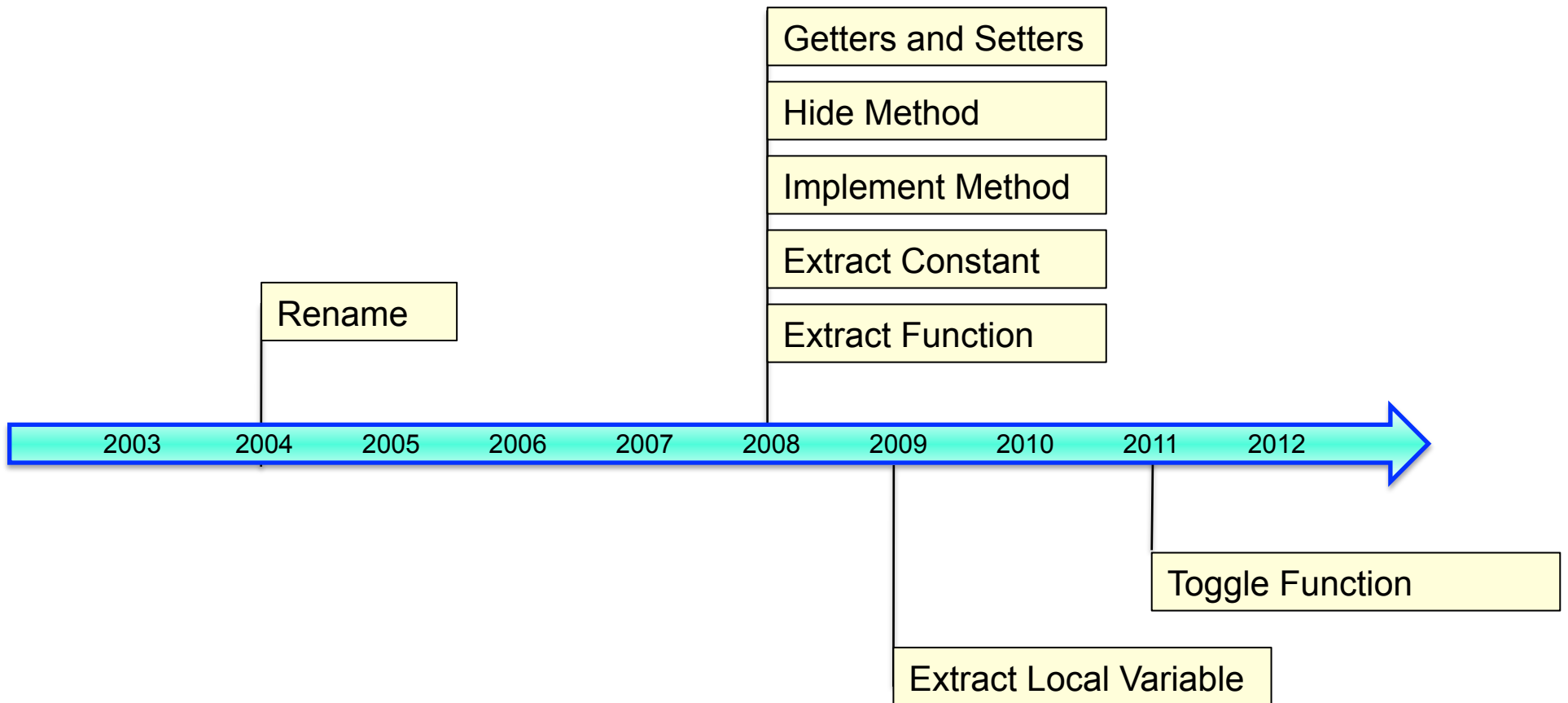


C++ Refactoring – Now for Real


Sergey Prigogin
Google

CDT committer, refactoring component lead

History of refactoring in CDT



What's wrong with refactoring in CDT?

- Corrupts surrounding code
 - Generated code is not formatted properly
 - Often produces semantically invalid code
 - Does it very slowly
 - Requires you to save all files
- 
- A decorative footer at the bottom of the slide features a horizontal line above several overlapping, semi-transparent circles in shades of green, blue, pink, and yellow.

Why was it slow?

- Refactoring requires AST
- AST requires index read lock
- Refactoring steps:
 - checkInitialConditions
 - checkFinalConditions
 - createChange

How we made it fast

- Disposable refactoring context object
- Caching of ASTs
- Using editor AST when available
 - Index doesn't help with dirty editors

Preventing collateral damage

- AST rewrite infrastructure
- Creation of textual changes based on AST delta



Passing parameters by value or reference

```
class A {  
public:  
    void setIntField(int intField) {  
        this->intField = intField;  
    }  
  
    void setStrField(const std::string& strField) {  
        this->strField = strField;  
    }  
  
private:  
    int intField;  
    std::string strField;  
};
```

Which variables should be returned

```
int factorial(int k) {  
    int i = 1;  
    int f = 1;  
    while (i <= k)  
        f *= i++;  
    return f;  
}
```

```
int multiply(int f, int* i) {  
    f *= *i++;  
    return f;  
}
```

```
int factorial(int k) {  
    int i = 1;  
    int f = 1;  
    while (i <= k)  
        f = multiply(f, &i);  
    return f;  
}
```

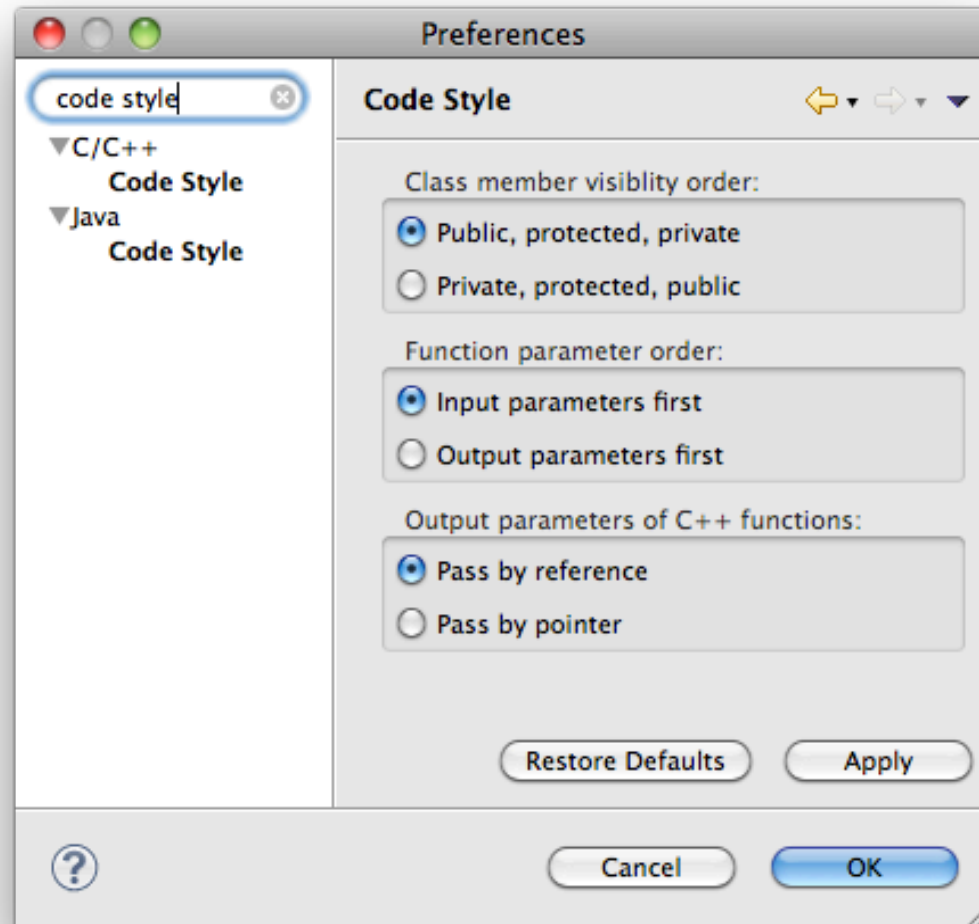

How is was in CDT 8.0

```
int factorial(int k) {  
    int i = 1;  
    int f = 1;  
    while (i <= k)  
        f *= i++;  
    return f;  
}
```

```
int multiply(int f, int i)  
{  
    f *= i++;  
}
```

```
int factorial(int k) {  
    int i = 1;  
    int f = 1;  
    while (i <= k)  
        f = multiply(f, i);  
    return f;  
}
```


New code style preferences



How can I try it myself?

Download <https://hudson.eclipse.org/hudson/job/cdt-nightly/lastSuccessfulBuild/artifact/releeng/org.eclipse.cdt.repo/target/org.eclipse.cdt.repo.zip>

What's next?

- Focus on quality
 - *Change Function Signature and Inline* refactorings
 - Rename file when renaming a class
 - *Organize Includes*
-
- 
- A decorative footer at the bottom of the slide featuring a horizontal line and several overlapping, semi-transparent circles in shades of green, blue, pink, and yellow.