

Specification Reuse using Data Refinement in Dafny

Prepared By : M. Asif Saleem

Supervisor : Dr. Rosemary Monahan

National University of Ireland

Maynooth



NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

Arís
Arís



AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



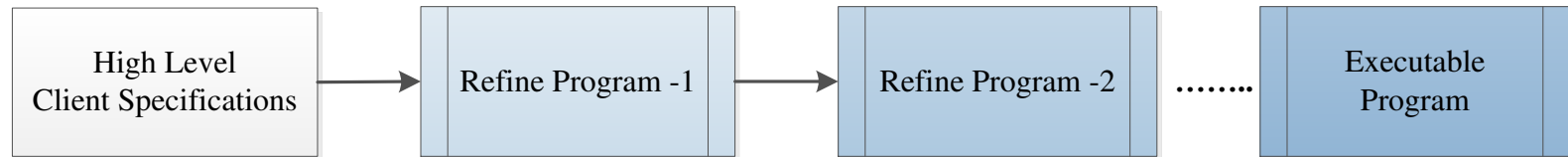
INTRODUCTION

- Specifications
 - Set of client requirements
 - Formally expressed
 - Set theory
- Specifications Reuse
 - Changing the system internally
- Specification languages
 - Dafny, Spec#, JML and Eiffel.

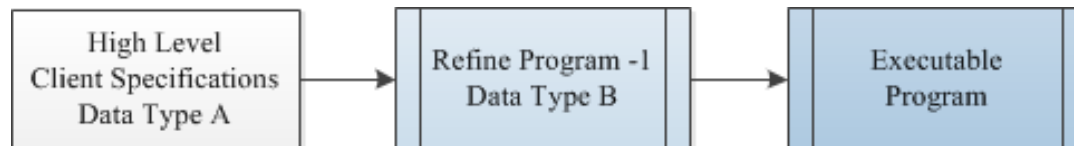


INTRODUCTION (Cont)

- Refinement



- Data Refinement





AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



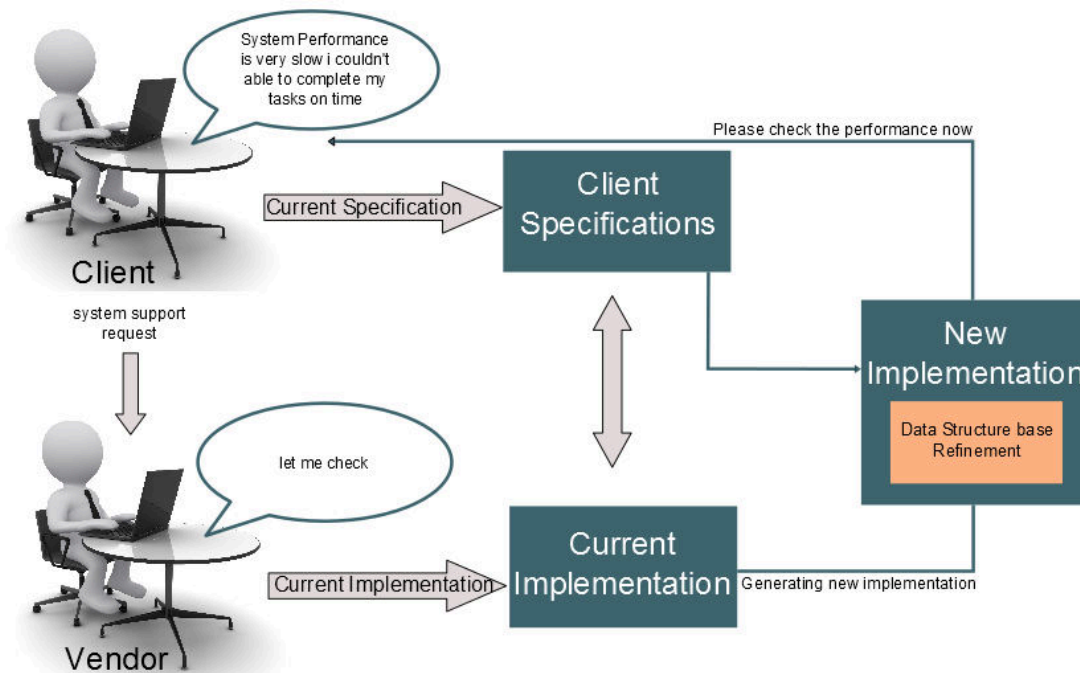
PROBLEM DESCRIPTION

- Scenario's
 - Slow system performance
 - Multiple implementations
 - Regardless of any specific language
 - Differs in data structure
 - Specs in Sets
 - Implementation in arrays for Spec# and in link-list for Dafny



Solution

- Solution
 - Data Structure Based Data Refinement





Solution (Cont)

- Goals of this research
 - Assisting programmers for reusing of specifications
 - Identifying programmers overhead for reusing specification manually
 - Proposing generic refinement framework based on data structure
 - Proof of Concept tool for data structure based Data refinement



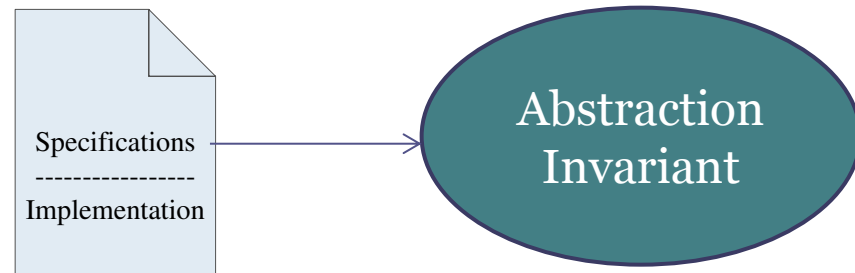
AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS

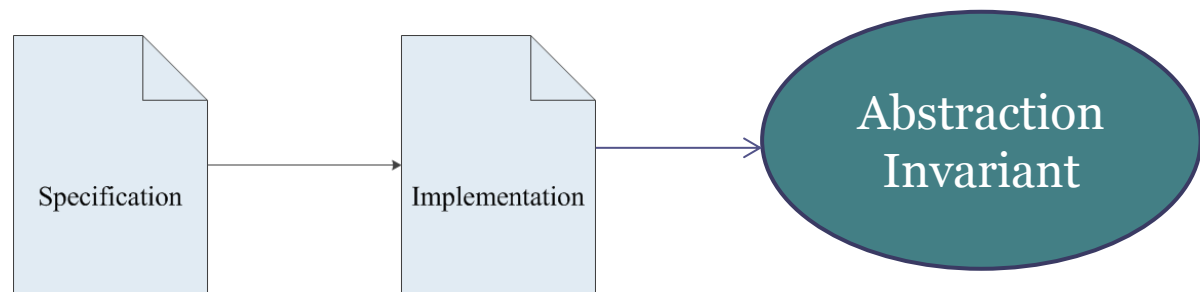


RELATED WORK

- Data Refinement in specification based languages
 - One class approach



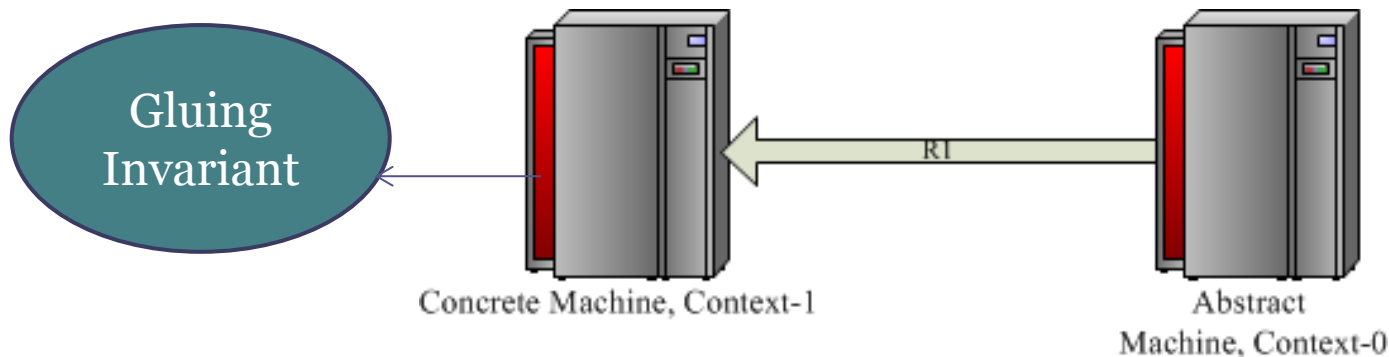
- Two class approach





RELATED WORK (Cont)

- Data Refinement in Event-B
 - A modeling tool
 - Modeling is based on set theory
 - Contexts
 - Machines (Variables, Invariants and Events)





AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



Identifying Programmer Overhead

- Effort Calculation Metric
 - Modified COCOMO model
 - Organic, semi-detached, Embedded

Project Type	a	b	c	d
Organic/Small	3.2	1.05	2.5	0.38

Cost Driver	Rating		
	Low	Moderate	High
Verification attributes			
Verification Skills	1.25	1.00	0.75
Programming Language Knowledge	1.20	1.00	0.67

Cost Driver	Rating
Verification Type	
Manual	1.00
Automatic	0.20



Identifying Programmer Overhead (Cont)

Cost Driver	Rating
Data Structure used	
Sequence	0.05
Arrays	0.10
Link list	0.20
Tree	0.60

- $\text{Effort} = a * (\text{KLOC})^b * \text{EAF}$ [man-months]
 - $\text{Development Time} = c * (\text{KLOC})^d$ [months]
 - $\text{Total Days} = \text{Development Time} * 30$ [days]
 - $\text{Persons Required} = \text{Effort} / \text{Development Time}$
- Effort Adjustment Factor is the **multiplication** of all cost drivers according to the scenario



Identifying Programmer Overhead (Cont)

- High skills in verification and programming with automatic verification. The effort adjustment factor would be
- $EAT = (0.75) * (0.67) * (0.20) * (\text{data structure to be used rating})$

Specification	Data Structures	KLOC	Implementation and Verification effort in term of DT
Sequence/Set	Sequence	500 => 0.5	12 days ←
Sequence/Set	Link-list	500 => 0.5	20 days
Sequence/Set	Arrays	500 => 0.5	15 days ←
Sequence/Set	Tree	500=> 0.5	30 days



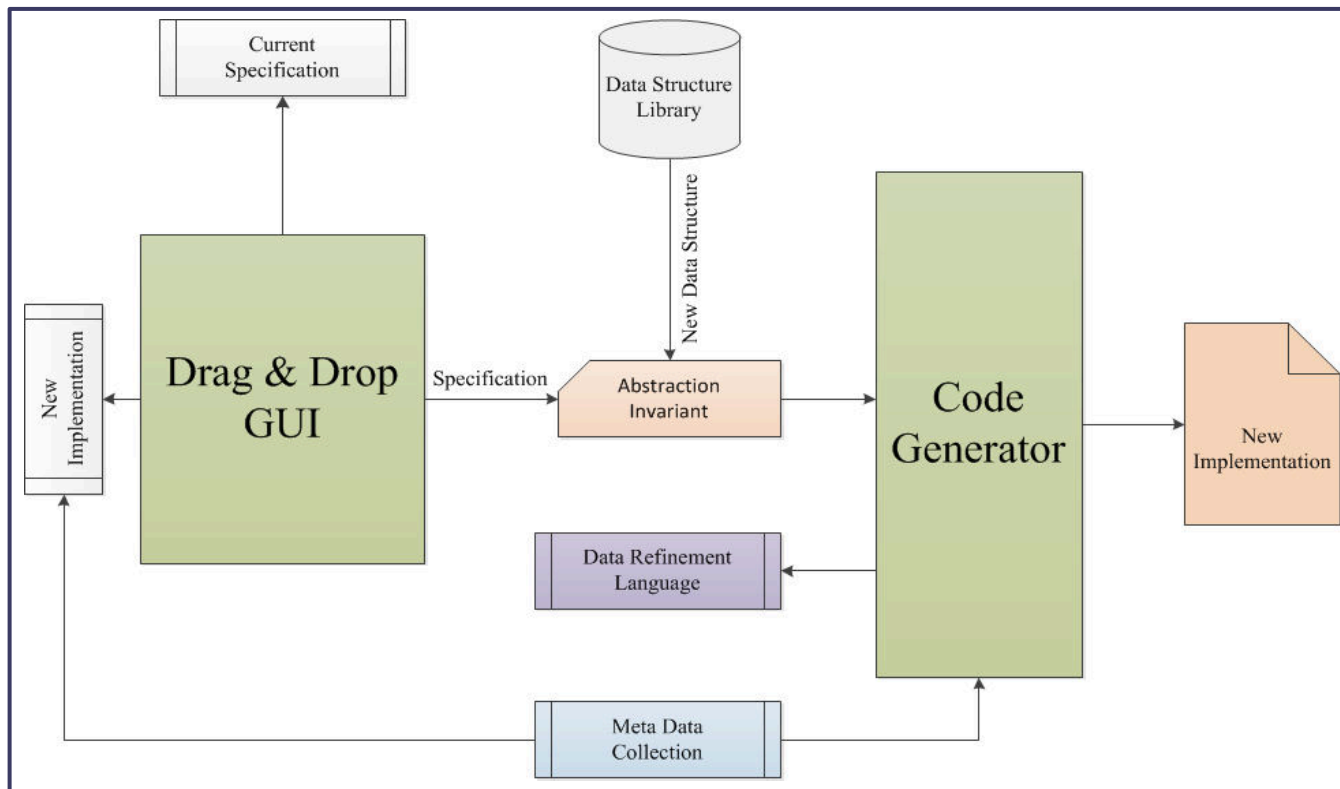
AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



SOLUTION DESIGN

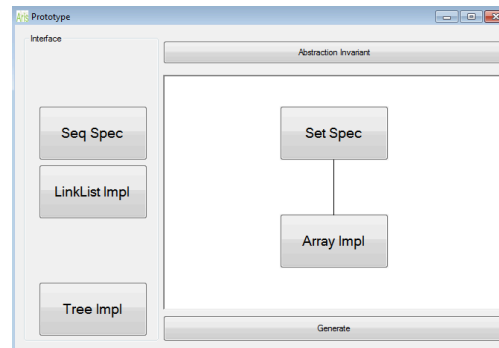
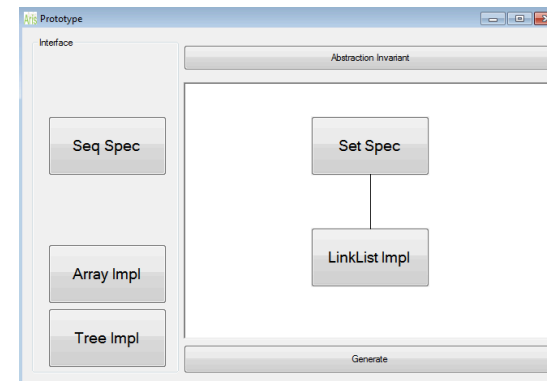
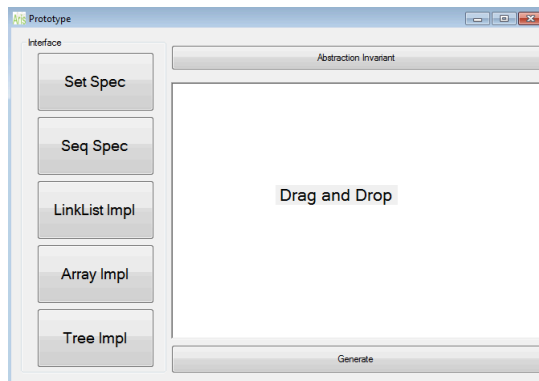
- Proposed Architecture





SOLUTION DESIGN (Cont)

- Interface Design





AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



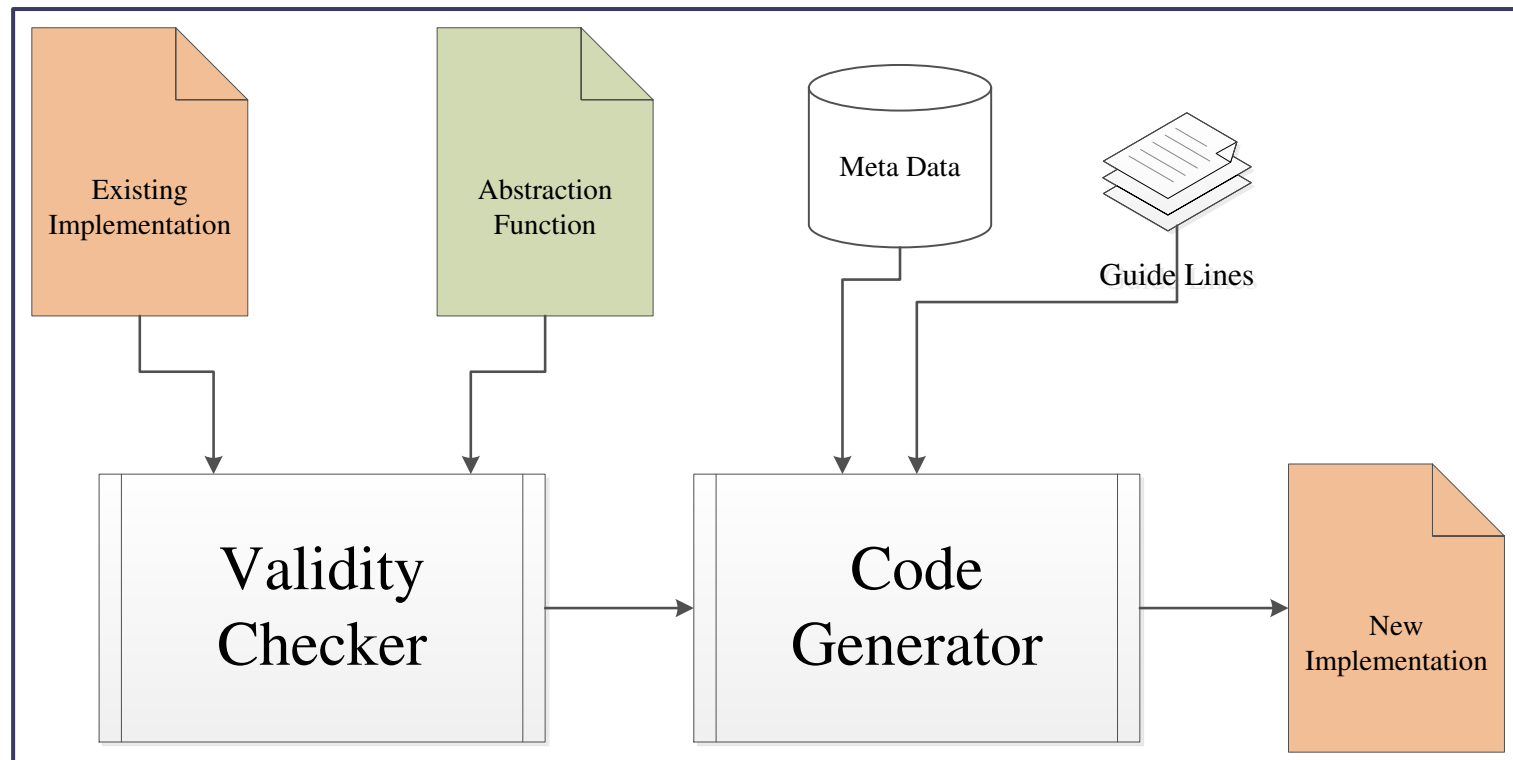
IMPLEMENTATION

- Issues for implementing ideal design
 - Unavailability of external data structure library
 - Lack of external data refinement language
 - Lack of support for meta data collection
- Dafny
 - Support data refinement



IMPLEMENTATION (Cont)

- Specification Reuser





AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



PROOF OF CONCEPT

- Birthday's recording system
 - Add birthday
 - Find birthday
 - Remind birthday

The screenshot shows the 'Specification Reuser' interface. At the top, there are two radio buttons: 'Seq to Array' (checked) and 'Seq to LinkList'. Below this is a diagram showing a 'Specifications' document icon pointing to a sequence of three 'Implementation' cylinders. To the right, there are two boxes: 'Arrays' (green) and 'Link-List' (orange), with vertical ellipses between them. The bottom half of the window is split into two panes: 'Current Implementation' and 'New Implementation', each containing code for a class named 'BBooksequence'.

```
class BBooksequence<NAME(==),DATE(==)> {
  ghost var names_a: seq<NAME>;
  ghost var dates_a: seq<DATE>;
  ghost var Repr: set<object>;

  var nameseq_c: seq<Data<NAME,DATE>>;
  ghost var elems: seq<Data<NAME,DATE>>;

  function Valid(): bool
  reads this, Repr;
  {
    this in Repr &&
    !names_a == !dates_a && !elems == !names_a &&
    !nameseq_c == !elems &&
    (forall i :: 0 <= i && i < names_a ==>
     elems[i] != null &&
     elems[i] in Repr &&
     elems[i].name == names_a[i] &&
     elems[i].date == dates_a[i] &&
     elems[i] == nameseq_c[i])
  }

  method Init()
  modifies this;
  ensures Valid() && fresh(Repr - {this});
  ensures names_a == 0;
  {

```

```
class BBooksequence<NAME(==),DATE(==)> {
  ghost var names_a: seq<NAME>;
  ghost var dates_a: seq<DATE>;
  ghost var Repr: set<object>;

  //var nameseq_c: seq<Data<NAME,DATE>>;
  ghost var elems: seq<Data<NAME,DATE>>;

  function Valid(names_a_c:array
  <NAME> dates_a_c:array<DATE> elems_c:array<Data
  <NAME,DATE>>): bool
  requires names_a_c != null && dates_a_c != null &&
  elems_c != null;
  requires names_a_c.Length == dates_a_c.Length &&
  dates_a_c.Length == elems_c.Length;
  requires names_a_c.Length >= 0 && dates_a_c.Length >
  = 0 && elems_c.Length >= 0;

  reads this, Repr;
  {
    this in Repr &&
    !names_a == !dates_a && !elems == !names_a &&
    !nameseq_c == !elems &&
    (forall i :: 0 <= i && i < names_a ==>
     elems[i] != null &&
     elems[i] in Repr &&
     elems[i].name == names_a[i] &&
```



AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



EVALUATION

- Criteria
 - Automation
 - Verification
 - Effort Reduction



EVALUATION (Cont)

- Evaluation

Specifications	Data Structure	Generation	Comments
Sequence	Arrays	Automatic	Specification is in term of a sequence and implementation is in term of arrays
Sequence	Link List	Automatic	Specifications is in term of a sequence and implementation is in term of link list

- Validation

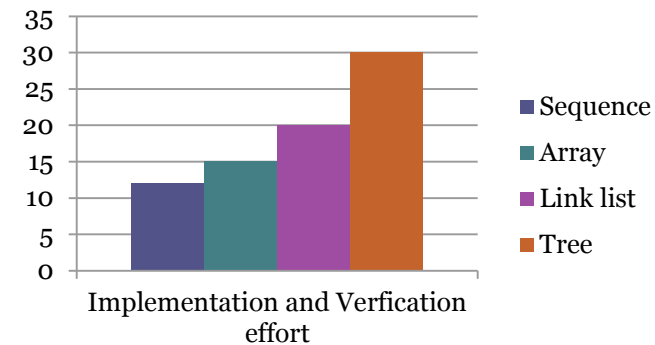
Specifications	Data Structure	Generation	verified
Sequence	Arrays	Automatic	Yes
Sequence	Link List	Automatic	Yes



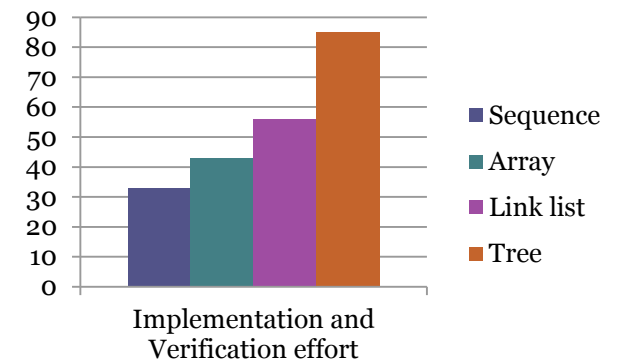
EVALUATION (Cont)

• Results

Specification	Data Structures	KLOC	Implementation and Verification effort in term of DT
Sequence/Set	Sequence	500 => 0.5	12 days
Sequence/Set	Link-list	500 => 0.5	20 days
Sequence/Set	Arrays	500 => 0.5	15 days
Sequence/Set	Tree	500=> 0.5	30 days



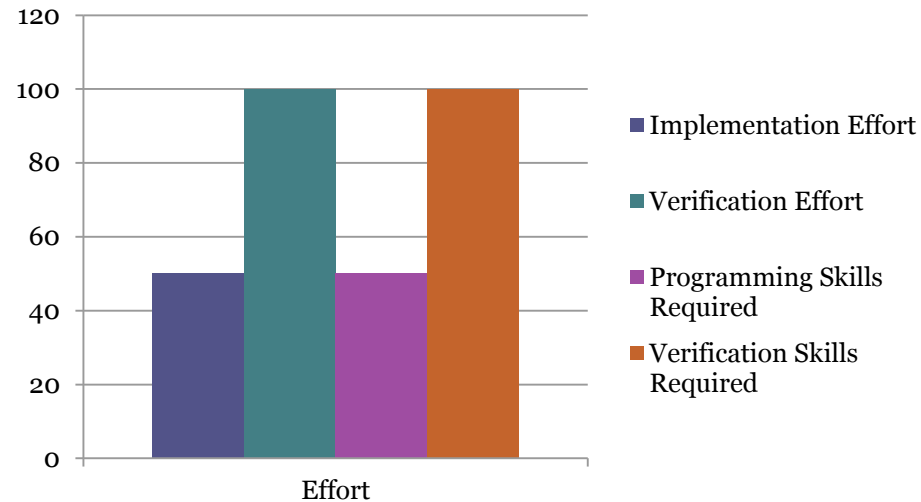
Specification	Data Structures	KLOC	Implementation and Verification effort in term of DT
Sequence/Set	Sequence	500 => 0.5	33 days
Sequence/Set	Link-list	500 => 0.5	56 days
Sequence/Set	Arrays	500 => 0.5	43 days
Sequence/Set	Tree	500=> 0.5	85 days





EVALUATION (Cont)

Specification	Data Structure	KLOC	Implementation effort	Verification effort	Programming Skills	Verification Skills
Sequence	Arrays	Doesn't matter	Low	Moderate	Low	Moderate
Sequence	Link-list	Doesn't matter	Low	Moderate	Low	Moderate





EVALUATION (Cont)

- **Critical Analysis**
 - **Effort Metric**
 - Specific to specification based languages such as Spec# and Dafny
 - Providing rough estimation
 - **Genericity**
 - For small programs : 70-90%



EVALUATION (Cont)

- Verification
 - For small programs : 70-90%



AGENDA

- INTRODUCTION
- PROBLEM STATEMENT/SOLUTION
- RELATED WORK
- IDENTIFYING PROGRAMMER OVERHEAD
- SOLUTION DESIGN
- IMPLEMENTATION
- PROOF OF CONCEPT
- EVALUATION
- CONCLUSIONS



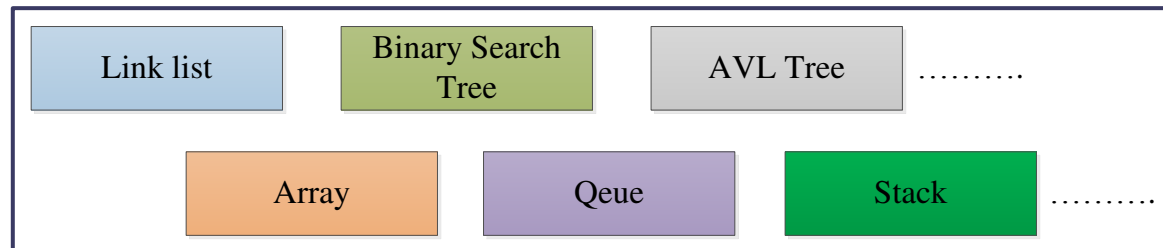
CONCLUSIONS

- **Summary**
 - Proposed generic refinement framework
 - Modify the cost drivers of the COCOMO model
 - Calculated programmer overhead
 - Developed POC tool for data structure based Data Refinement



CONCLUSIONS (Cont)

- Future Work
 - Library development



- Language Development
- Meta Data Collection
- Improved GUI
- Full Danfy tool Implementation



Thanks for Listening!

Q&A