SEALS: A framework for building Self-Adaptive Virtual Machines

Gwendal Jouneaux [1] Olivier Barais [1]
Benoit Combemale [1] Gunter Mussbacher [2]

[1]Univ. Rennes, Inria, IRISA – Rennes, France

[2]McGill University – Montreal, Canada

SLE — October 18, 2021

Context

Software . . .

Context

Software . . .

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)

Context

Software …

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo[1], Netflix[1])

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Context

Software …

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo[1], Netflix[1])

Software languages …

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Context

Software . . .

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo[1], Netflix[1])

Software languages . . .

- Can abstract concerns into high level constructs (e.g., memory management)

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Context

Software . . .

- Evolve in complex/changing environment (e.g, Cloud, embedded systems)
- Need dynamic adaptation to best deliver the service (e.g., Waymo[1], Netflix[1])

Software languages . . .

- Can abstract concerns into high level constructs (e.g., memory management)

Challenge : provide functionality for self-adaptation in DSLs

[1] Cf. `https://waymo.com`, `https://www.netflix.com`

Contribution overview

The SEALS Framework supports the definition of . . .

1. Language abstract syntax and **semantics**
2. Correctness envelope defining the valid semantics for a domain concept
3. The feedback loop and associated trade-off reasoning
4. Adaptations and the predictive model of their impact on the trade-off

What are Self-Adaptable Virtual Machines

What are Self-Adaptable Virtual Machines

- A specific case of Self-Adaptable Languages [1]
- Reification of a **feedback loop** [2] in language **operational semantics**

[1] G. Jouneaux, O. Barais, B. Combemale, *et al.*, "Towards Self-Adaptable Languages," in *Onward! 2021*, Chicago, United States, Oct. 2021. [Online]. Available: `https://hal.inria.fr/hal-03318816`

[2] IBM, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, 2006

What are Self-Adaptable Virtual Machines

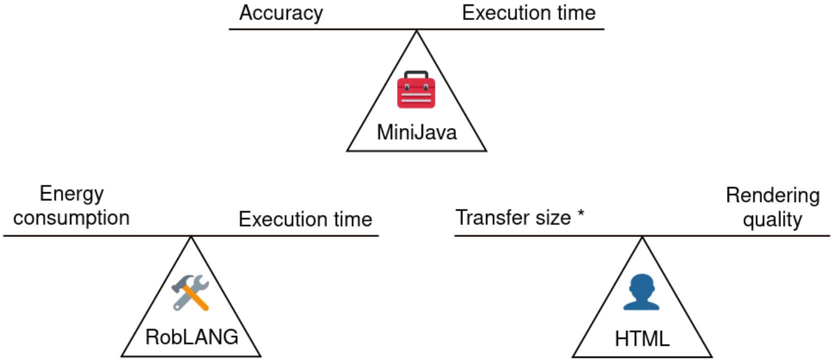
- A specific case of Self-Adaptable Languages [1]
- Reification of a **feedback loop** [2] in language **operational semantics**

Additionally, in our case :

- Pluggable architecture for adaptations
- Delegation of configuration responsibilities

[1] G. Jouneaux, O. Barais, B. Combemale, *et al.*, “Towards Self-Adaptable Languages,” in *Onward! 2021*, Chicago, United States, Oct. 2021. [Online]. Available: `https://hal.inria.fr/hal-03318816`
[2] IBM, “An architectural blueprint for autonomic computing,” *IBM White Paper*, vol. 31, 2006

Motivating Examples



Accuracy

Execution time

MiniJava

Energy consumption

Execution time

RobLANG

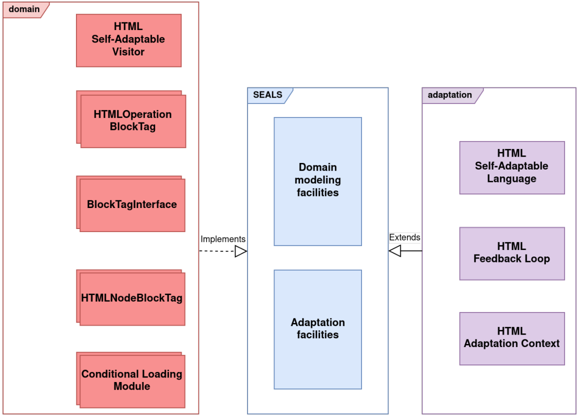
Transfer size *

Rendering quality

HTML

* Transfer size is proportional to energy consumption (Cf. `https://www.websitecarbon.com/`)

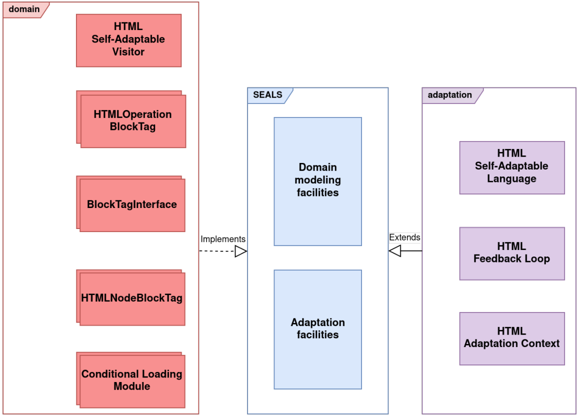
Approach overview



Approach overview on the HTML use case

Approach overview

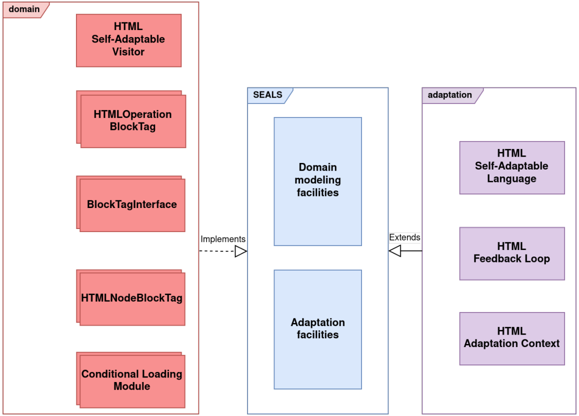
- Concepts provided by SEALS



Approach overview on the HTML use case

Approach overview

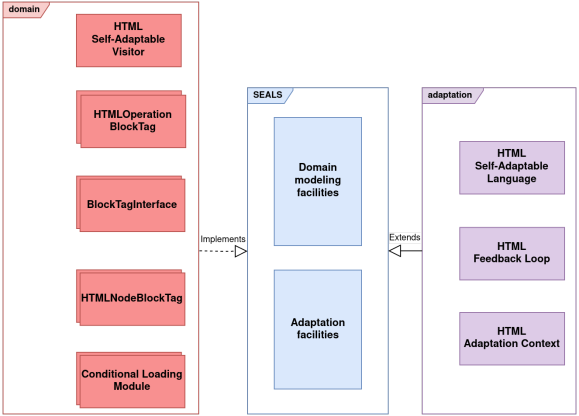
- Concepts provided by SEALS
- Modeling of domain concepts



Approach overview on the HTML use case

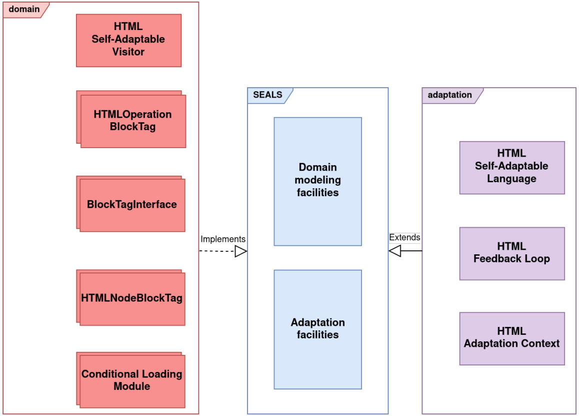
Approach overview

- Concepts provided by SEALS
- Modeling of domain concepts
- The specialized adaptation process



Approach overview on the HTML use case

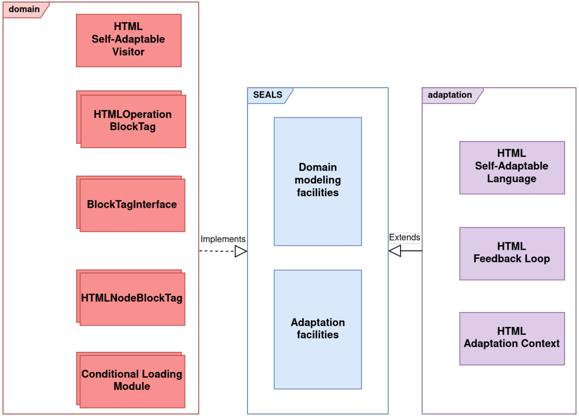
Modeling of domain concepts



Approach overview on the HTML use case

Modeling of domain concepts

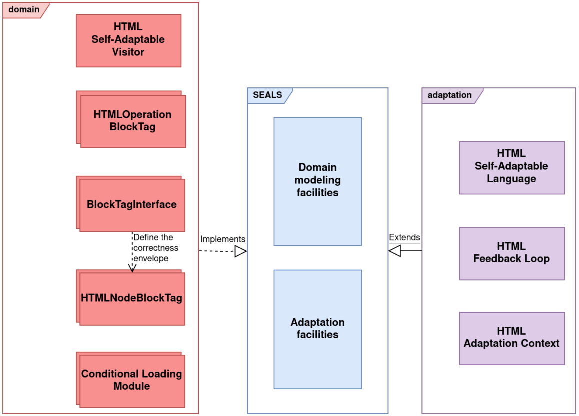
1. Define the abstract syntax



Approach overview on the HTML use case

Modeling of domain concepts

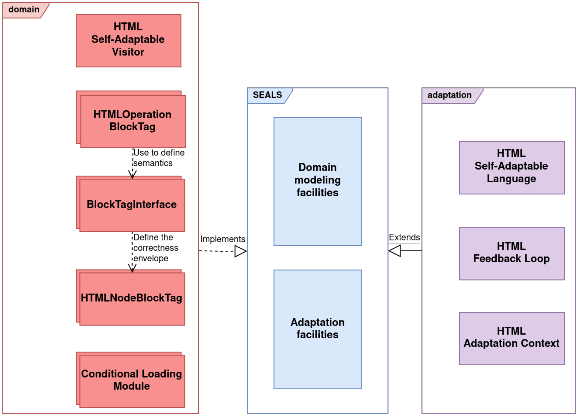
1. Define the abstract syntax
2. Create the correctness envelope



Approach overview on the HTML use case

Modeling of domain concepts

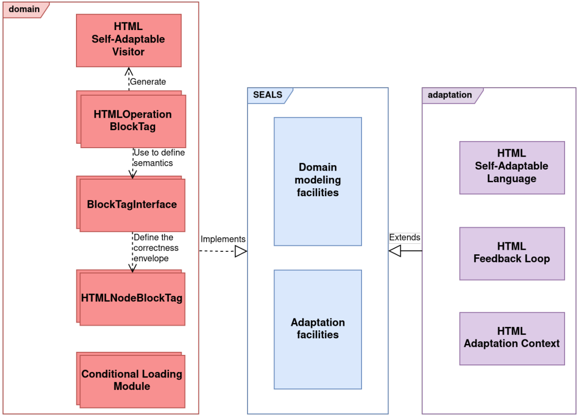
1. Define the abstract syntax
2. Create the correctness envelope
3. Implement the operations



Approach overview on the HTML use case

Modeling of domain concepts

1. Define the abstract syntax
2. Create the correctness envelope
3. Implement the operations
4. Generate the concrete visitor

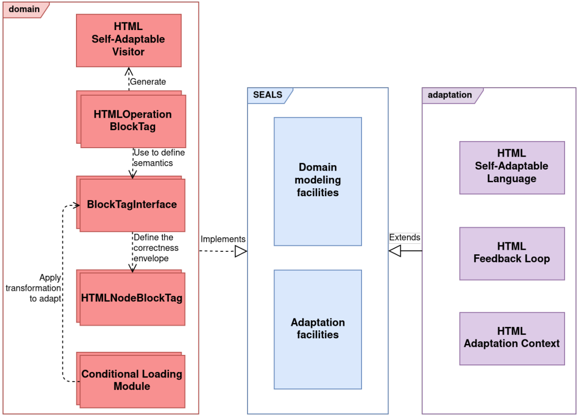


Approach overview on the HTML use case

Modeling of domain concepts

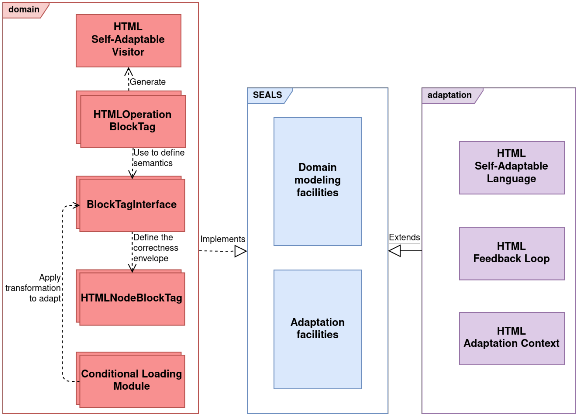
1. Define the abstract syntax
2. Create the correctness envelope
3. Implement the operations
4. Generate the concrete visitor

→ Develop adaptation modules



Approach overview on the HTML use case

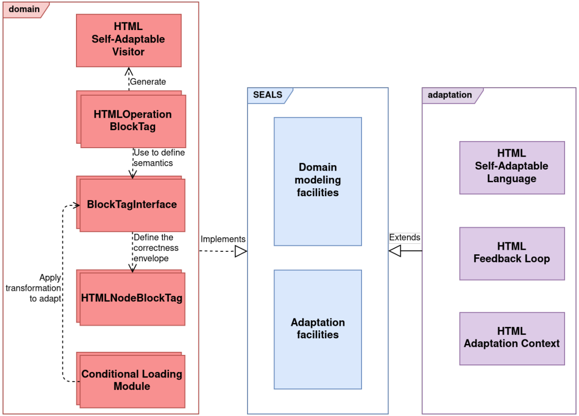
Specialize the adaptation process



Approach overview on the HTML use case

Specialize the adaptation process

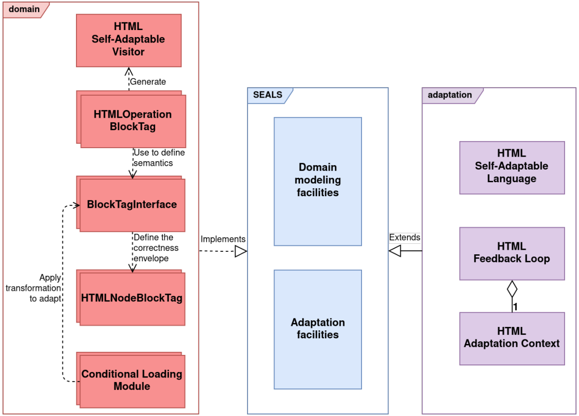
1. Specialize the Adaptation Context
   - Define the KPIs
   - Define the monitored resources



Approach overview on the HTML use case

Specialize the adaptation process

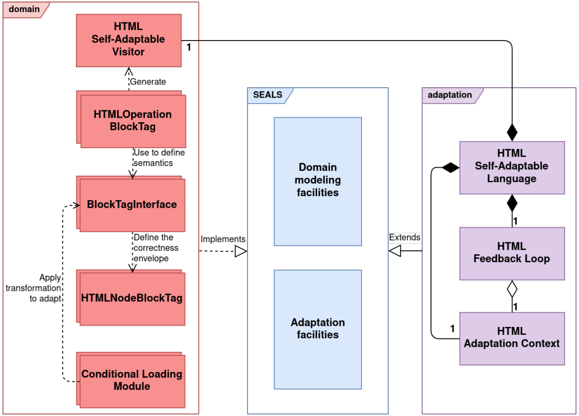
1. Specialize the Adaptation Context
   - ▶ Define the KPIs
   - ▶ Define the monitored resources
2. Specialize the Feedback loop
   - ▶ Implement the **MAPE** steps
   - ▶ Define the loop trigger



Approach overview on the HTML use case

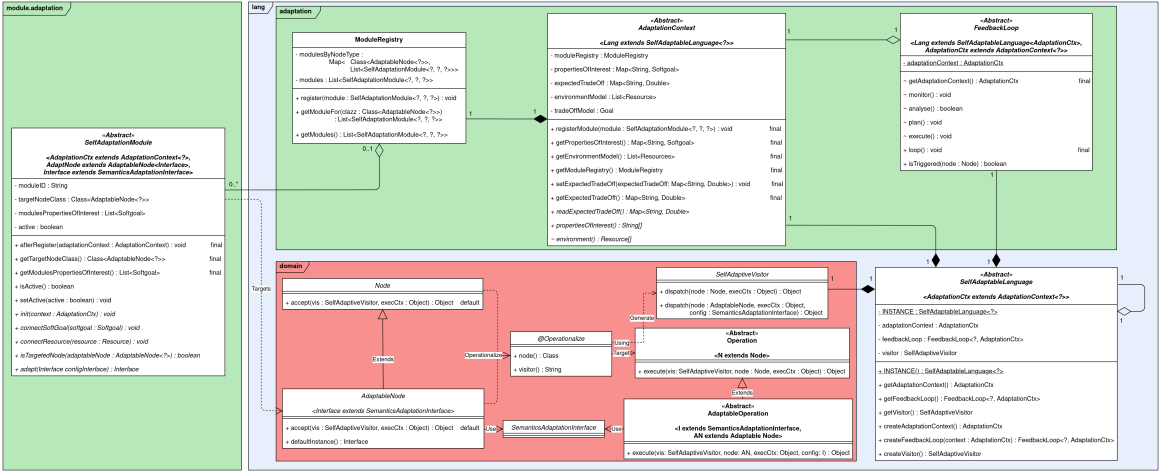
Specialize the adaptation process

1. Specialize the Adaptation Context
   - Define the KPIs
   - Define the monitored resources
2. Specialize the Feedback loop
   - Implement the **MAPE** steps
   - Define the loop trigger
3. Connect the components



Approach overview on the HTML use case

Java Implementation



module.adaptation

«Abstract»
SelfAdaptationModule

<AdaptationCtx extends AdaptationContext<?>,
AdaptNode extends AdaptableNode<Interface>,
Interface extends SemanticsAdaptationInterface>

- moduleID : String
- targetNodeClass : Class<AdaptableNode<?>>
- modulesPropertiesOfInterest : List<Softgoal>
- active : boolean

+ afterRegister(adaptationContext : AdaptationContext) : void final
+ getTargetNodeClass() : Class<AdaptableNode<?>> final
+ getModulesPropertiesOfInterest() : List<Softgoal> final
+ isActive() : boolean
+ setActive(active : boolean) : void
+ *init(context : AdaptationCtx) : void*
+ *connectSoftGoal(softgoal : Softgoal) : void*
+ *connectResource(resource : Resource) : void*
+ *isTargetedNode(adaptableNode : AdaptableNode<?>) : boolean*
+ *adapt(Interface configInterface) : Interface*

lang

adaptation

ModuleRegistry

- modulesByNodeType :
  Map< Class<AdaptableNode<?>>,
  List<SelfAdaptationModule<?, ?, ?>>>
- modules : List<SelfAdaptationModule<?, ?, ?>>

+ register(module : SelfAdaptationModule<?, ?, ?>) : void
+ getModuleFor(clazz : Class<AdaptableNode<?>>)
  : List<SelfAdaptationModule<?, ?, ?>>
+ getModules() : List<SelfAdaptationModule<?, ?, ?>>

«Abstract»
AdaptationContext

<Lang extends SelfAdaptableLanguage<?>>

- moduleRegistry : ModuleRegistry
- propertiesOfInterest : Map<String, Softgoal>
- expectedTradeOff : Map<String, Double>
- environmentModel : List<Resource>
- tradeOffModel : Goal

+ registerModule(module : SelfAdaptationModule<?, ?, ?>) : void final
+ getPropertiesOfInterest() : Map<String, Softgoal> final
+ getEnvironmentModel() : List<Resources> final
+ getModuleRegistry() : ModuleRegistry final
+ setExpectedTradeOff(expectedTradeOff: Map<String, Double>) : void final
+ getExpectedTradeOff() : Map<String, Double> final
+ *readExpectedTradeOff() : Map<String, Double>*
+ *propertiesOfInterest() : String[]*
~ *environment() : Resource[]*

«Abstract»
FeedbackLoop

<Lang extends SelfAdaptableLanguage<AdaptationCtx>,
AdaptationCtx extends AdaptationContext<?>>

- adaptationContext : AdaptationCtx

~ getAdaptationContext() : AdaptationCtx final
~ monitor() : void
~ analyse() : boolean
~ plan() : void
~ execute() : void
+ loop() : void final
+ isTriggered(node : Node) : boolean

domain

Node

+ accept(vis : SelfAdaptiveVisitor, execCtx : Object) : Object default

AdaptableNode

<Interface extends SemanticsAdaptationInterface>

+ accept(vis : SelfAdaptiveVisitor, execCtx : Object) : Object default
+ defaultInstance() : Interface

@Operationalize

+ node() : Class
+ visitor() : String

SelfAdaptiveVisitor

+ dispatch(node : Node, execCtx : Object) : Object
+ dispatch(node : AdaptableNode, execCtx : Object,
  config : SemanticsAdaptationInterface) : Object

«Abstract»
Operation

<N extends Node>

+ execute(vis: SelfAdaptiveVisitor, node : Node, execCtx : Object) : Object

SemanticsAdaptationInterface

«Abstract»
AdaptableOperation

<I extends SemanticsAdaptationInterface,
AN extends Adaptable Node>

+ execute(vis: SelfAdaptiveVisitor, node: AN, execCtx: Object, config: I) : Object

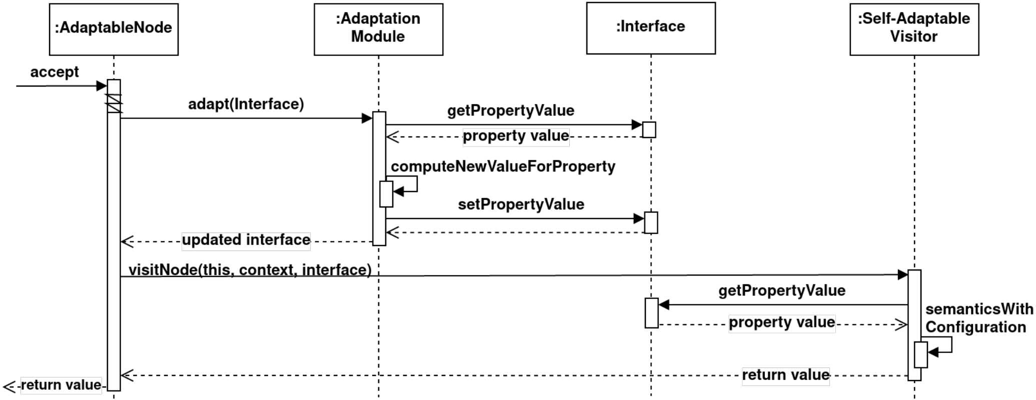
«Abstract»
SelfAdaptableLanguage

<AdaptationCtx extends AdaptationContext<?>>

- INSTANCE : SelfAdaptableLanguage<?>
- adaptationContext : AdaptationCtx
- feedbackLoop : FeedbackLoop<?, AdaptationCtx>
- visitor : SelfAdaptiveVisitor

+ INSTANCE() : SelfAdaptableLanguage<?>
+ getAdaptationContext() : AdaptationCtx
+ getFeedbackLoop() : FeedbackLoop<?, AdaptationCtx>
+ getVisitor() : SelfAdaptiveVisitor
+ createAdaptationContext() : AdaptationCtx
+ createFeedbackLoop(context : AdaptationCtx) : FeedbackLoop<?, AdaptationCtx>
+ createVisitor() : SelfAdaptiveVisitor

Correctness envelope implementation



Example of use of the correctness envelope

Abstraction

Is the set of proposed constructs relevant for a language engineer ?

Abstraction

Is the set of proposed constructs relevant for a language engineer ?

Two mandatory components :

Abstraction

Is the set of proposed constructs relevant for a language engineer ?

Two mandatory components :

- The Feedback Loop (MAPE-K [2])
  - *Adaptation Context* (**K**)
  - *Feedback Loop* (**MAPE**)

[2] IBM, “An architectural blueprint for autonomic computing,” *IBM White Paper*, vol. 31, 2006

Abstraction

Is the set of proposed constructs relevant for a language engineer ?

Two mandatory components :

- The Feedback Loop (MAPE-K [2])
  - *Adaptation Context* (**K**)
  - *Feedback Loop* (**MAPE**)
- The Operational Semantics
  - *Adaptable Operation* / *Operation*
  - Correctness envelope

[2] IBM, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, 2006

Abstraction

Is the set of proposed constructs relevant for a language engineer ?

Two mandatory components :

- The Feedback Loop (MAPE-K [2])
  - *Adaptation Context* (**K**)
  - *Feedback Loop* (**MAPE**)
- The Operational Semantics
  - *Adaptable Operation* / *Operation*
  - Correctness envelope

→ Abstraction of their interactions ←

[2] IBM, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, 2006

Correctness envelope

Do the correctness envelopes provide assurances on the validity of the adapted semantics?

Correctness envelope

Do the correctness envelopes provide assurances on the validity of the adapted semantics?

- Prevent adaptations execution as default behavior

Correctness envelope

Do the correctness envelopes provide assurances on the validity of the adapted semantics?

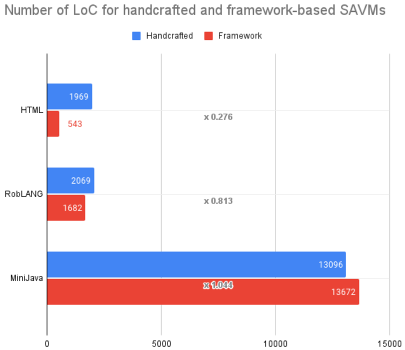
- Prevent adaptations execution as default behavior
- Allowing adaptations requires to :
  - Define subjects of adaptation
  - Implement operations on them

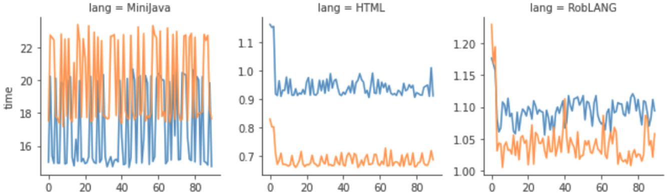
Correctness envelope

Do the correctness envelopes provide assurances on the validity of the adapted semantics?

- Prevent adaptations execution as default behavior
- Allowing adaptations requires to :
  - Define subjects of adaptation
  - Implement operations on them
- Semantics changes are bounded by the set of possible instances

Performances and code size





- Operations implementation as class is heavy
- No general trend
- These results point to future work

Conclusion

The SEALS Framework . . .

- Provides abstractions to define Self-Adaptive Virtual Machines
  - Language abstract syntax and operational semantics
  - The feedback loop and associated trade-off reasoning
- Provides abstractions to define and restrain adaptations
  - Adaptations and the predictive model of their impact on the trade-off
  - Correctness envelope defining the valid semantics for a domain concept

Github repository : `https://github.com/gwendal-jouneaux/SEALS`

Perspectives

- Language tooling for better understanding by the language user.
- Managing the feature interaction between adaptations
- A principled approach to Self-Adaptable Virtual Machines
- Work on the framework performances

References

- G. Jouneaux, O. Barais, B. Combemale, and G. Mussbacher, “Towards Self-Adaptable Languages,” in *Onward! 2021*, Chicago, United States, Oct. 2021. [Online]. Available: `https://hal.inria.fr/hal-03318816`.
- IBM, “An architectural blueprint for autonomic computing,” *IBM White Paper*, vol. 31, 2006.