer to the Martian account.

How to combine?

Known ***alternatives*** are risky:

while changes perform,
contradictory information may be accessed.

Retrospective paradigm:

⚠ No *current state* in *real time*.

Retrospective paradigm:

⚠ No *current state* in *real time*.

⚒ Known request is proceeded for time *t* = *Moment of Truth*.

Retrospective paradigm:

⚠ No *current state* in *real time*.

⚒ Known request is proceeded for time *t* = *Moment of Truth*.

➫ Response is obtained in <0.1 second and marked with *t*.

Retrospective paradigm.

⚠ No *current state* in *real time*.

⚒ Known request is proceeded for time *t* = *Moment of Truth*.

➫ Response is obtained in <0.1 second and marked with *t*.

✌ All data is fairy consistent for each *t*

Retrospective paradigm:

⚠ No *current state* in *real time*.

⚒ Known request is proceeded for time *t* = *Moment of Truth*.

➩ Response is obtained in <0.1 second and marked with *t*.

✌ All data is fairy consistent for each *t*

Tolerance for delays and resilience:

Updates frequency degrade while system is overloaded or link broken, but restores immediately.

Requests are never lost, as opposite to any *resistant real-time system.*

Retrospective paradigm:

⚠ No *current state* in *real time*.

⚒ Known request is proceeded for time *t* = *Moment of Truth*.

➯ Response is obtained in <0.1 second and marked with *t*.

✌ All data is fairy consistent for each *t*

A way to fast responsive, fully consistent, fault tolerant system?

Retrospective paradigm:

Architecture?

The organizational structure of a system or component, their relationships, and the principles and guidelines governing their design and evolution over time.

IEEE 610.12

A way to fast responsive, fully consistent, fault tolerant system?

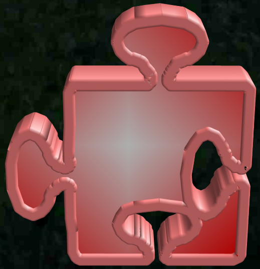
What a view on system architecture to be the best suited to a long-life responsible distributed service

e.g. Weather forecast, OS development, … ?

What a view on system architecture to be the best suited to a long-life responsible distributed service *e.g. Weather forecast, OS development, … ?*

Ontological — system as an object of activity:

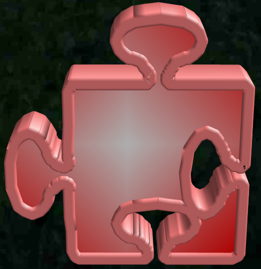
❀ divided to smaller isolated components;
⚒ come planned, designed, implemented and replaced;
⚖ serve to meet requirements of creation time;
♣ based on *ER, BPM, QoS, OO, SOA, …*



What a view on system architecture to be the best suited to a long-life responsible distributed service *e.g. Weather forecast, OS development, … ?*

Epistemological — system as an environment for the sustainable development, based on growing understanding:
❀ divided to interconnected environments for smaller activities;
⚒ comes discovered, forgotten and successfully rediscovered;
⚖ adaptive to changing environment;
♣ based on ...

Ontological — system as an object of activity:
❀ divided to smaller isolated components;
⚒ come planned, designed, implemented and replaced;
⚖ serve to meet requirements of creation time;
♣ based on *ER, BPM, QoS, OO, SOA, …*



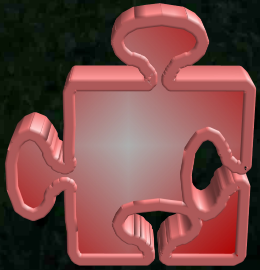
What a view on *system architecture* *be the best suited* for a long-life responsible distributed service *e.g. Weather forecast, OS development, … ?*

Epistemological — system as an environment for the sustainable development, based on growing understanding:

❀ divided to interconnected environments for smaller activities;
⚒ comes discovered, forgotten and successfully rediscovered;
⚖ adaptive to changing environment;
♣ based on ... ***something else***...

Ontological — system as an object of activity:

❀ divided to smaller isolated components;
⚒ come planned, designed, implemented and replaced;
⚖ serve to meet requirements of creation time;
♣ based on *ER, BPM, QoS, OO, SOA, …*



A view on *system architecture?*

Sustainable Activities hierarchy (never forget/replace)

� *Research and development*
� *Resource Control (Networking,Scheduling)*
� *Total Quality Management*

Replaceable components and Modules

� *Hardware*
� *Algorithms*
� *Digital libraries*
� *Observational Data*

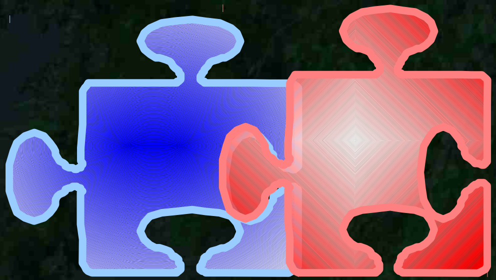
A view on *system architecture?*

Sustainable Activities hierarchy (never forget/replace)

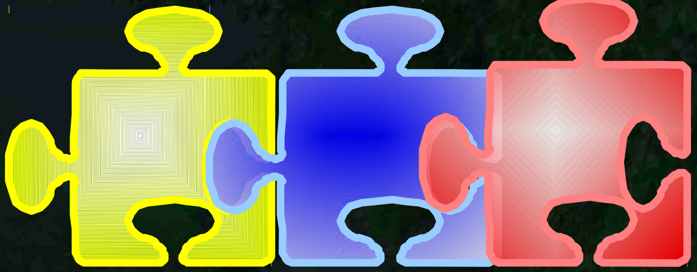
- *Research and development*
- *Resource Control (Networking,Scheduling)*
- *Total Quality Management*

Replaceable components and Modules

- *Hardware*
- *Algorithms*
- *Digital libraries*
- *Observational Data*

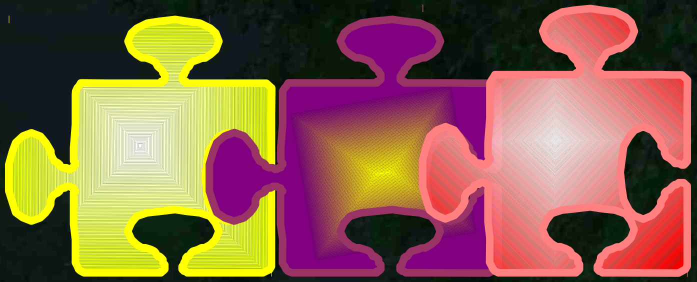


A view on *system architecture?*

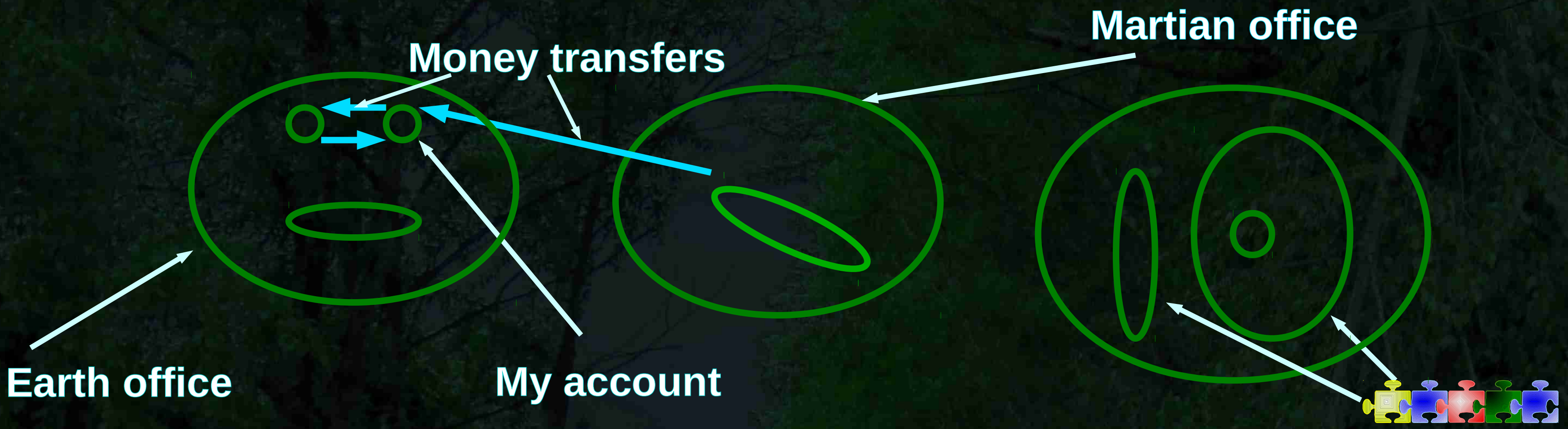


A view on *system architecture?*

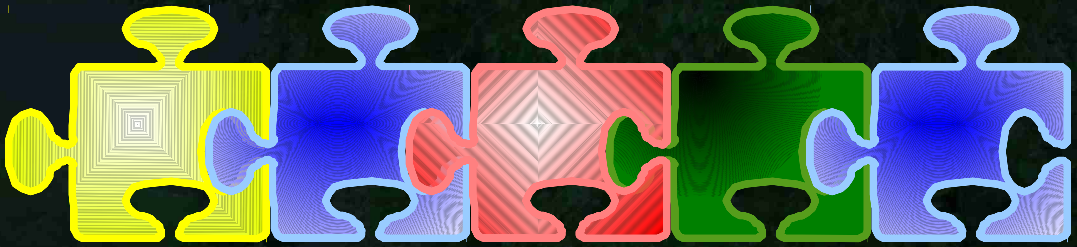
to keep consistency over delays...



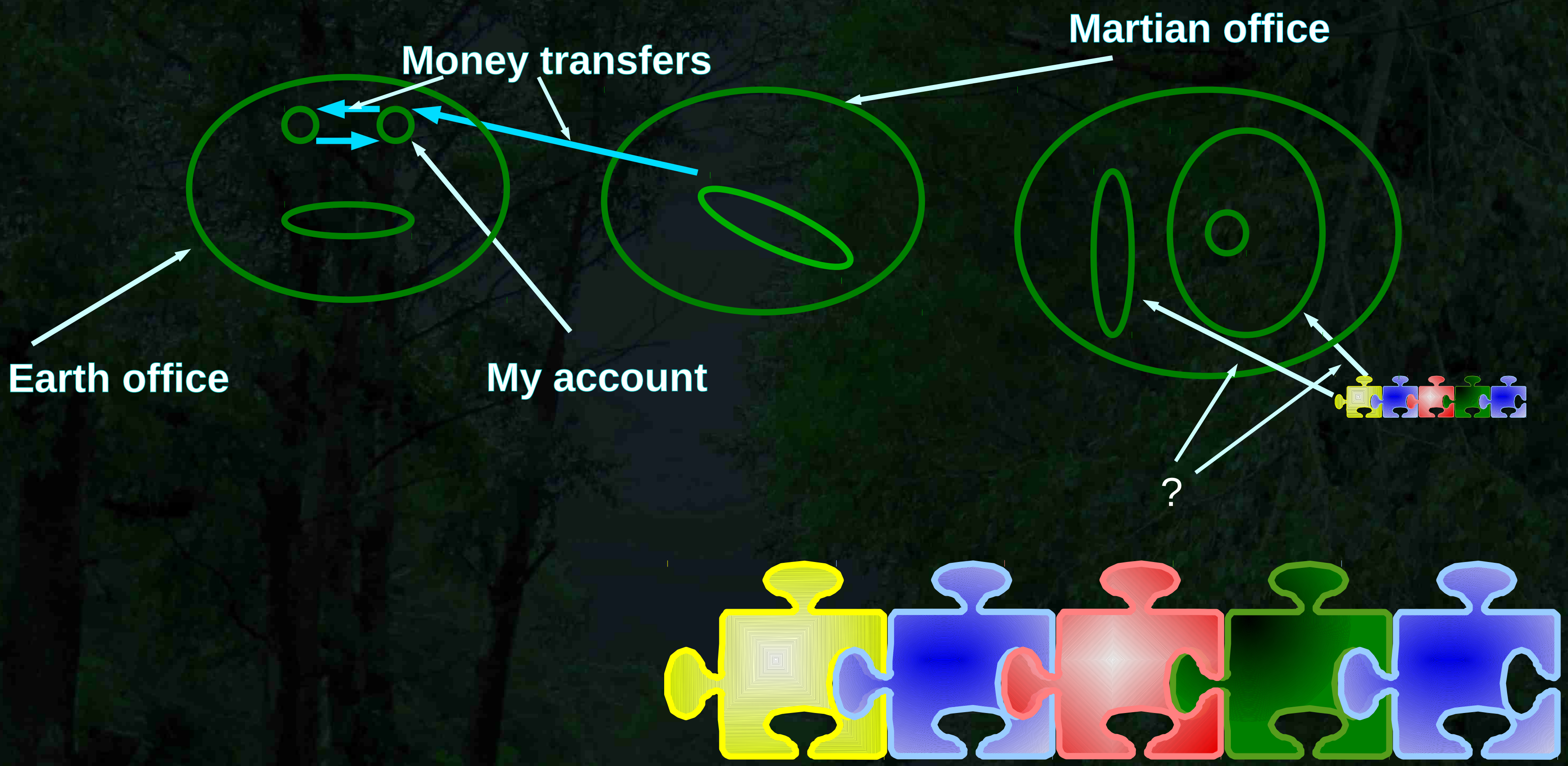
A view on *system architecture?*



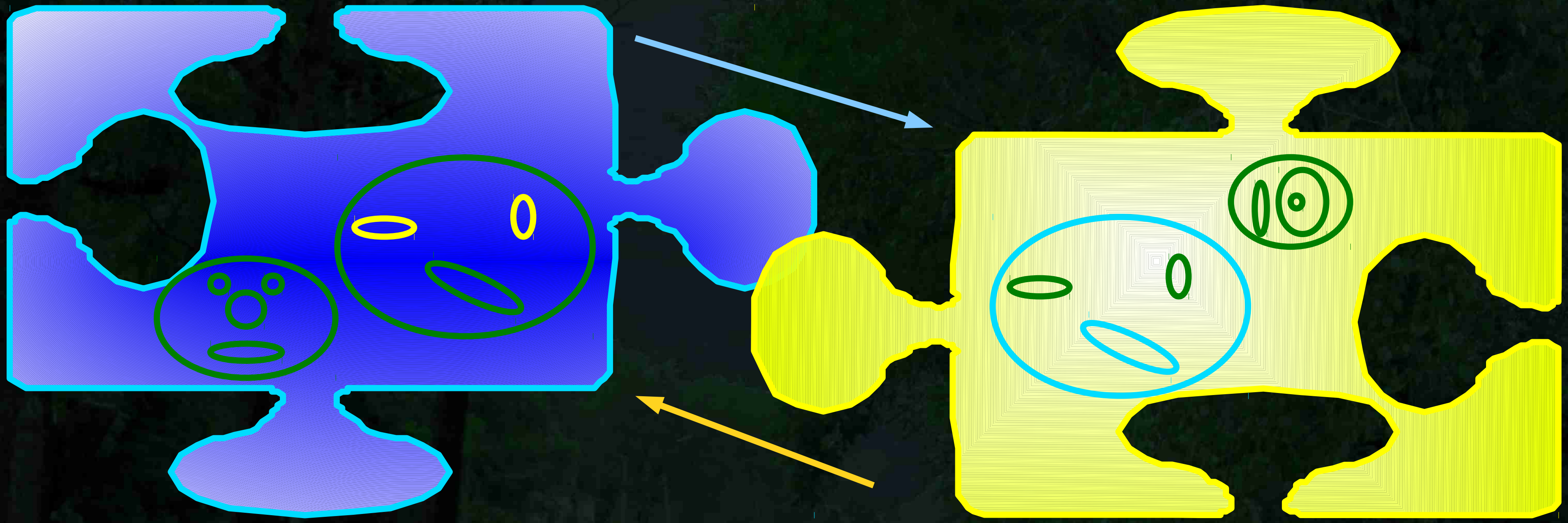
- *Hardware*
- *Algorithms*
- *Digital libraries*
- *Observational Data*



A view on *system architecture*



??????????? Middlware ?????????????



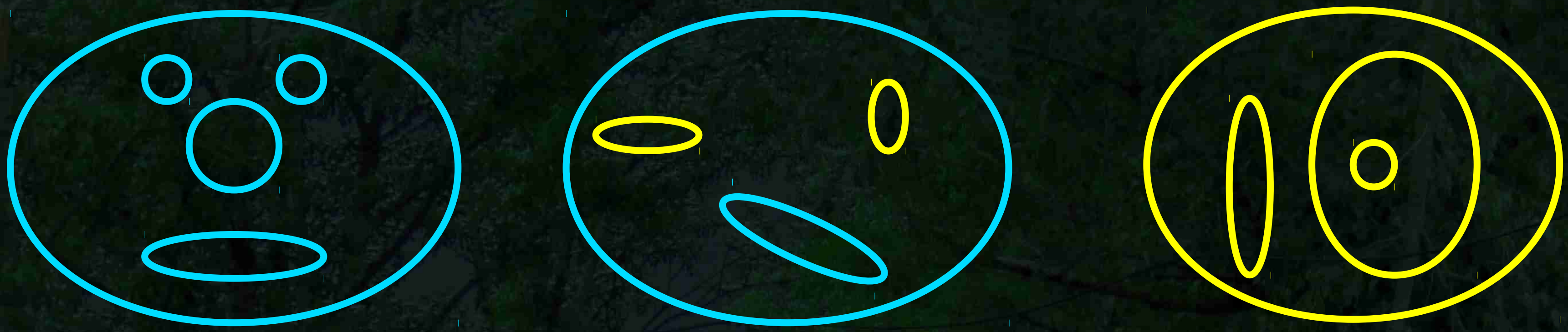
Retrospective paradigm

need Middleware to avoid :

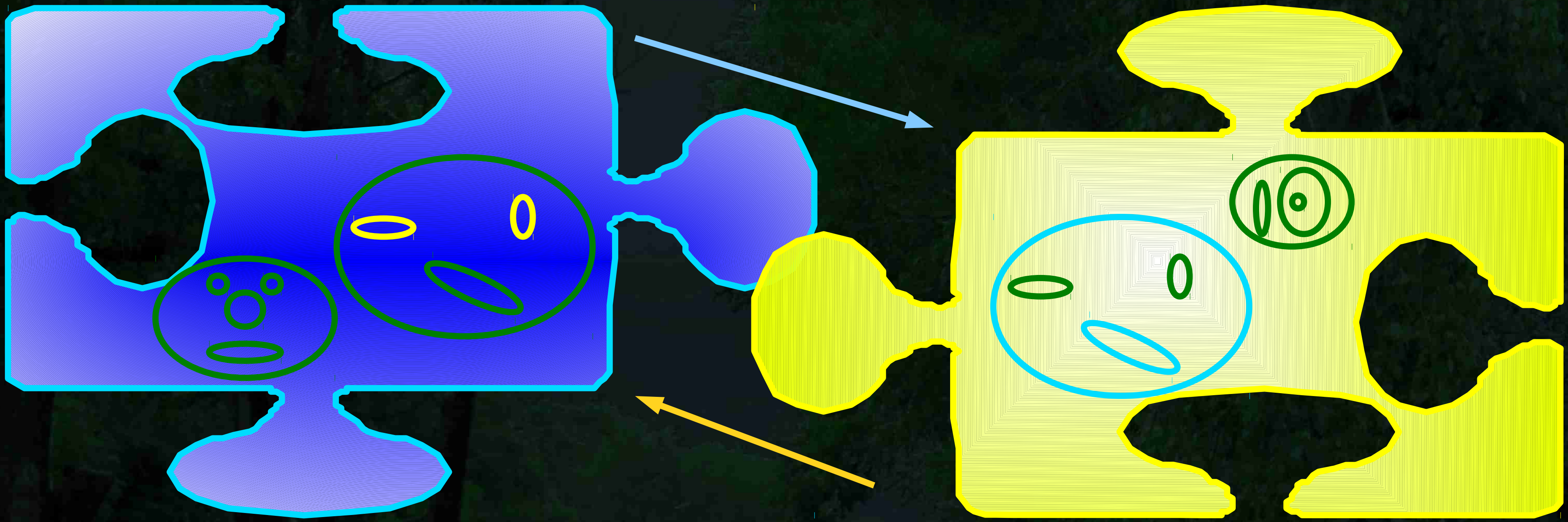
- ❀ Data mismatch for same *t*,
- ❀ Missing *t* for some data set,
- ❀ Access conflicts,
- ❀ Resource exhausting

in order to come

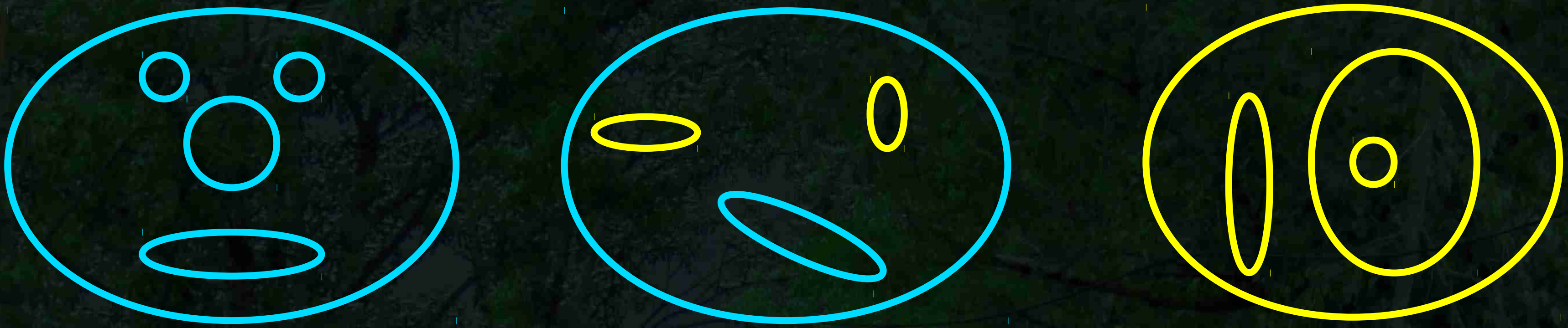
a way to fast responsive, fully consistent, fault tolerant system!



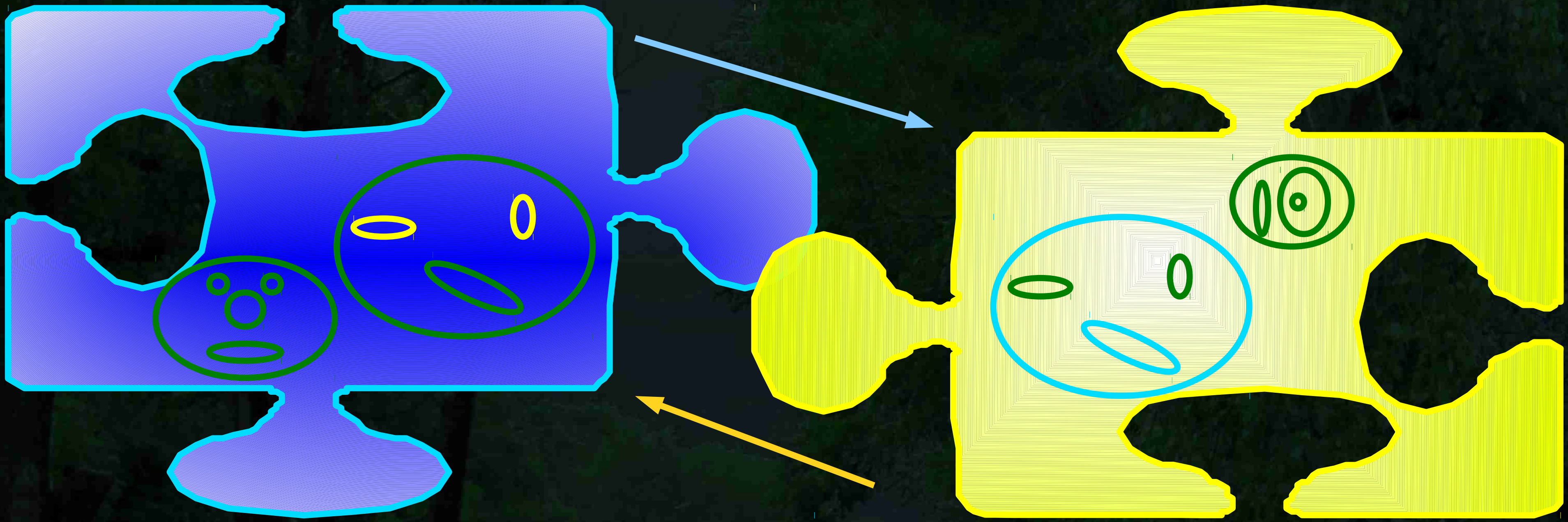
Middlware proposed to be the



Global address space of shared memory



for original data (including monitoring and code sources with history)



stored in network of retrospective DBMS.

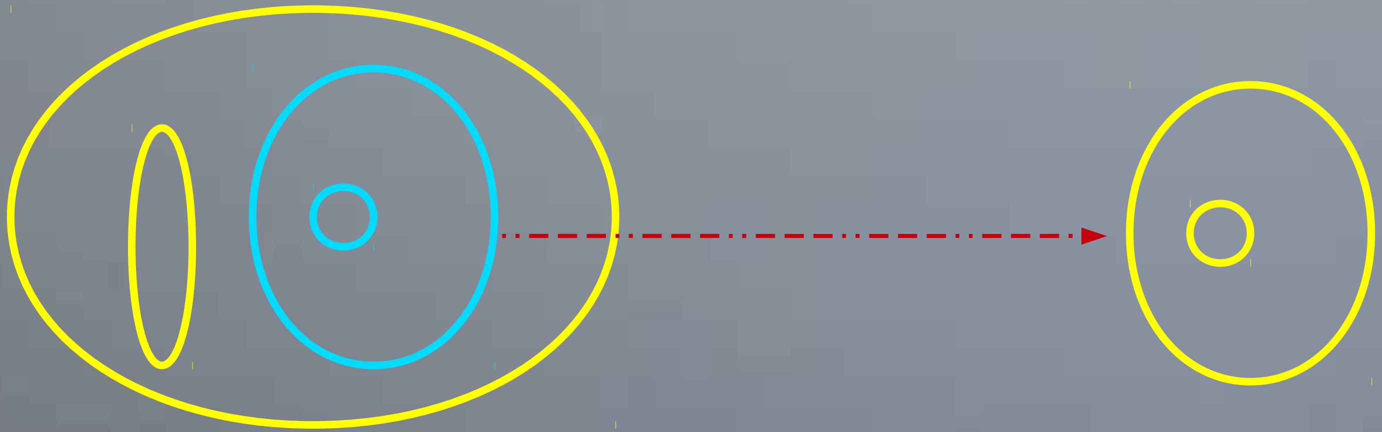
The retrospective DBMS

- store and retrieve time marked data objects,
- transmit (to subscribers) and receive changes in related activities,
- perform proper history degradation to free necessary space,
- keep trace of activity states,
- support frozen timed symbolic links for branching of activities.

The retrospective DBMS

- store and retrieve time marked data objects,
- transmit (to subscribers) and receive changes in related activities,
- perform proper history degradation to free necessary space,
- keep trace of activity states,
- support frozen timed symbolic links for branching of activities.

The retrospective DBMS



- store and retrieve time marked data objects,
- transmit (to subscribers) and receive changes in related activities,
- perform proper history degradation to free necessary space,
- keep trace of activity states,
- support frozen timed symbolic links for branching of activities.

The retrospective DBMS

- store and retrieve time marked data objects,
- transmit (to subscribers) and receive changes in related activities,
- perform proper history degradation to free necessary space,
- keep trace of activity states,
- support frozen timed symbolic links for branching of activities.

My thanks for support to:

Mathnet (2004)

RCDL
(2005-2012)

Ailamazyan PSI RAS
(2005-2012)