# New concepts for EASy-Producer (Languages)

This living document contains a summary of discussions about new EASy-Producer concepts, partly just as a memory protocol and not fully elaborated. Second part contains already realized new concepts.

# In discussion

## IVML: Extension of enums

1. Klaus...
2. Type refinement

Example:

project A {

enum Color {red, green};

}

project B {

enum MyColor **refines Color** {yellow};

}

Potential problem:

* Constraints: shall work, however the refined enum can be used wherever the original enum was used (akin to compounds) so that equality and unequality checks can then be applied to a broader range, which might lead to a different semantic as intended.
* Workaround: Make explicit which enums can be extended, which not (“final”)
* Ordered enums: no overriding of existing ordinal numbers

## IVML: Evals

Current state (and decision): The evaluation sequence determined by an enum is applied only once for the first evaluation. Actually, evals shall only improve reasoning speed rather than implicitly introduce an evaluation order leading to different results.

To check: The “initialization” of “anchor” types in QualiMaster did not work independently of the sequence (reasoner).

Type consistency

Current state: Although not proven, this shall be ok for now due to the assignment compatibility along compound refinements, the base-type compatibility of typedefs and even in the case of typedefs on refined compounds.

## IVML: Mandatory

Problem: isConfigured(.) applies immediately and breaks staged configuration.

A mandatory modifier for variable forcing a check at the end of reasoning would potentially fall too short. A specification of the “variable lifecycle phase” (or however we call that) would be more generic. Further discussion is needed.

Initial idea: something more global than “mandatory”., “optional”, but what?

## IVML: Specifying details of a reference (new)

Current solution – might be the most generic:

compound Flow {

refTo(Node) next;

// additional information

}

compound Node {

setOf(Flow) next;

}

Problem: If node hierarchy becomes complex and flows shall be limited to specific parts of a topology, we need a flow and a node hierarchy.

Discussion:

* Allow co-variance in compounds – redefine slot with more specific type.

TBD: Detail example – does this help?

* Some form of “association class”

TBD: Detail example – does this help? Is it only helpful for undirected references.

* Do we need undirected / bi-directed references for general topology modeling?

## IVML: Aggregation relation (new)

See “To connect or not to connect” (really needed)

## IVML: Do we need constraint values in eval blocks?

It might be an issue that constraint expressions work, constraint variables (referring to a constraint) are not considered. Identified by Roman…

## VIL: Creating a configuration (new)

Testing (as well as profiling of pipelines in QM) requires the creation of a permanent or temporary configuration. So VIL shall be able to create new instances of project, imports, variables (not necessarily types), the related configuration and to pass the new instances on for further instantiation as well as to store them if needed (what about the existing model).

Background: Currently, in QM we need the creation of projects and variables as well as value assignment. Thereby, it would be beneficial to copy values of existing variables on demand (also for differentiating instances of configured elements at runtime).

# Tooling functionality

## IVML: Constraint / Reasoning debugger

Stepwise evaluation pointing into IVML code, created constraints, value display

## VIL: Debugger

Stepwise execution pointing into VIL/VTL code, value display, breakpoints

## EASy: New UI

Oh yeah, the old discussion

# Done / Realized

## VIL/VTL: explicit for and while loops

Clarify semantics of map vs loops and introduce explicit for and while loops.

## Constants

Defining a constant value that can be assigned instead of repeating over and over the same value would be helpful. BTW, TVL has constants.

Agreed solution: constant modifier for variables.

## IVML: Multiple inheritance

As part of the IVML article…

Agreed solution: IVML compounds shall support multiple inheritance (done)

## IVML: Multiple inheritance

As part of the IVML article… can we re-define an IVML compound slot, in particular with a refined compound type compared to the original slot.

Agreed solution: Re-definition of compound slots shall be supported, re-definition is allowed only if the type remains the same (re-definition of default through slot shadowing) or by a refining slot type. Constrants, if require must be re-defined.