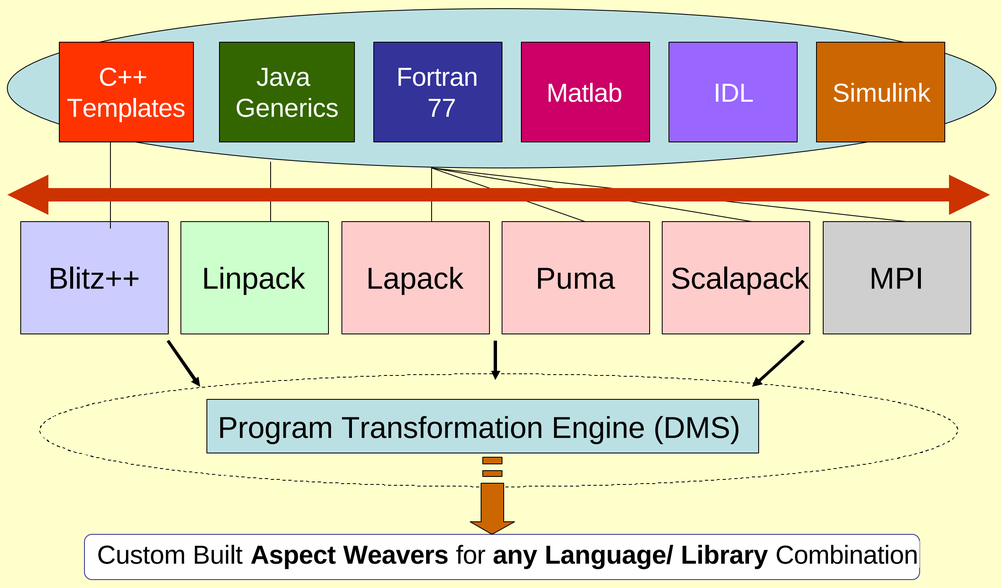
Separating Crosscutting Concerns in Scientific Computing through Program Transformations

Suman Roychoudhury (*roychous@cis.uab.edu*)
Jeff Gray (*gray@cis.uab.edu*)
University of Alabama at Birmingham

Software Composition & Modeling Laboratory
SOFTCOM
Department of Computer and Information Sciences
University of Alabama at Birmingham

Project Objective:

- Support customized aspect weaving for high performance scientific computing applications through scalable parallel program transformation techniques using DMS™
- AST-based source to source transformations
- Current support for aspect weaving into C++ template libraries
- Future directions: Extension to other languages (e.g., FORTRAN, Ada) and libraries (e.g., MPI)



Case Study : C++ Standard Template Library and Blitz++

```
template <class T>
class vector{
   //...
public:
   void push_back(const T& x ) {
      if (finish != end_of_storage) {
         construct(finish, x);
         finish++;
      } else
         insert_aux(end(), x);
   }
  //...
};
```

```
class A {
    vector<int> ai;
    void foo() {
         vector<int> fi1;
         vector<int> fi2;
         vector<float> ff;
         //...
         ai.push_back(1);
         fi1.push_back(2);
         fi2.push_back(3);
         ff.push_back(4.0);
         //...
    }
};

class B {
    vector<char> bc;
    void bar() {
         vector<int> bi;
         vector<float> bf;
         //...
         bc.push_back('a');
         bi.push_back(1);
         bf.push_back(2.0);
         //...
    }
};
```

Designator	Description
C:*	All template instantiations within the declaration of class C, which are not local instantiations in any methods of C
(C:* || * C.*(..):*)	All template instantiations within class C
* C.*(..):*	All local template instantiations within all methods of class C
* C.M(..):*	All local template instantiations within method M of class C
* C.*(..):I	Local template instantiation I, in method M of class C
* C.M(..):I	Any template instantiation that is named I, in all methods of class C

Base Case Implementation + Pointcut Specification

```
template <class T>
class vector_copy {
...
public:
   void push_back(const T& x ) {
      log.add(x);
      if (finish != end_of_storage) {
         construct(finish, x);
         finish++;
      } else
         insert_aux(end(), x); }
   ...
```

```
class A {
   vector<int> ai;
   void foo() {
       vector_copy<int> fi1;
       vector_copy<int> fi2;
       vector<float> ff;
      //...
    }
};
```

```
class B {
 vector<char> bc;
 void bar() {
       vector_copy<int> bi;
       vector<float> bf;
       //...
    }
};
```

```
pointcut push_back_method():
 execution(
 * A.Foo(..):*<-
   vector<int>::push_back(..));
```

```
pointcut push_back_method():
 execution(
  * B.bar(..):bi<-
    vector<int>::push_back(..));
```

Crosscutting in Blitz++

File Name: array_impl.h

```
template<typename T_expr>
        _bz_explicit Array
             (_bz_ArrayExpr<T_expr> expr);
Array(int length0, int length1,
     GeneralArrayStorage<N_rank> storage =
     GeneralArrayStorage<N_rank>())
     : storage_(storage)
{
     BZPRECONDITION(N_rank >= 2);
     // implementation code omitted
     setupStorage(1);
}

Array(int length0, int length1, int length2,
     GeneralArrayStorage<N_rank> storage =
     GeneralArrayStorage<N_rank>())
     : storage_(storage)
{
     BZPRECONDITION(N_rank >= 3);
     // implementation code omitted
     setupStorage(2);
}
…….
```

Before Advice

After Advice

This appears 25 times in the array-impl.h source header, and in 57 places in resize.cc

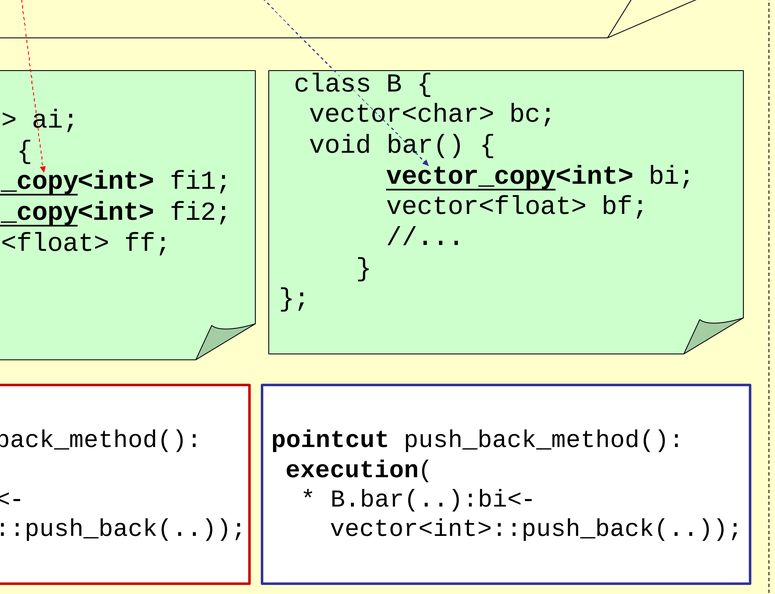
This appears 23 times in both array-impl.h and resize.cc

Aspect Specification

```
aspect InsertBZPreCond_MemAllocation
{
  pointcut ArrayConstuctor():
       execution(Array<*>::Array(..));

  before(): ArrayConstuctor()
    {  BZPRECONDITION(N_rank
         >= tjp.getArgs().length());
    }
  after(): ArrayConstuctor()
    {  setupStorage(
         tjp.getArgs().length()-1);}
    }
}
```

High-Level Architecture



GenAWeave
http://www.cis.uab.edu/gray/Research/GenAWeave