CLOPS: A DSL for Command Line Options

Mikoláš Janota Fintan Fairmichael Viliam Holub
Radu Grigore Julien Charles Dermot Cochran Joe Kiniry

Lero and CASL
University College Dublin
Ireland

DSL WC 2009

SFI grant no. 03/CE2/I303_1

Why are Command Line Options Interesting?

- Many programs start with a small number of options and new ones are added as the project evolves.
- As a consequence, processing of options is coded non-systematically...
- ...resulting in large volumes of code that are difficult to read and to document. (typically hundreds of lines of code)

Why are Command Line Options Interesting?

- Many programs start with a small number of options and new ones are added as the project evolves.
- As a consequence, processing of options is coded non-systematically...
- ...resulting in large volumes of code that are difficult to read and to document. (typically hundreds of lines of code)
- Parsing is difficult because options have various names, formats, values, and dependencies between them e.g.,

```
gzip -h
gzip --help
gzip -v -q
gzip -v -v
gzip -3
tar czv --file archive.tar.gz --append
```

Main Goals for the DSL

- It should be easy to write even small command line descriptions (at the beginning),

Main Goals for the DSL

- It should be easy to write even small command line descriptions (at the beginning),
- and it should be easy to maintain and evolve these.

Main Goals for the DSL

- It should be easy to write even small command line descriptions (at the beginning),
- and it should be easy to maintain and evolve these.
- Writing (sometimes complex) dependencies between options should be straightforward.

Main Goals for the DSL

- It should be easy to write even small command line descriptions (at the beginning),
- and it should be easy to maintain and evolve these.
- Writing (sometimes complex) dependencies between options should be straightforward.
- Generating parser and documentation from the same DSL description.

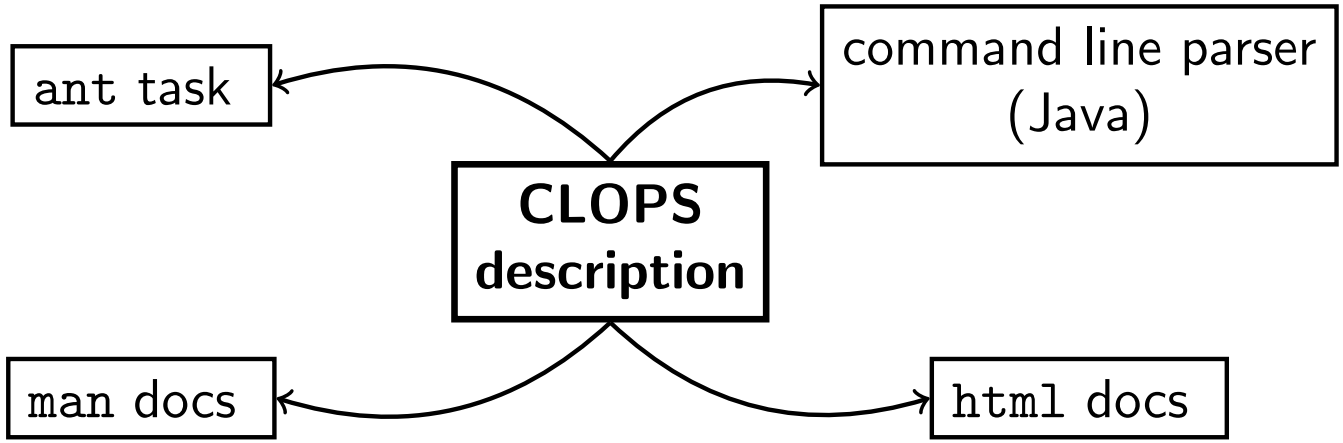
Generating Multiple Artifacts

- Generation is more likely to ensure consistency.
- Generation reduces the amount of work for the user.

CLOPS description

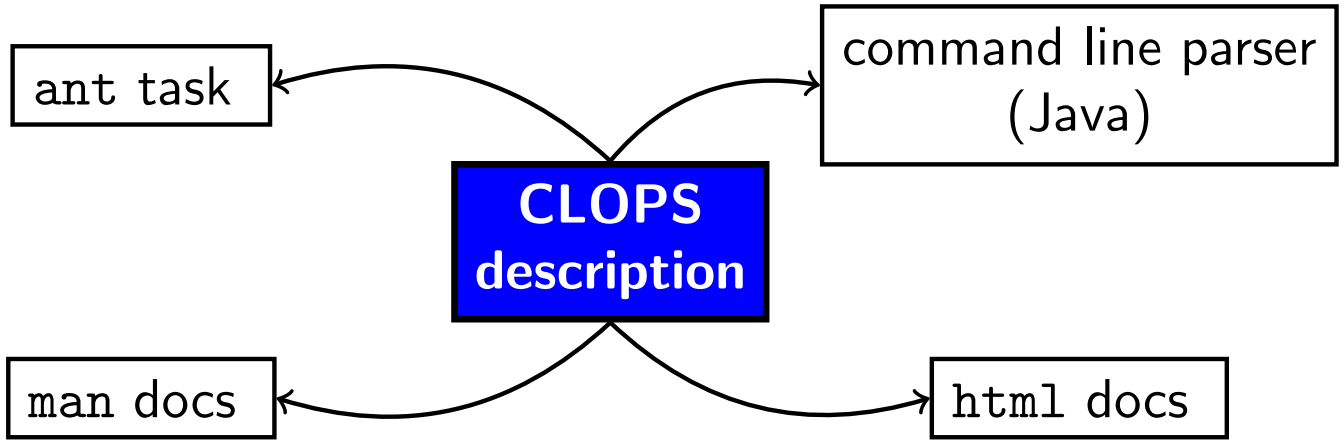
Generating Multiple Artifacts

- Generation is more likely to ensure consistency.
- Generation reduces the amount of work for the user.



Generating Multiple Artifacts

- Generation is more likely to ensure consistency.
- Generation reduces the amount of work for the user.



CLOPS Description Structure

- Option declarations—the set of supported options
- Dynamic rules—how options interact with each other
- Validity conditions—permitted final value combinations
- Format of the command line

CLOPS Description Structure

- Option declarations—the set of supported options

```
quiet    :{"-q","--quiet"}
         :"Suppress all warnings."
suffix   :{"-S","--suffix"}:{string}
         :  "Use suffix xxx instead of .gz."
```

- Dynamic rules—how options interact with each other
- Validity conditions—permitted final value combinations
- Format of the command line

CLOPS Description Structure

- Option declarations—the set of supported options

```
quiet    :{"-q","--quiet"}
         :"Suppress all warnings."
suffix   :{"-S","--suffix"}:{string}
         :  "Use suffix xxx instead of .gz."
```

- Dynamic rules—how options interact with each other

```
trigger   { condition }    →   effect
quiet     { quiet }        →   verbose:=0;
verbose   { verbose > 0 }  →   quiet:=false;
```

- Validity conditions—permitted final value combinations
- Format of the command line

CLOPS Description Structure

- Option declarations—the set of supported options

```
quiet    :{"-q","--quiet"}
         :"Suppress all warnings."
suffix   :{"-S","--suffix"}:{string}
         :  "Use suffix xxx instead of .gz."
```

- Dynamic rules—how options interact with each other

trigger	{ *condition* }	→	*effect*
quiet	{ quiet }	→	verbose:=0;
verbose	{ verbose > 0 }	→	quiet:=false;

- Validity conditions—permitted final value combinations

```
quiet==false || verbose==0
```

- Format of the command line

CLOPS Description Structure

- Option declarations—the set of supported options

```
quiet    :{"-q","--quiet"}
         :"Suppress all warnings."
suffix   :{"-S","--suffix"}:{string}
         :  "Use suffix xxx instead of .gz."
```

- Dynamic rules—how options interact with each other

trigger	{ *condition* }	→	*effect*
quiet	{ quiet }	→	verbose:=0;
verbose	{ verbose > 0 }	→	quiet:=false;

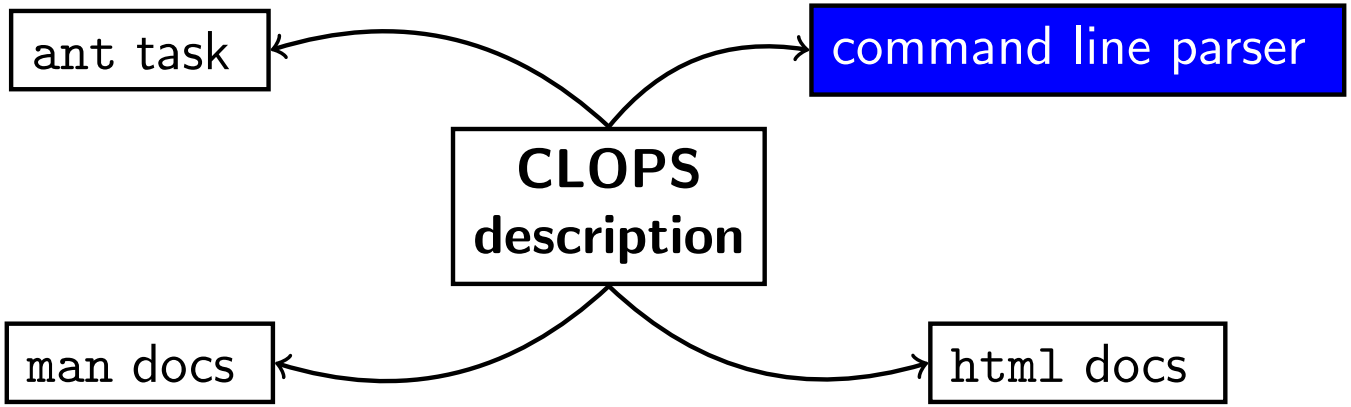
- Validity conditions—permitted final value combinations

```
quiet==false || verbose==0
```

- Format of the command line

```
options* filename
```

Talk Progress



Order of Input Strings Matters

- The order might determine rôles of strings, e.g.,

```
svn add add
```

Order of Input Strings Matters

- The order might determine rôles of strings, e.g.,

```
svn add add
```

- The format lets us specify such, e.g.,

```
option* command filename*
```

Order of Input Strings Matters

- The order might determine rôles of strings, e.g.,

```
svn add add
```

- The format lets us specify such, e.g.,

```
option* command filename*
```

- Another example with files

```
rm -y       // unknown option
rm *        // dangerous if there's a file -rf
rm -- -y    // removes the file -y
```

Order of Input Strings Matters

- The order might determine rôles of strings, e.g.,

```
svn add add
```

- The format lets us specify such, e.g.,

```
option* command filename*
```

- Another example with files

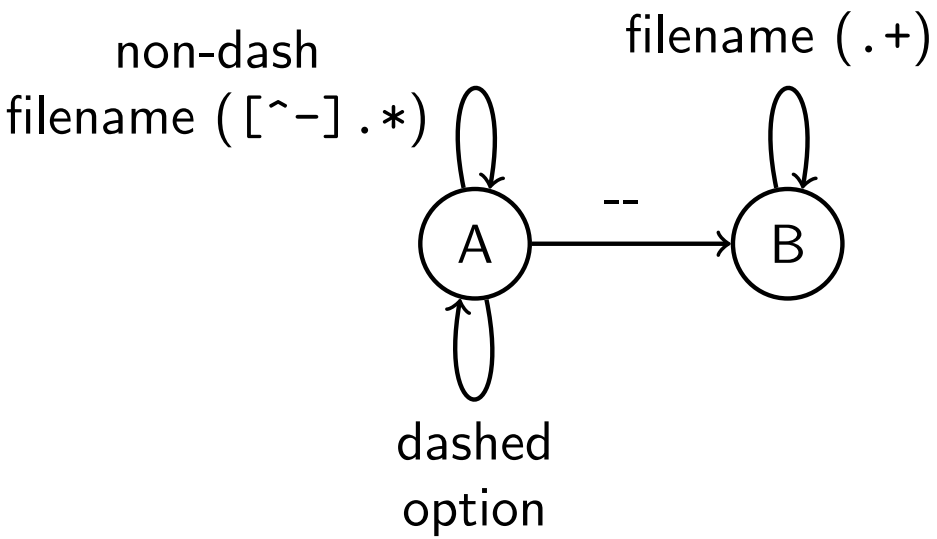
```
rm -y       // unknown option
rm *        // dangerous if there's a file -rf
rm -- -y    // removes the file -y
```

- CLOPS solution

```
(Option | Files)* (DashDash DashFiles*)?
```

The Command line Format Drives Parsing

- An automaton is constructed according to the format of the command-line, where edges are labeled with options.
- If the automaton finishes in a terminal state then the parse of the command line was valid according to the format.



Parsing Individual Options

- The command line is presented to the parser as one string.

```
cp␣src␣dest␣
```

Parsing Individual Options

- The command line is presented to the parser as one string.

```
cp␣src␣dest␣
```

- Each option is responsible for parsing a piece of the string, from the current parse index, including ␣

Parsing Individual Options

- The command line is presented to the parser as one string.

```
cp␣src␣dest␣
```

- Each option is responsible for parsing a piece of the string, from the current parse index, including ␣
- This is necessary to handle

```
tar xf filename
```

Parsing Individual Options

- The command line is presented to the parser as one string.

```
cp␣src␣dest␣
```

- Each option is responsible for parsing a piece of the string, from the current parse index, including `␣`
- This is necessary to handle

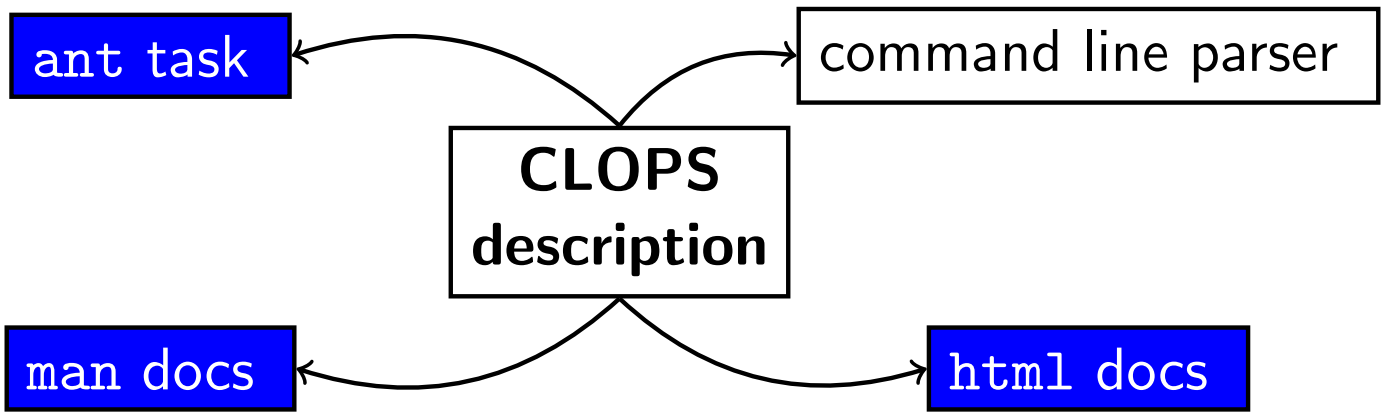
```
tar xf filename
```

- We don't need to specificy this most of the time.

```
add:{"add"}
```

  parses on `add␣`

Talk Progress



Documentation Generation

- Using the apache template tool `velocity`.

```
#foreach($option in $info.getOptionDescriptions())

Name:  $option.Identifier
Description:  $!option.Description
#end
```

Documentation Generation

- Using the apache template tool `velocity`.

```
#foreach($option in $info.getOptionDescriptions())

Name:  $option.Identifier
Description:  $!option.Description
#end
```

- High flexibility, but a little crude.
- At the moment we don't have a nice way of rendering argument types and rules.

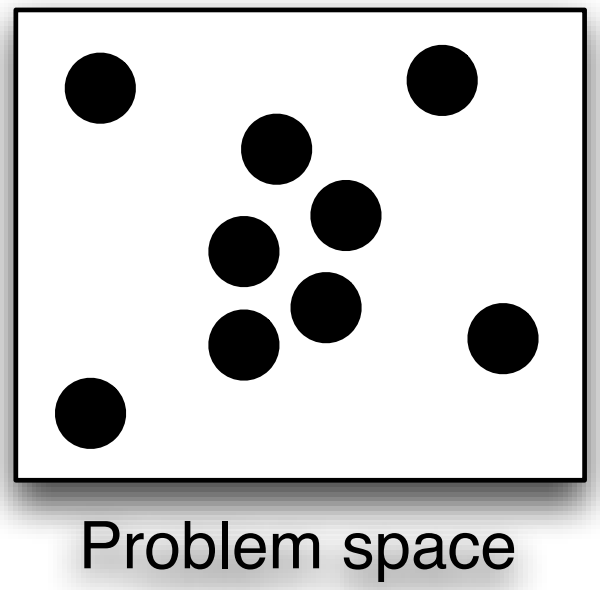
Talk Progress

How-it-all-started story

Variability in Software

- Believe it or not, this work has been inspired by *Software Product Lines*

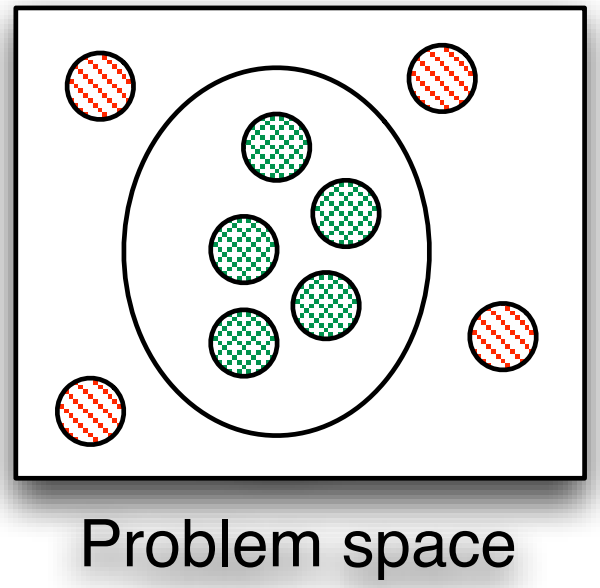
Domain



Variability in Software

- Believe it or not, this work has been inspired by *Software Product Lines*

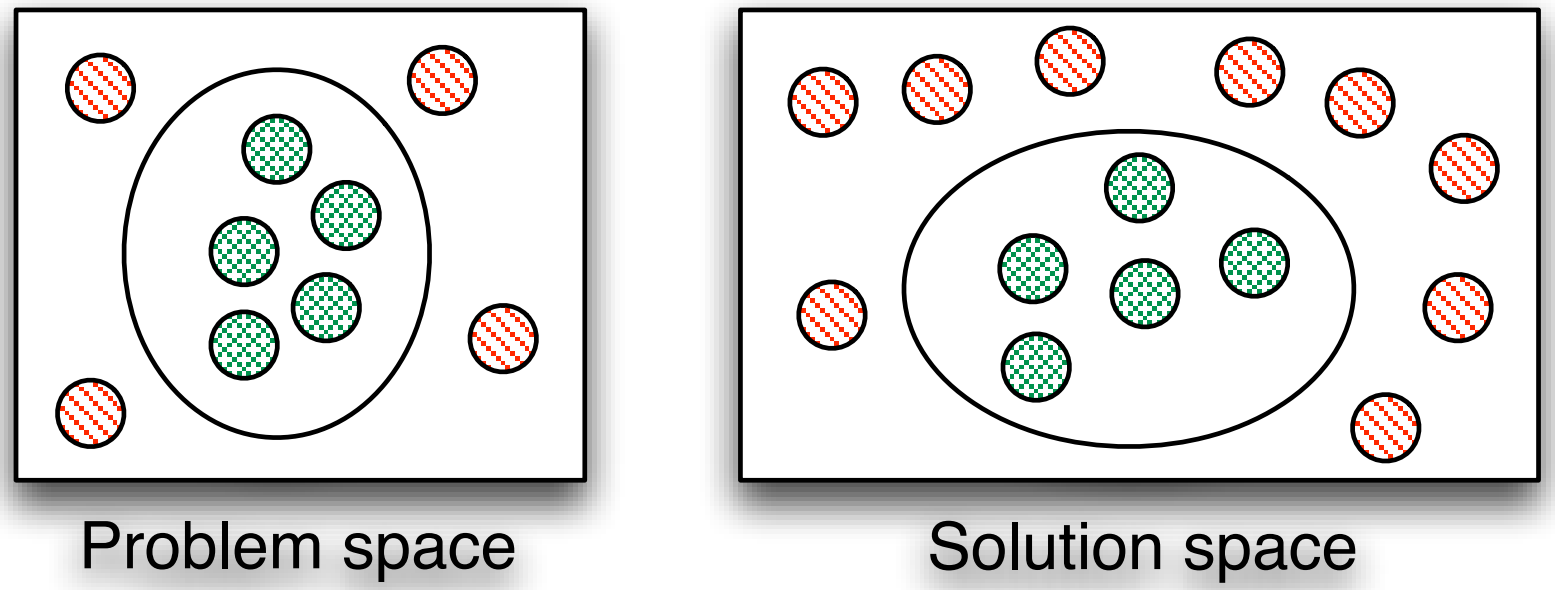
Domain model



Variability in Software

- Believe it or not, this work has been inspired by *Software Product Lines*

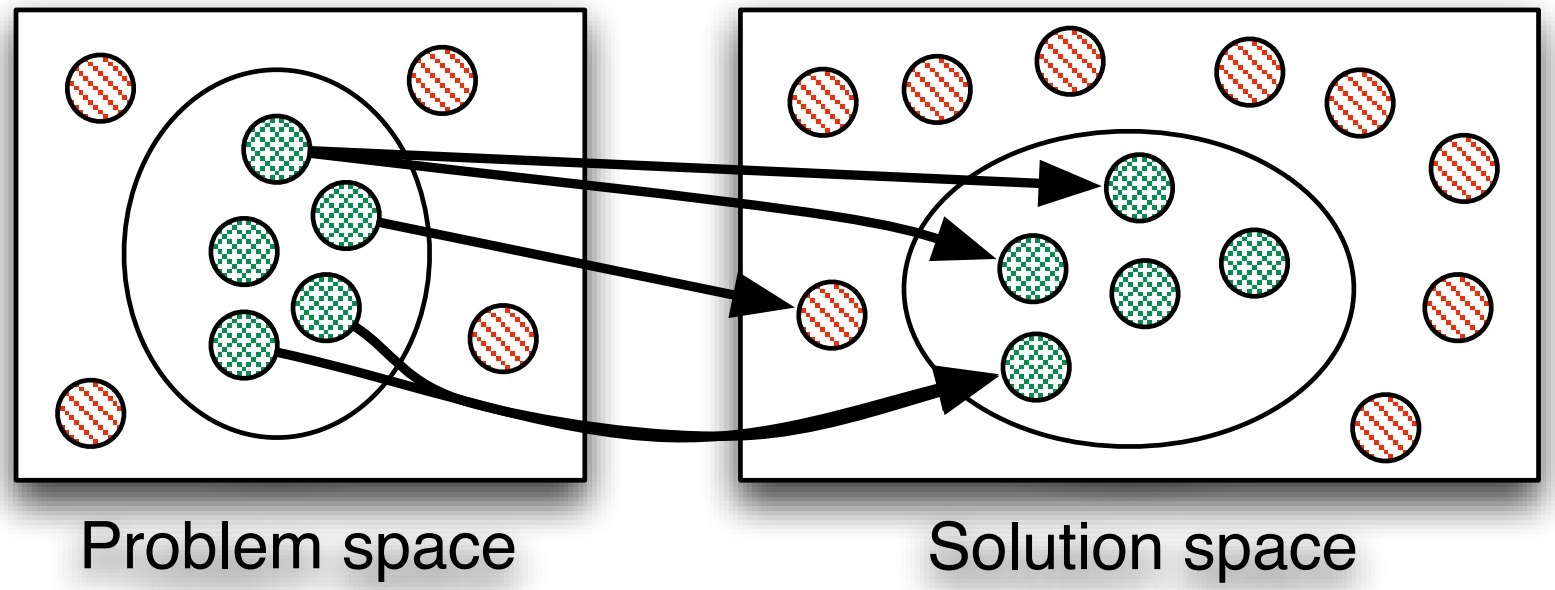
Domain and Solution model



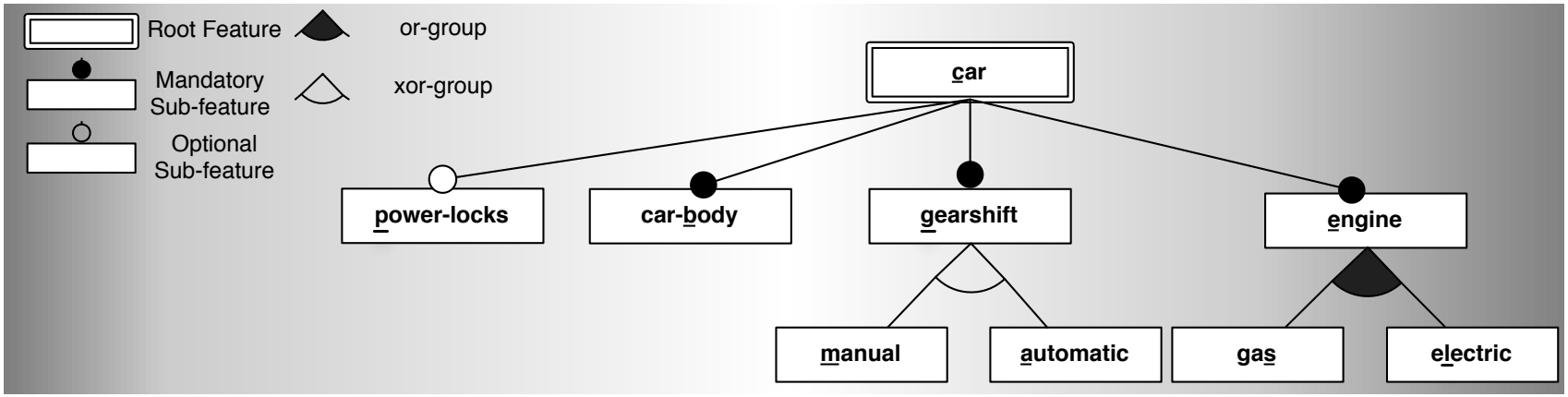
Variability in Software

- Believe it or not, this work has been inspired by *Software Product Lines*

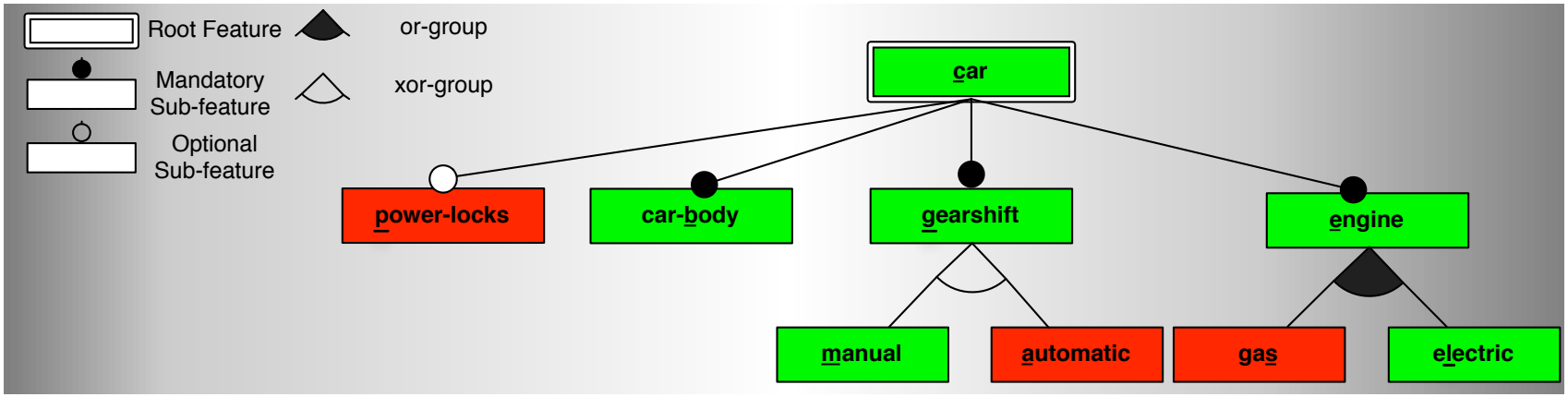
Domain and Solution model combined



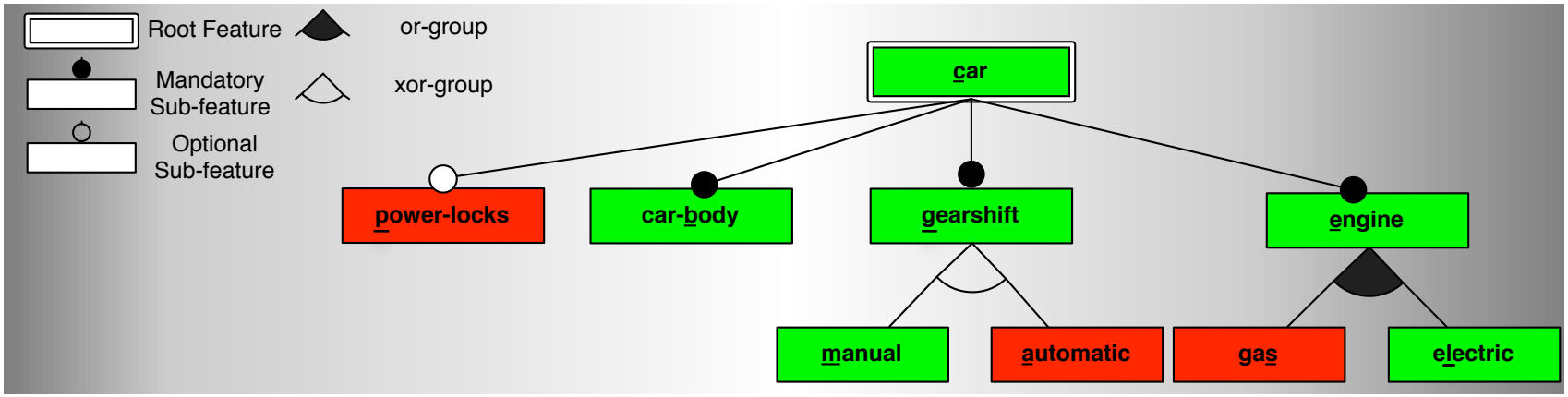
Variability Modeling



Variability Modeling



Variability Modeling



- Writing command line options can be seen as a product derivation.

```
alias ll='ls -l'
```

Usage

- Carried out several experiments (`svn`, `gzip`, `ls`, ...)
- Released on `source forge`
  `http://clops.sourceforge.net/`
- At least one commercial user
- Used inside our group for a number of open-source projects

Future Work and Challenges

- Tab-completion
- GUI generation
- More static checking
- Improve documentation
- Port to other languages — a general rule engine would be desirable

```
http://clops.sourceforge.net/
```