PerConIK

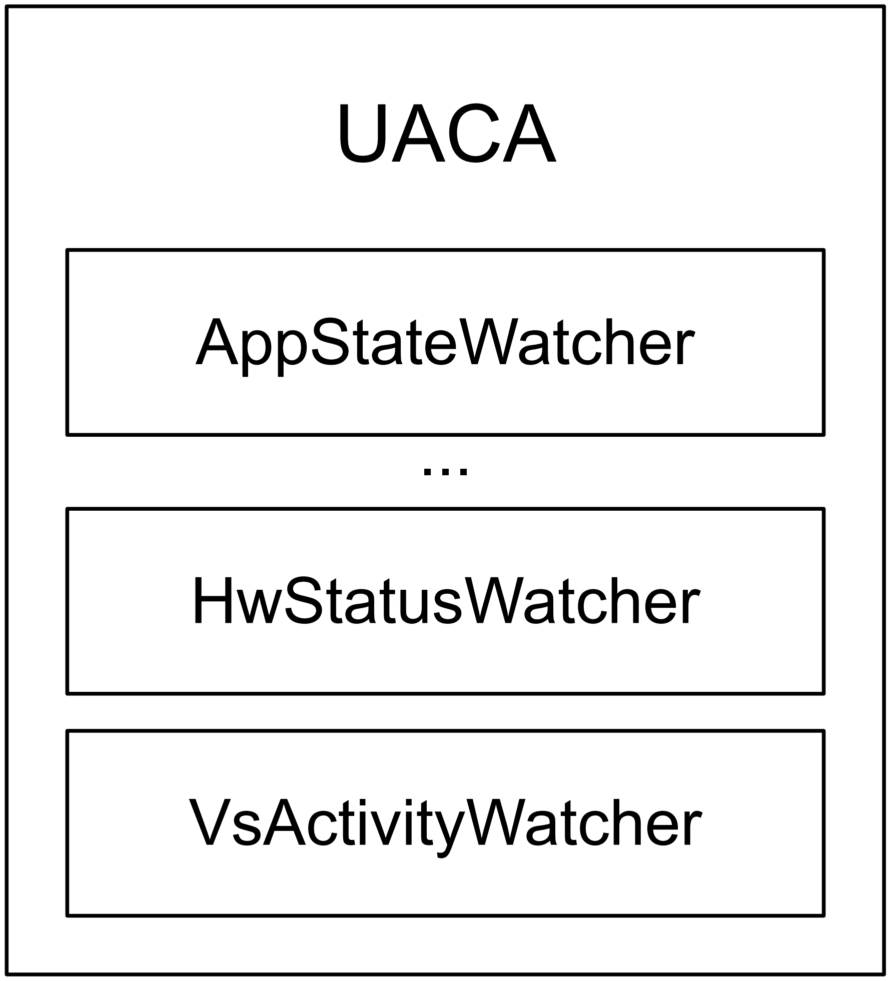
Monitoring Programmer's Activity by means of an Extension to Eclipse

@pavolzbell

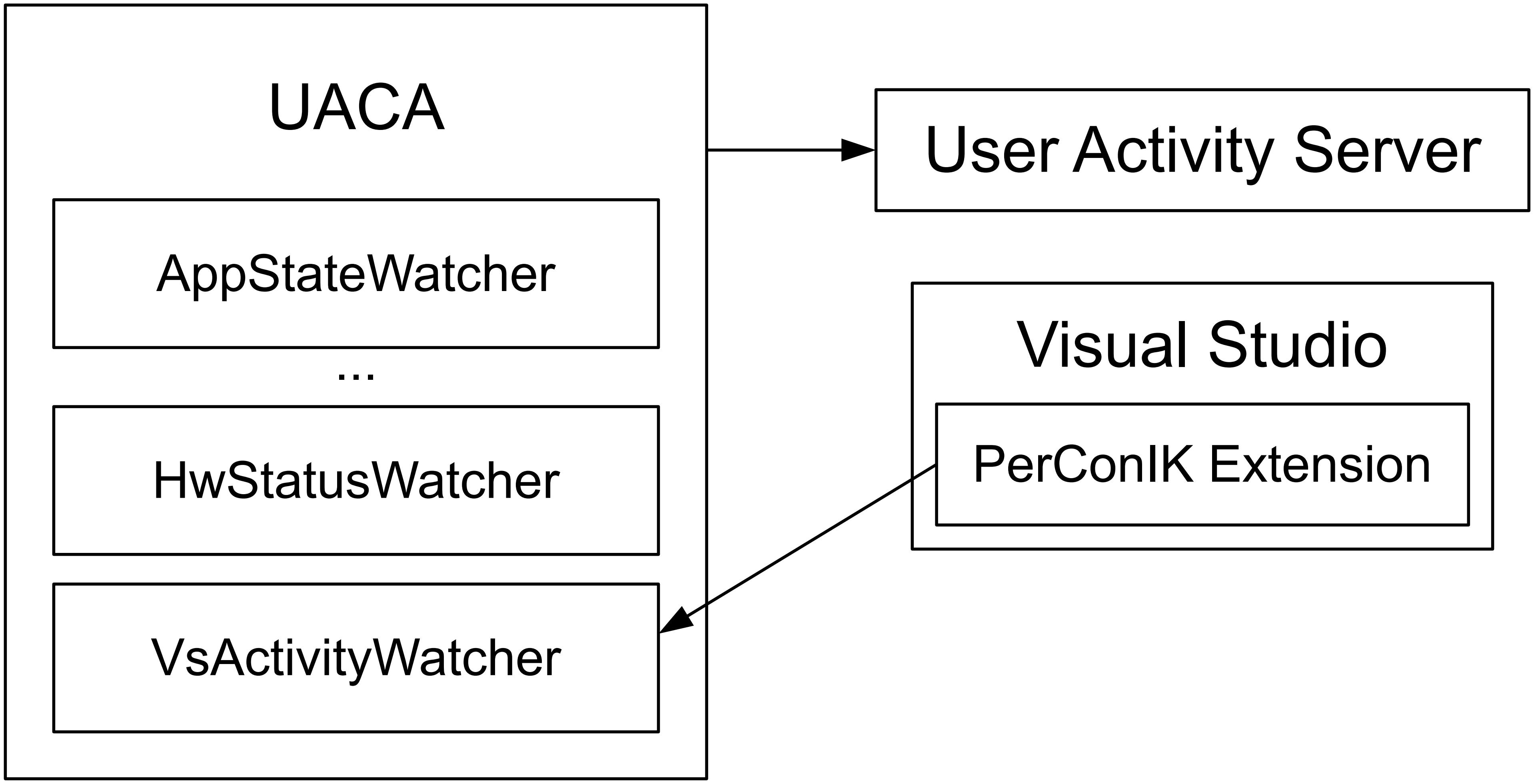
Project PerConIK

Research of methods for acquisition, analysis and personalized conveying of information and knowledge

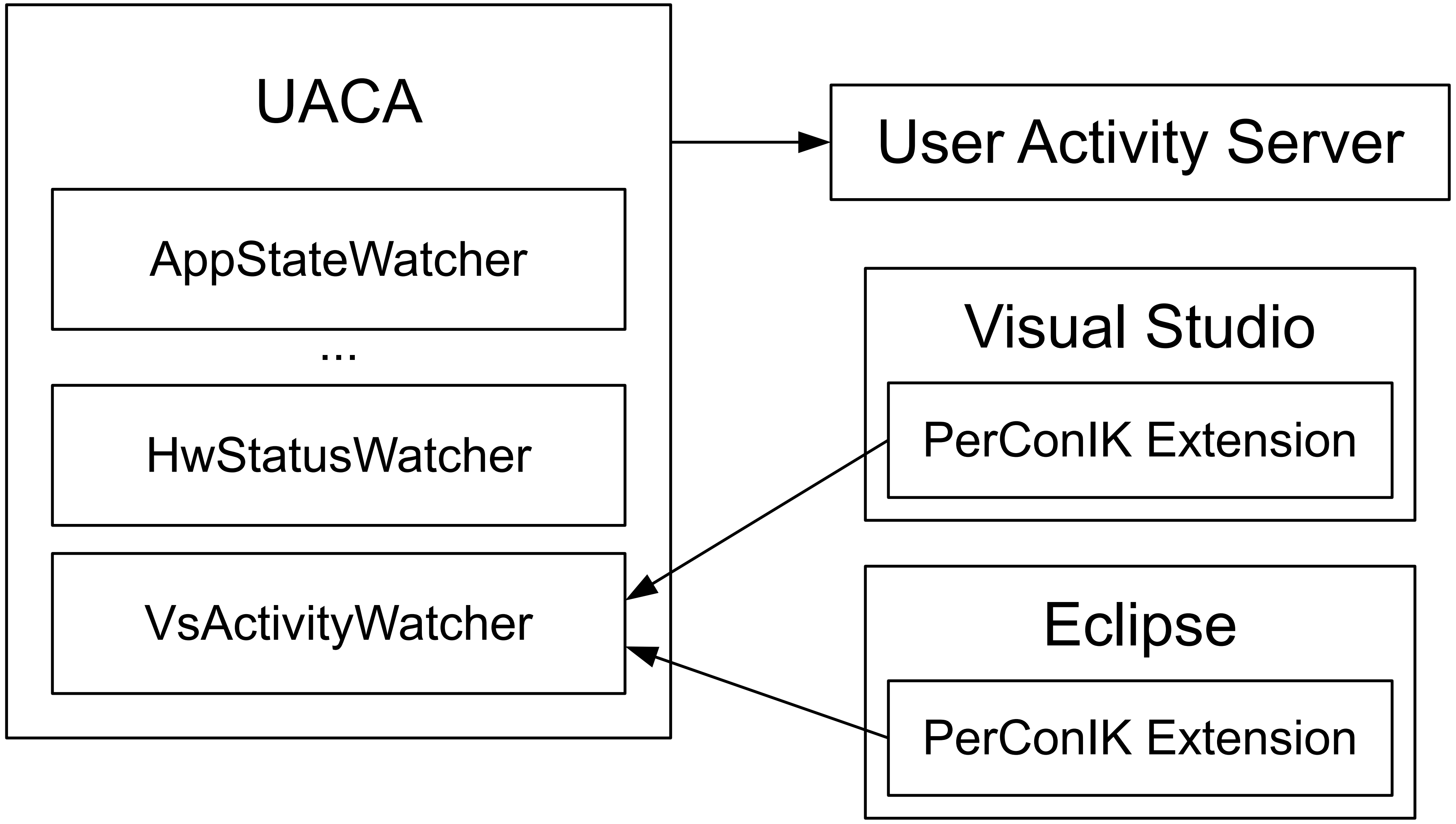
Overview


UACA
AppStateWatcher
...
HwStatusWatcher
VsActivityWatcher

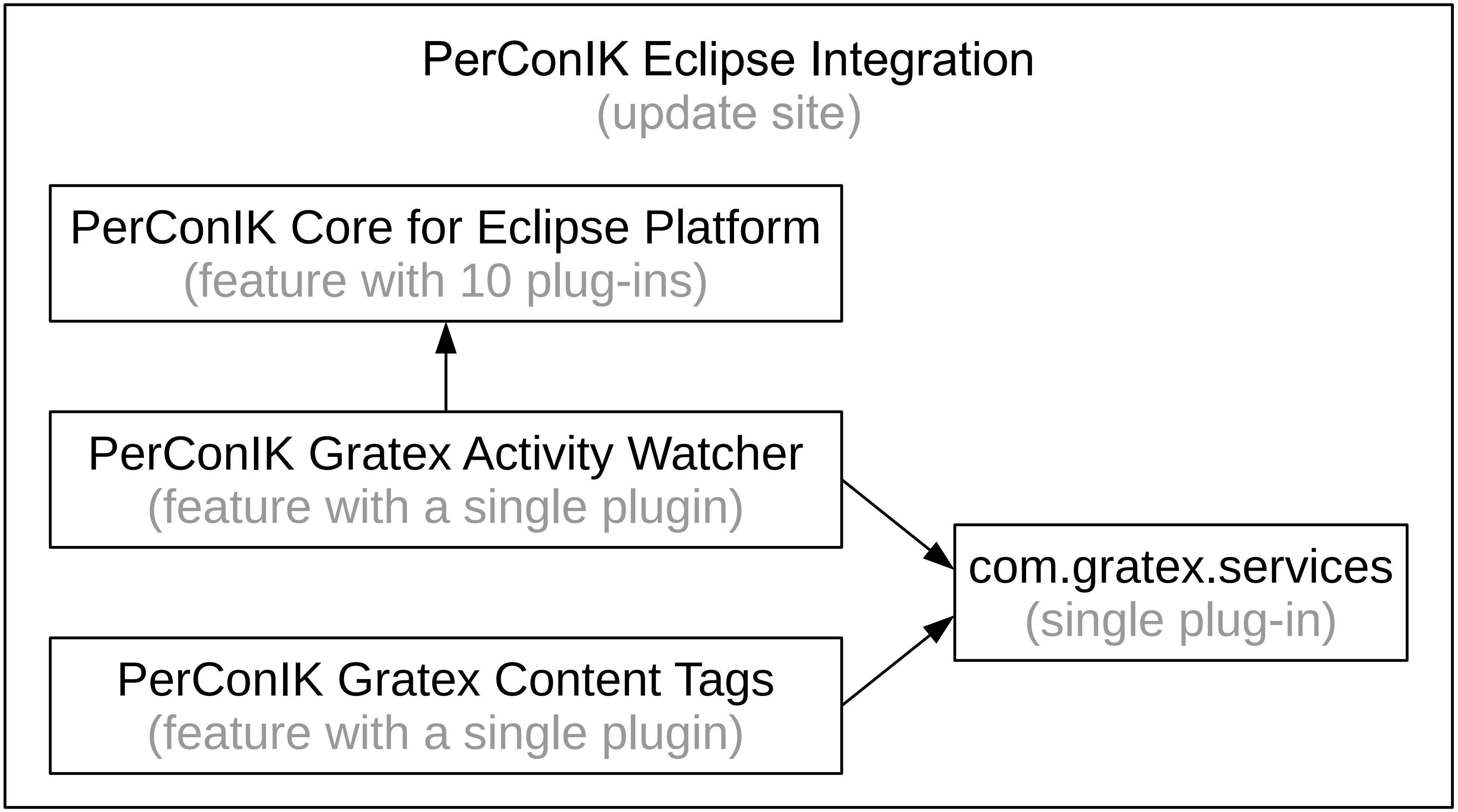
Overview


UACA
AppStateWatcher
...
HwStatusWatcher
VsActivityWatcher
User Activity Server
Visual Studio
PerConIK Extension

Overview


UACA
AppStateWatcher
...
HwStatusWatcher
VsActivityWatcher
User Activity Server
Visual Studio
PerConIK Extension
Eclipse
PerConIK Extension

Eclipse Features and Plug-ins


PerConIK Eclipse Integration
(update site)
PerConIK Core for Eclipse Platform
(feature with 10 plug-ins)
PerConIK Gratex Activity Watcher
(feature with a single plugin)
PerConIK Gratex Content Tags
(feature with a single plugin)
com.gratex.services
(single plug-in)

Core Features

- Support for 30 most used Eclipse listener interfaces
- Automatic listener management via plug-in ext.

  (automatic registration, startup & shutdown hooks, and more)
- Synchronized with UI threads
- Configurable at runtime
- Listener serialization and annotations
- Easy to use, both directly or via plug-in ext.

- Javadocs (http://perconik.github.io/docs)

Core Listeners

- package sk.stuba.fiit.perconik.core.listeners
- Java Elements (AST node changes), Git (repository changes)
- Resources (projects, packages, directories, files)
- Documents and File Buffers (low level, SWT like)
- Launches (run, debug), Refactorings (execution, history)
- Commands (execution), Operations (history)
- Selections (text, tree structure, marks), Search (files)
- Completitions, Debug Events, Test Runs, …
- Workbench, Windows, Pages, Parts, Perspectives

Core Plug-ins

- sk.stuba.fiit.perconik.core

  (listener interfaces, adapters and management in general)
- sk.stuba.fiit.perconik.core.java

  (AST APIs: transformations, filters, tokenization, difference, …)

- sk.stuba.fiit.perconik.eclipse (extensions to Eclipse APIs)
- sk.stuba.fiit.perconik.utilities (strings, enums, reflection)

- sk.stuba.fiit.perconik.{core.persistence, debug, environment, libraries, preferences, ui}

Core Libraries

- sk.stuba.fiit.perconik.libraries

- ANTLR 4.1 (ANother Tool for Language Recognition)
- java-diff-utils 1.2.1 (computing diffs, applying patches, …)
- Guava 15.0 (Google's libraries: collections, concurrency, …)
- JSR-305 2.0.1 (Annotations for Software Defect Detection)
- INTT (Identifier Name Tokenisation Tool)

Activity Watcher

- com.gratex.perconik.activity
- Reference implementation of core listener interfaces
- Listeners:
  - IdeCodeListener (code selections)
  - IdeCommitListener (commits using Git)
  - IdeDocumentListener (add, open, rename, save, switch to, …)
  - IdeProjectListener (add, open, refresh, switch to, …)
  - IdeStateListener (state changes)
  - IdeElementListener and IdeFindListener (not implemented yet)
- Implemented at UISI FIIT STU

Content Tags

- com.gratex.perconik.tag
- Update of the legacy ConMark plug-in
- Implemented at Gratex International

Services

- com.gratex.perconik.services
  - ActivitySvc
  - AstRcsWcfSvc
  - ITMService
  - TagProfileWcfSvc
  - VsActivityWatcherService (local to UACA)
- No need to generate DTOs again, reuse this plug-in!
  - Remember the PermGen, -XX:PermSize -XX:MaxPermSize
- Implemented at Gratex International

Under the Hood

- Class vs. instance based listener registering
- Listener registration hooks
- Resources – Low level listener registration handlers
- Dynamic singleton instance resolving
- Core Services (high level listener or resource management)

Class vs. Instance Based Listeners

Classic listener registration problem:

- Class based

```
JavaCore.addElementChangedListener(listener);
```

- Instance based

```
IDocument document = ...;
document.addDocumentListener(listener);
```

Class vs. Instance Based Listeners

Classic listener registration solution:

sk.stuba.fiit.perconik.{

debug.listeners.DocumentDebugListener@63faaf8b

core.resources.DocumentHook$WindowHook for debug.listeners.DocumentDebugListener@63faaf8b

core.resources.PartHook$PageHook for core.resources.EditorHandler$PartFilter for core.resources.DocumentHook$WindowHook for debug.listeners.DocumentDebugListener@63faaf8b

}

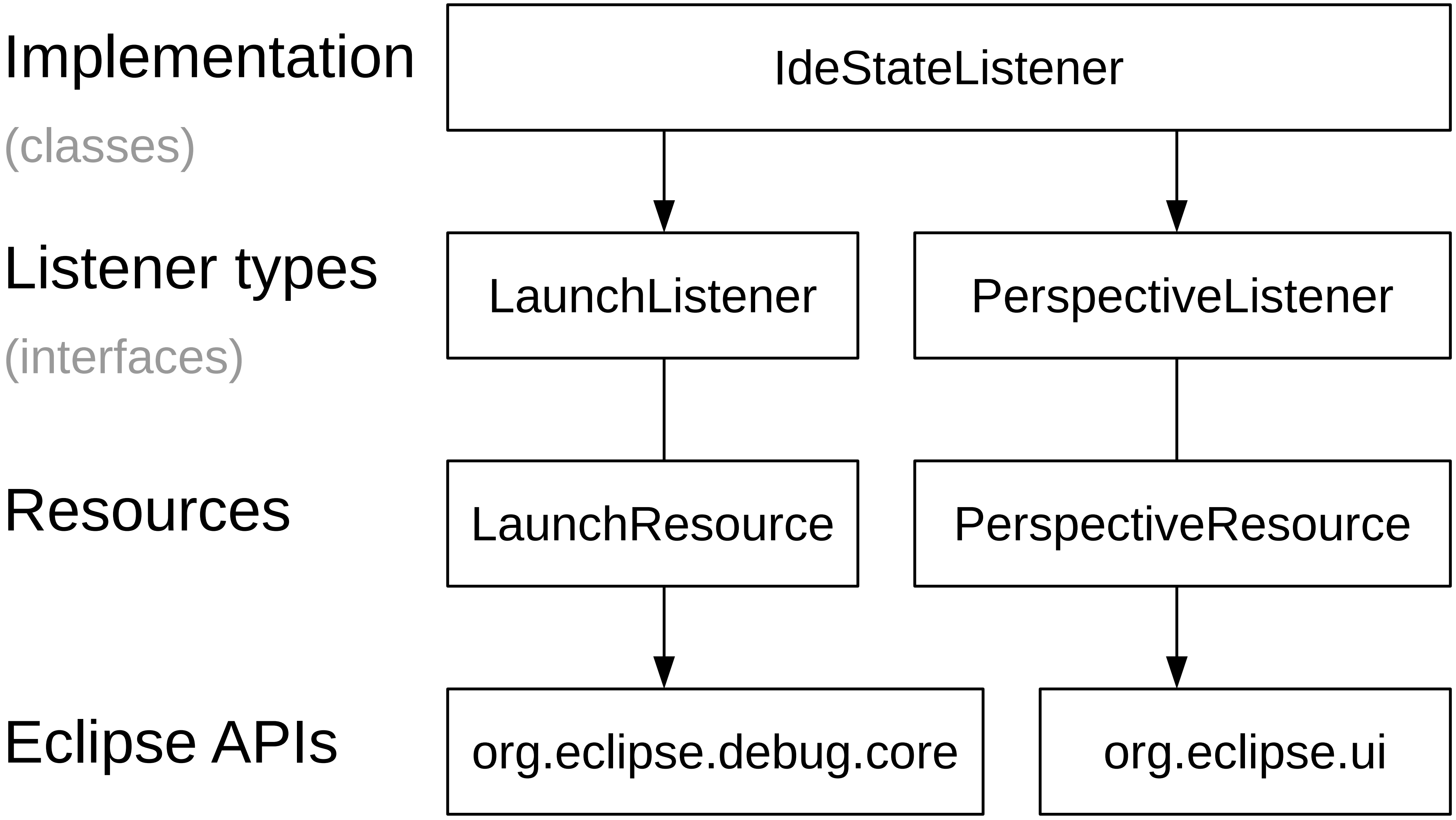
Listener Registration Hooks

- When to load or store listener caches?
- How to log events of type `OPEN` for files opened before listener registration?

- Invoked automatically during registration and unregistration on specified listener:

```
preRegister()      postRegister()
preUnregister()    postUnregister()
```

Listener Registration Handlers



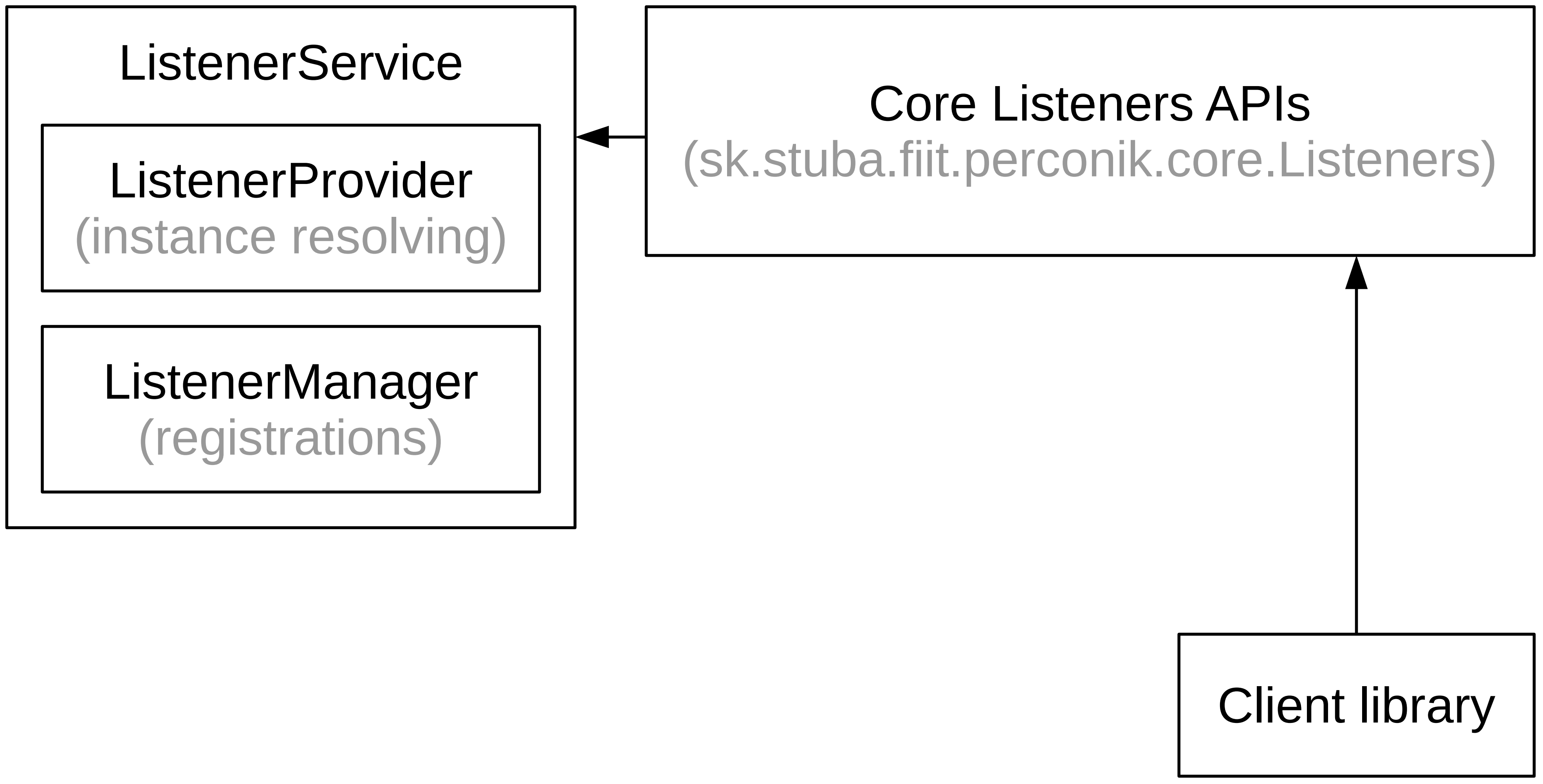
Dynamic Singleton Resolving

- Listeners must be singletons
- How to implement? What is the best way? (EJSE#3)

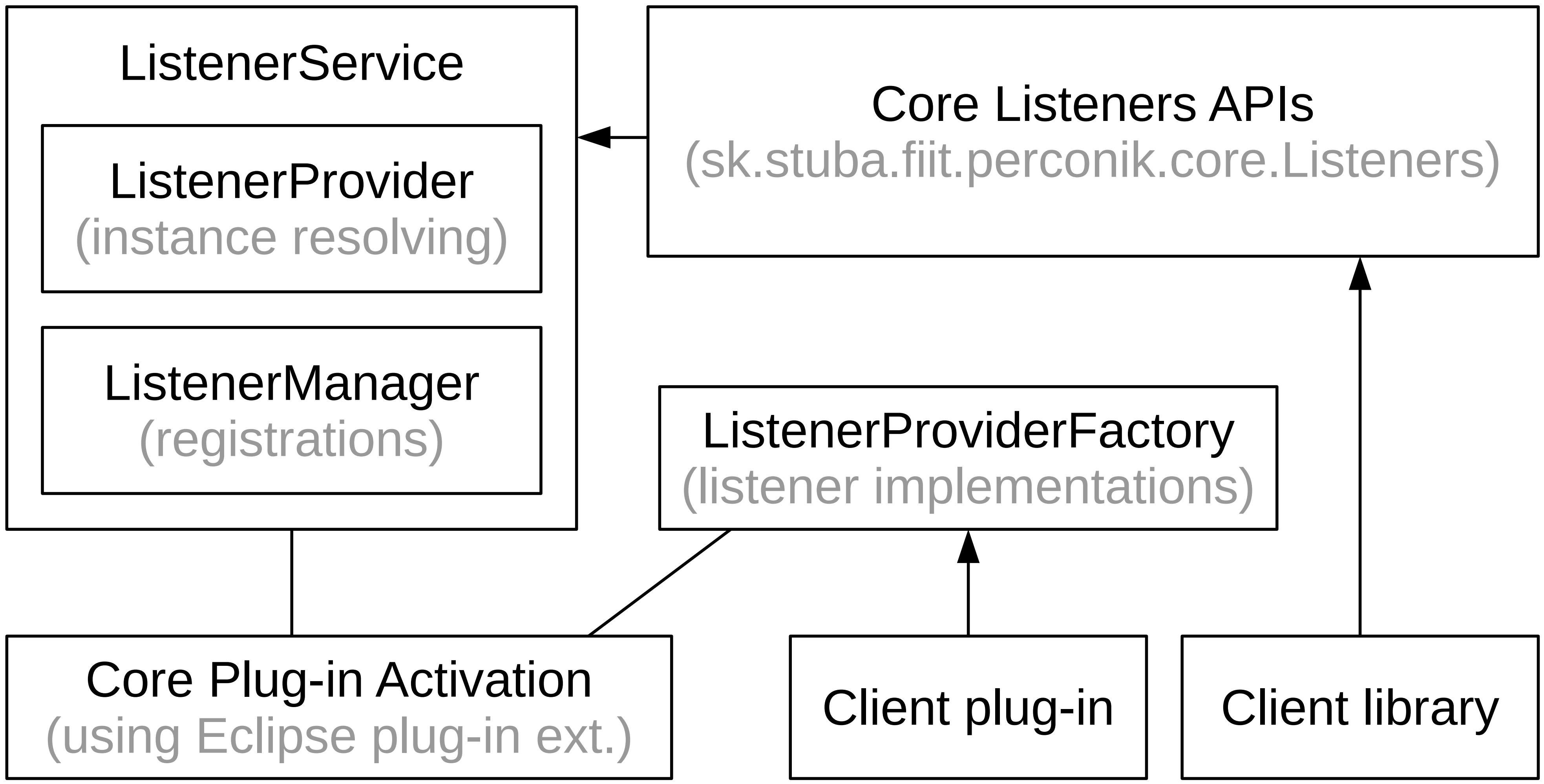
- Class based listener singleton instance lookup:
  - Enum constant `INSTANCE` (case insensitive)
  - Class method `getInstance` (`static`)
  - Class constant `INSTANCE` (case insensitive, `static final`)
  - Class constructor with no parameters (Java Beans style)
- Resolved instance is cached and reused

sk.stuba.fiit.perconik.utilities.reflect.accessor.DelayedLookup

Services and Plug-in Extensions



Services and Plug-in Extensions



(same for core resources – listener registration handlers)

Examples

```
Listener listener = new TextSelectionAdapter() {




};

Listeners.register(listener);
```

Examples

```
Listener listener = new TextSelectionAdapter() {
  public void selectionChanged(IWorkbenchPart part,
  ITextSelection selection) {
    System.out.println(selection.getText());
  }
};

Listeners.register(listener);
```

Examples

```
enum MyListener implements TextSelectionListener {
  INSTANCE;

  ...
};

Listeners.register(MyListener.INSTANCE);
```

Examples

```
public class MyListenerProviderFactory implements
ListenerProviderFactory {
  public ListenerProvider
  create(ListenerProvider parent) {

    return ListenerProviders.builder()
    .add(MyListener.class)

    .parent(parent)
    .build();
  }
}
```

Examples

```
public class MyListenerProviderFactory implements
ListenerProviderFactory {
  public ListenerProvider
  create(ListenerProvider parent) {
    Set<Class<? extends Listener>> classes = ...;
    return ListenerProviders.builder()
    .add(MyListener.class)
    .addAll(classes)
    .parent(parent)
    .build();
  }
}
```

Examples

```
public class MyListenerProviderFactory implements
ListenerProviderFactory {
  public ListenerProvider
  create(ListenerProvider parent) {
    Set<Class<? extends Listener>> classes = ...;
    return ListenerProviders.builder()
    .add(MyListener.class)
    .addAll(classes)
    .parent(parent)
    .build();
  }
}
// Plug-in XML - Extensions – Add
// sk.stuba.fiit.perconik.core.services.listeners
// New provider - Set class to
// MyListenerProviderFactory
```

Examples

```
enum MyListener implements TextSelectionListener {
  INSTANCE;
  static Executor executor = newCachedThreadPool();













  ...
};
```

Examples

```
enum MyListener implements TextSelectionListener {
  INSTANCE;
  static Executor executor = newCachedThreadPool();

  public void selectionChanged(IWorkbenchPart part,
  ITextSelection selection) {

    executor.execute(new Runnable() {
      public void run() {
        process(selection);
      }
    });
  }

  ...
};
```

Examples

```
enum MyListener implements TextSelectionListener {
  INSTANCE;
  static Executor executor = newCachedThreadPool();

  public void selectionChanged(IWorkbenchPart part,
  ITextSelection selection) {
    final long time = currentTime();
    executor.execute(new Runnable() {
      public void run() {
        process(time, selection);
      }
    });
  }

  ...
};
```

Source Code

- Open Source under the MIT License
- Organization and repositories at GitHub

Links

http://perconik.github.io

Eclipse Integration Site

http://perconik.github.io/update

Eclipse Update Site

http://github.com/perconik/perconik

Source Code Repository

Summary

- 3 Eclipse Features and Plug-ins
- Several nifty features + Javadocs
- Support for 30 most used Eclipse listeners
- Synchronized with UI, configurable at runtime
- Easy to use, both directly or via plug-in ext.
- Gratex Activity Watcher, Content Tags and Services
- Under the hood (implementation & tricks)
- Open Source under MIT License at GitHub