A Framework for Designing and Implementing the Ada Standard Container Library

Jordi Marco and Xavier Franch
Universitat Politècnica de Catalunya (UPC)
Catalonia (Spain)

DALI project: http://www.lsi.upc.es/~gessi/

SIGAda 2003. December 7-11, 2003, San Diego, California, USA

Contents

Motivation

Assumption: Ada shall include a Standard Container Library (SCL)

- As other Object Oriented Languages (C++, Java, Eiffel, …)
- Initiatives
  - Some Action Items issued by the Ada Conformity Assessment Authority (remarkably AI-302)
  - Booch Components, Charles Container Library, …
  - Some events, such as the Standard Container Library for Ada workshop held during the Ada-Europe 2002 Conference
  - Other claims

Goal: to provide a framework named the ***Shortcut-based Framework*** **(SBF)** to be considered as a baseline upon which a high-quality Ada Standard Container Library can be built

- SBF solve the majority of quality drawbacks present in the most widespread container libraries

A Quality Model for the Ada Standard Container Library

Based on the ISO/IEC 9126-1 Quality Standard which:

- provides a good framework for determining a quality model

  general characteristics → subcharacteristics → attributtes

- just fixes the top level hierarchy (characteristics and subcharacteristics)
- mentions the convenience of creating hierarchies of quality features
- is widespread

The ISO/IEC 9126-1 Quality Standard

- Multilevel hierarchy defined by:
  - 6 top level characteristics and their subcharacteristics
  - Attributes: Measurable, values computed by a metric.
- In our approach, intermediate hierarchies of subcharacteristics or attributes may appear
- Quality requirements may be defined as restrictions over the model

Characteristics	Subcharacteristics
Functionality	suitability, accuracy, interoperability, security , functionality compliance
Reliability	maturity, fault tolerance, recoverability, reliability compliance
Usability	understandability, learnability, operability, attractiveness, usability compliance
Efficiency	time behavior, resource behavior, efficiency compliance
Maintainability	analyzability, changeability, stability, testability, maintainability compliance
Portability	adaptability, installability, co-existence, replaceability, portability compliance

Quality Attributes for Functionality in Container Libraries

Functionality is probably the most relevant quality characteristic in the domain of container libraries.

Success of the Ada SCL requires exhibiting the appropriate (not necessarily exhaustive) functionality once considered its design requirements.

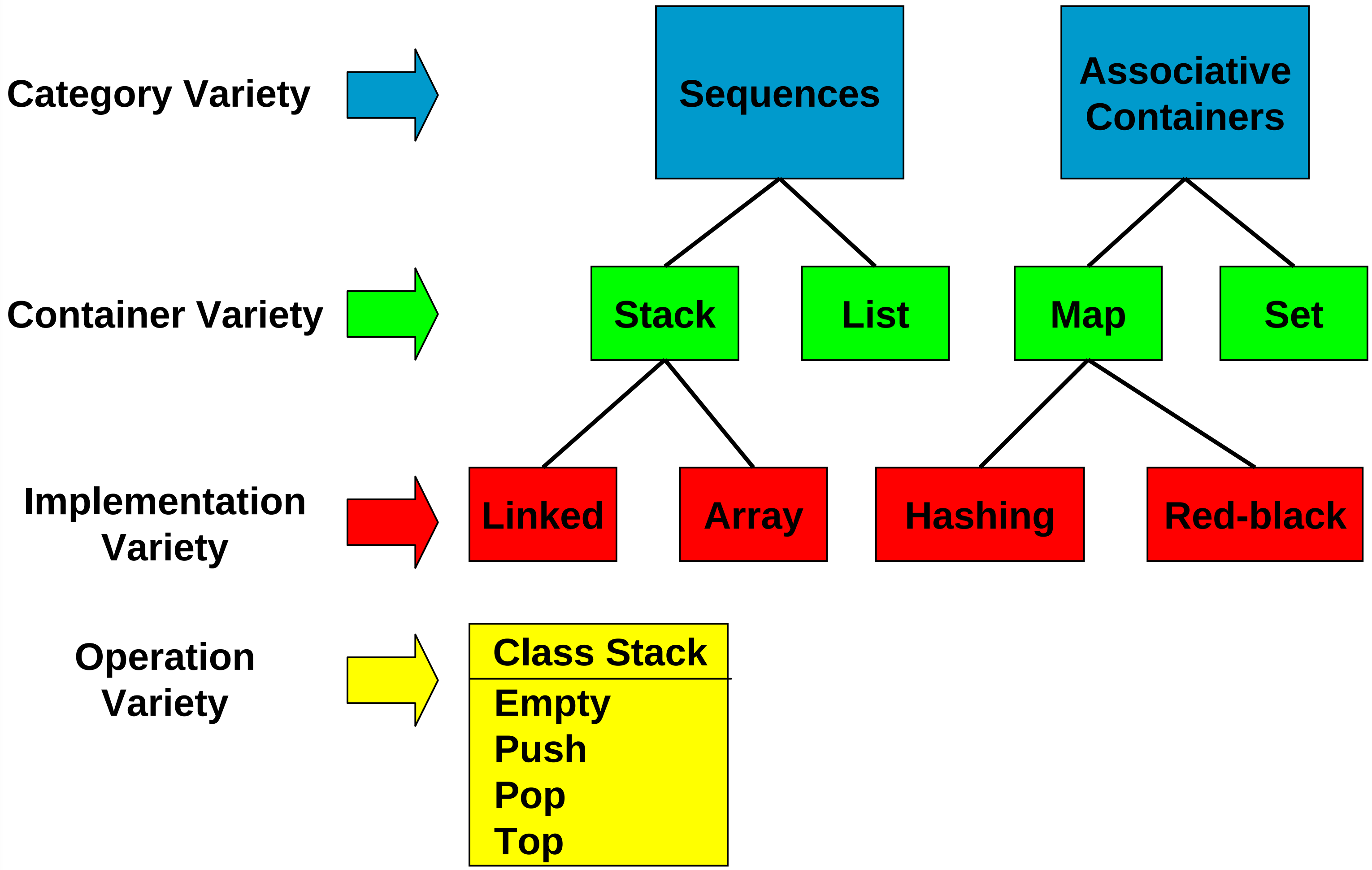
Suitability

Suitability is perhaps the more complex *functionality subcharacteristic*

We decompose it into two new *subcharacteristics*:

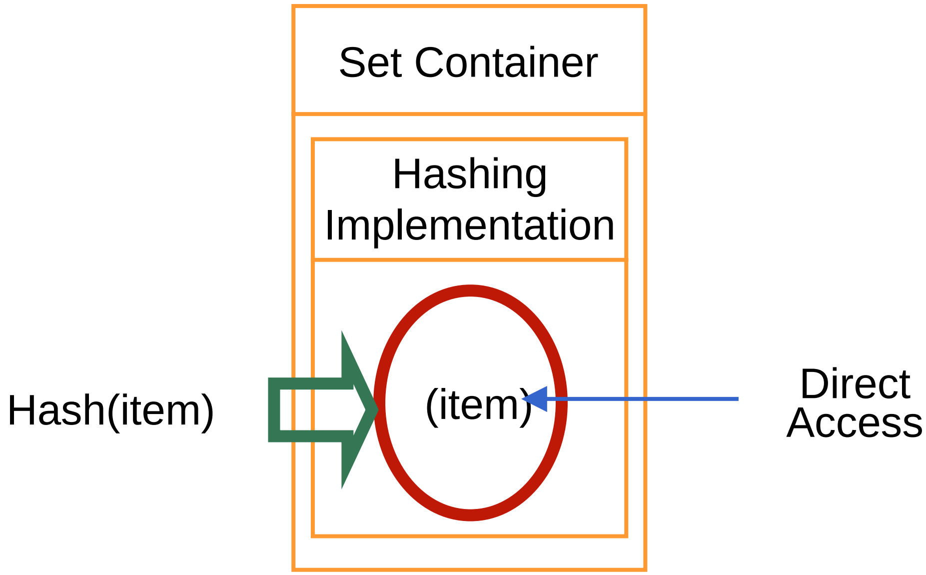
- *Core Suitability*. Types of containers and their implementations
- *General Suitability*. Additional functionalities

Core Suitability attributes

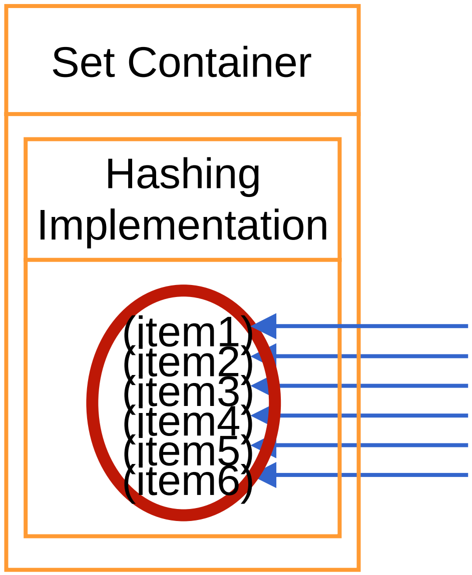


General Suitability attributes

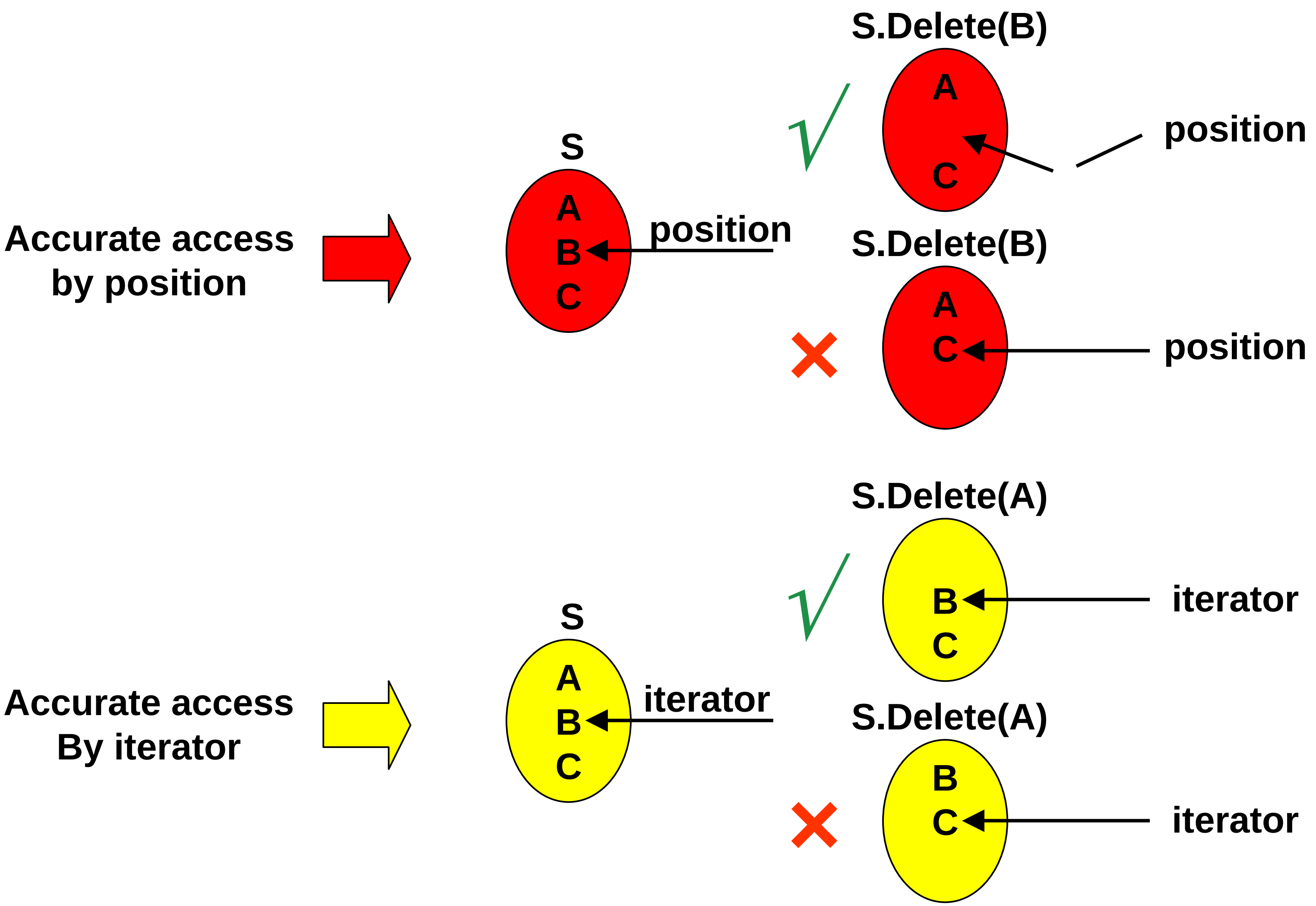
Direct access by position:



Iterators:



Accuracy attributes



The Shortcut Based Framework (SBF)

The Shortcut Concept

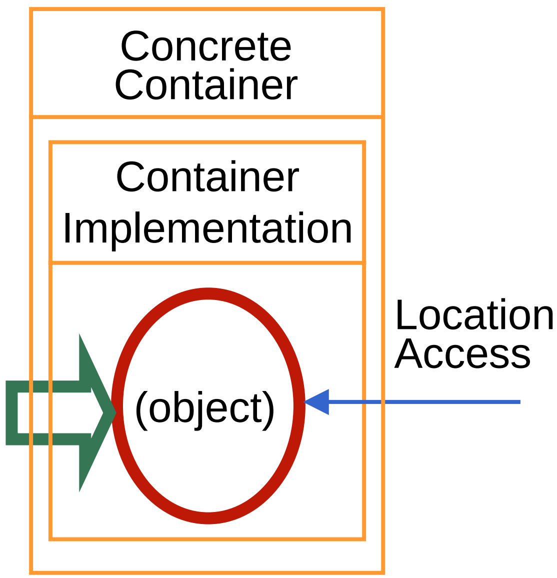
Design an object that encapsulates the concept of location or position of an object in a container, with the following requirements:

- Time efficiency
  - All the shortcuts operations have constant time
- Accuracy
  - It is bound to one and only one object in the container
  - It does not change while the object which it is bound to is inside the container
- Seccurity
  - Access by out-of-date or undefined shortcuts is avoided

The Shortcut Based Framework (SBF)

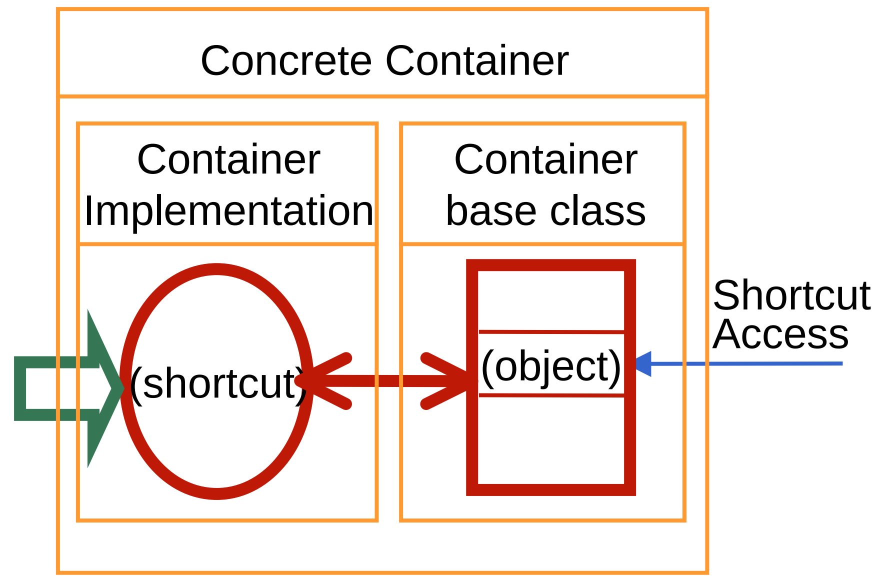
Classical solution

Key points:
- the objects are stored in the container implementation
- location access is provided by the container implementation



SBF solution

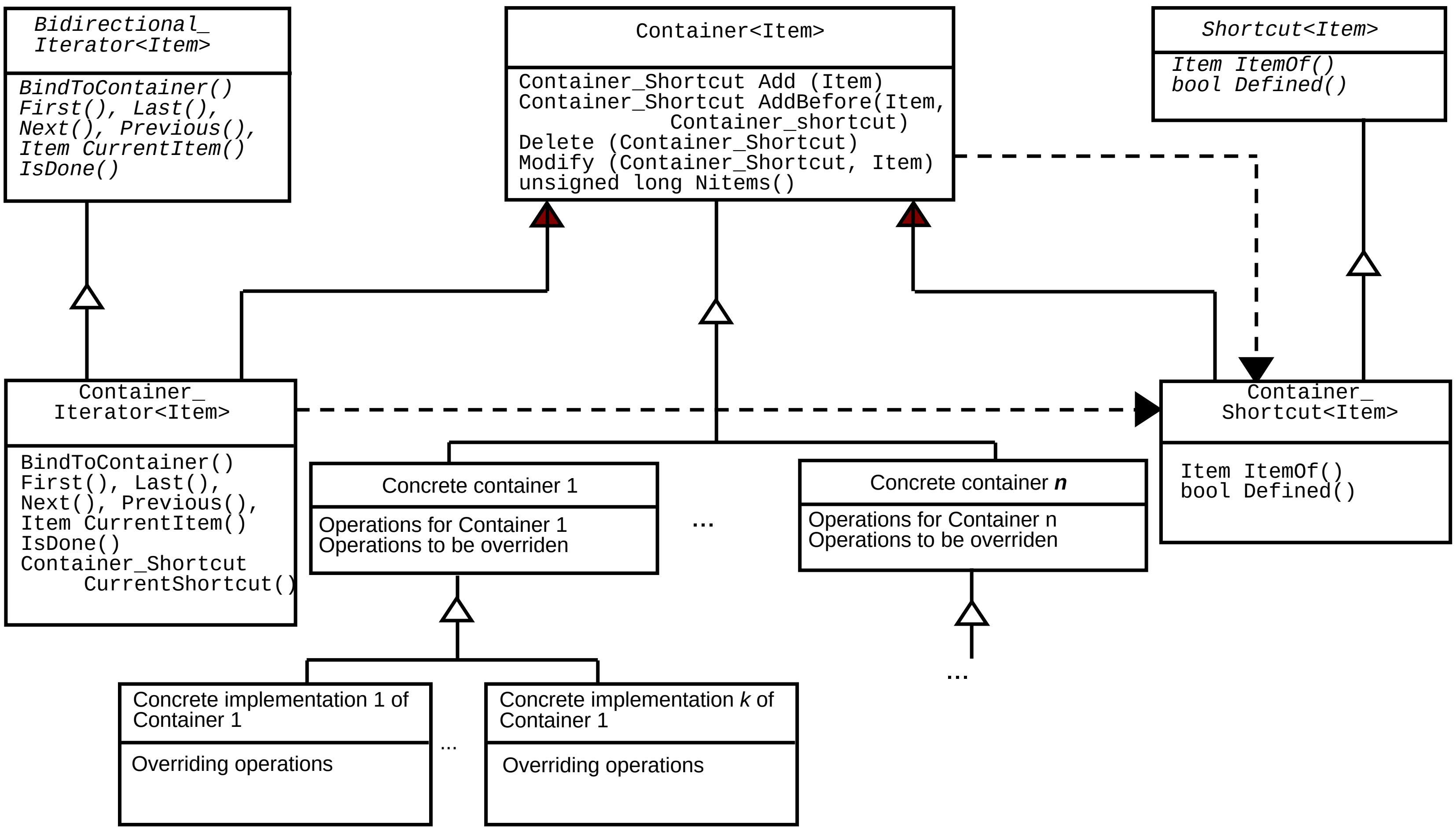
Key points:
- the objects are stored in a Container base class
- the container implementation store the shortcuts bound to them



Some remarks:
- shortcut operations and all iterator operations have constant time O(1)
- other former operations preserve the order of complexity
- It becomes easier to control and manage all exceptional situations

The Shortcut Based Framework (SBF)

The SBF Hierarchy



The Shortcut Based Framework (SBF)

Consequences

The Shortcut concept offers several benefits to container's inheritors:

- Access by shortcuts and iterations are independent of the underlying representation of a concrete container
- Efficiency of Shortcuts make possible reuse containers in context with high efficiency constraints
- The access to the objects by means of Shortcuts is accurate and secure
- The children classes inherits shortcuts operations as a *black* box
- Iterate with out committing to a specific container with the same performance

Drawbacks:

- Some time and space overhead $\rightarrow$ can be saved later

SBF Sample Code

Using SBF for an array based implementation (MapArray) of the concrete container Map

The Sample shows the main points of the SBF:

- The Application of the Template Method design pattern
- The use of parent and children classes as black boxes
- The persistence of iterators and shortcuts
- The possibility of define generic algorithms

Delete Containers.Maps procedure

```
procedure Delete (In_The_Container : in out Map; The_Key: Key) is
  Sh : Shortcut;
begin
    Sh := Dispatching_Get(In_The_Container,The_Key);
    Dispatching_Delete(In_The_Container,The_Key);
    Containers.Delete(Container(In_The_Container),Sh);
end Delete;

procedure Dispatching_Delete (In_The_Container : in out Map'Class; The_Key: Con_Key) is
begin
    Con_Delete(In_The_Container,The_Key);
end Dispatching_Delete;
```

Deleting the shorcut
from the implementation

Deleting the object
from the base class

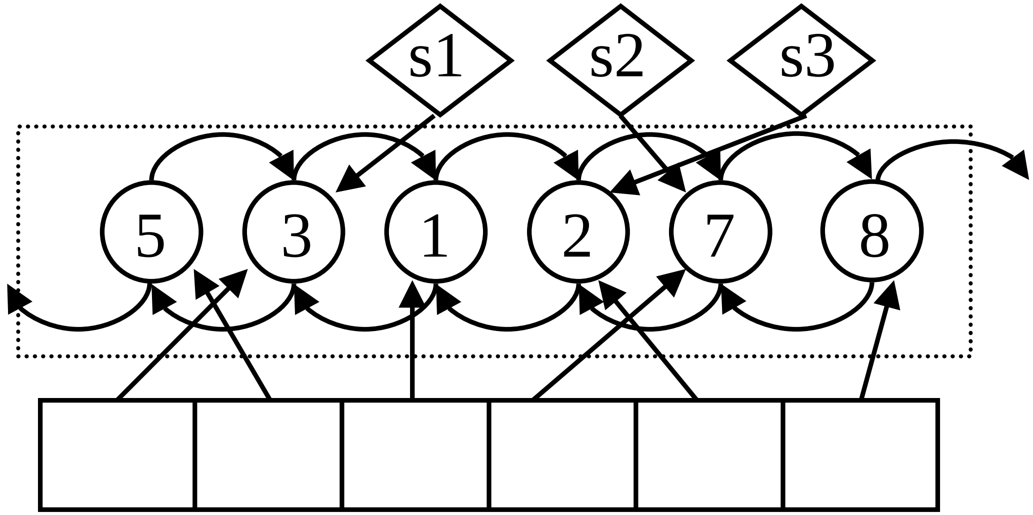
ConDelete Containers.Maps.Arrays procedure

```
procedure Con_Delete (In_The_Container : in out MapArray; The_Key: Con_Key) is
begin
   if not Con_Exist(In_The_Container,The_key) then
        raise Not_Existing_Key;
    end if;
    In_The_Container.FirstFree := In_The_Container.FirstFree -1;
    for i in In_The_Container.Cache .. In_The_Container.FirstFree-1 loop
        In_The_Container.MapA(i) :=
                In_The_Container.MapA(i+1);
    end loop;
    Finalize(In_The_Container.MapA(In_The_Container.FirstFree));
end Con_Delete;
```

SBF Sample Code generic algorithm: Sort

```
procedure Sort (C1: in out C.Container'Class) is
  function Min_In_Range is new GenericAlgorithms.Min_In_Range
                                  (Item => Item,  "<" => "<", BI => C.Container_Iterators);
  It1, It2, ItMin : C.Iterator;
begin
    if Cardinality(C1) /= 0 then
      Bind_To_Container(It1,C1); Bind_To_Container(It2,C1);
      Last(It2);  Next(It2);
      while not IsDone(It1) loop
         ItMin := Iterator(Min_In_Range(It1,It2));
         Swap(It1,ItMin);
         Next(It1);
      end loop;
    end if;
end Sort;
```

Sorting a MapArray Container

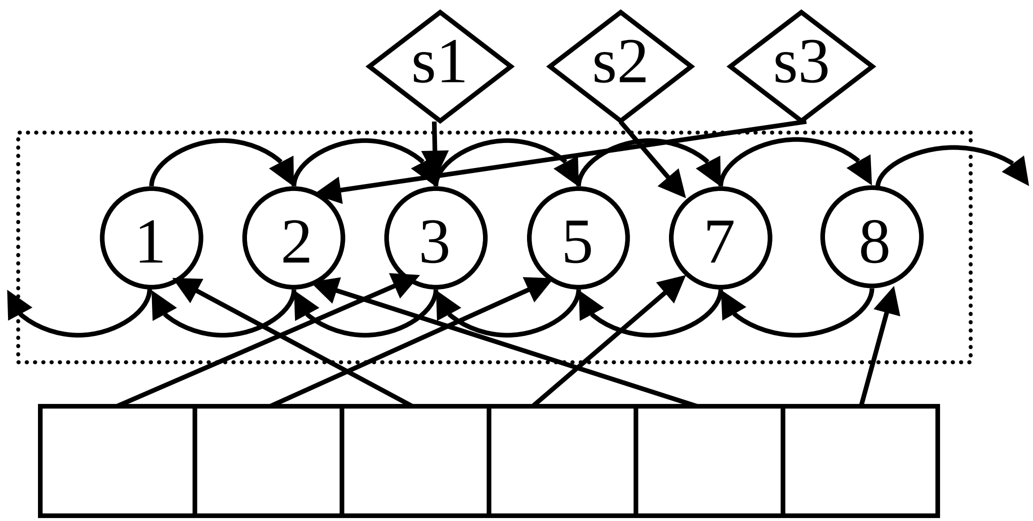


Client shortcuts

Container base class

Array implementation of the map class

After sorting a container client shortcuts refer to the same element and concrete container not change



Client shortcuts

Container base class

Array implementation of the map class

Evaluating the SBF

Assessment of the Goal Question Metrics:

- Core Suitability → not affected
- General Suitability → maximum values
- Algorithmic Variety (algorithms provided and posiblity of define new ones) → both cases are well-suited
- Accuracy and security → avoid wrong situations
- Efficiency → same order of magnitude
  → some overhead in real time and in resource utilization amortized iterating and in external references

Conclusions

Benefits:

- Provides High-quality access by position and iterators.
  - They are accurate and secure
  - Allow update the container during traversal
  - All the containers offers the same operations for iterating and access by position with the same performance
- Provides absolute freedom for the core suitability of the library
- Extending the library is easier due to the reuse of code of shortcuts and iterators
- Changeability is improved because there is no coupling between their components

Price:

- Small time and space overhead

Thanks