Designing and Implementing IDE Cross-Compatibility for Saros
Master’s Thesis Defense

Tobias Bouschen
Freie Universität Berlin

November 26th, 2020
Outline

Introduction
- Saros
- Short Saros Development History

Thesis Topic
- Problem Statement
- Thesis Goal

Approach
- Preparation
- Requirement Analysis

Identified Roadblocks
- Roadblock 1 — Issues With the Handling of Line Endings
- Roadblock 2 — Issues With the Filesystem Implementations
- Roadblock 3 — Issues With Determining Which Resources Not to Share

Results

Evaluation

Future Work
Outline

Introduction

Saros

Short Saros Development History

Thesis Topic

Approach

Identified Roadblocks

Results

Evaluation

Future Work
Saros

- Plugin for distributed pair programming (DPP)
  - Allows collaborative real-time editing of text files
  - Provides shared workspace awareness information

- Developed at the Freie Universität since 2006
- Available on GitHub: https://github.com/saros-project/saros
Outline

Introduction

Saros

Short Saros Development History

Thesis Topic

Approach

Identified Roadblocks

Results

Evaluation

Future Work
Short Saros Development History

- Initially created in 2006 as part of diploma thesis of R. Djemili [1]
- Designed as an Eclipse plugin (later named Saros/E)
- Later on decided to add support for other IDEs
- Moved core business logic of Saros to a separate component to simplify the process (Saros Core)
- Added Saros implementation for IntelliJ IDEA (Saros/I)
Outline

Introduction

Thesis Topic
Problem Statement
Thesis Goal

Approach

Identified Roadblocks

Results

Evaluation

Future Work
Problem Statement

- Many core Saros APIs were too closely modeled after Eclipse APIs
  - Made creating and maintaining Saros implementations for other IDEs slow and cumbersome

- Existing Saros implementations were not compatible
  - Tight integration with the underlying IDE systems led to clashes where IDE concepts didn’t match
Outline

Introduction

Thesis Topic
  Problem Statement
  Thesis Goal

Approach

Identified Roadblocks

Results

Evaluation

Future Work
Thesis Goal

- Enabling Saros IDE cross-compatibility

1. Analyze the state of the Saros plugin and its IDE implementations.

2. Identify potential roadblocks on the way to supporting IDE cross-compatibility.

3. Resolve the necessary roadblocks.
Outline

Introduction

Thesis Topic

Approach
  Preparation
  Requirement Analysis

Identified Roadblocks

Results

Evaluation

Future Work
Preparation

- Read up on Saros literature [2, 3, 4, 5, 6]
- Skimmed Saros codebase looking for potential issues
- Discussed general Saros design ideas and potential technical issues with other Saros developers
Outline

Introduction

Thesis Topic

Approach
  Preparation
  Requirement Analysis

Identified Roadblocks

Results

Evaluation

Future Work
Requirement Analysis

▶ Analyzed existing implementation
▶ Created list of roadblocks
  ▶ Proposed and discussed different solution approaches for each roadblock
  ▶ Created list of preliminary steps necessary to resolve each roadblock
Outline

Introduction

Thesis Topic

Approach

**Identified Roadblocks**
- Roadblock 1 — Issues With the Handling of Line Endings
- Roadblock 2 — Issues With the Filesystem Implementations
- Roadblock 3 — Issues With Determining Which Resources Not to Share

Results

Evaluation

Future Work
Roadblock 1 — Issues With the Handling of Line Endings

- Saros works on the editor content provided by the IDE
- IntelliJ IDEA normalizes editor contents to use UNIX line ending (‘\n’)
- Eclipse uses the original line ending for the editor content

⇒ Editor content does not necessarily match between IDEs even if content on disk is identical

**Implemented Solution:**
- Introduced line ending normalization for editor content handled by Saros
- Introduced line-ending-agnostic text position logic
Outline

Introduction

Thesis Topic

Approach

Identified Roadblocks

Roadblock 1 — Issues With the Handling of Line Endings
Roadblock 2 — Issues With the Filesystem Implementations
Roadblock 3 — Issues With Determining Which Resources Not to Share

Results

Evaluation

Future Work
Resource sharing logic was to closely coupled with the IDE project/module model
  ▶ Necessary to provide additional functionalities like creating a module/project as part of session start

Project model of Eclipse and module model of IntelliJ IDEA are very different
  ▶ Transforming resources from one model to the other is non-trivial
  ▶ A direct mapping is not always possible, even for “normal” project setups (e.g. Saros)

**Implemented Solution:**
▶ Dropped support for project/module creation as part of session start
▶ Introduced a new, IDE-independent resource handling based on sharing “reference points”
Outline

Introduction

Thesis Topic

Approach

Identified Roadblocks

Roadblock 1 — Issues With the Handling of Line Endings
Roadblock 2 — Issues With the Filesystem Implementations
Roadblock 3 — Issues With Determining Which Resources Not to Share

Results

Evaluation

Future Work
Roadblock 3 — Issues With Determining Which Resources Not to Share

- Projects/modules may contain resources that should not be shared
  - Automatically generated artifacts
  - User-/system-specific configurations
- Saros uses IDE logic-specific to identify such files
- IDEs might not agree on which resources should not be shared
- IDE logic might not cover all resources that are not supposed to be shared

Not Resolved — Solution approach:
- Host determines which resources are non supposed to be shared
- Offer other ways of determining what not to share
  - A ‘.sarosignore’ file
  - Integration with version control systems like Git (i.e. the ‘.gitignore’ configuration)
Outline

Introduction

Thesis Topic

Approach

Identified Roadblocks

Results

Evaluation

Future Work
Results

- Resolved roadblocks 1 & 2
- Fixed additional minor issues regarding IDE cross-compatibility
  ⇒ Allowed for basic IDE cross-compatibility between Saros/E and Saros/I
- Provided solution approaches for roadblock and issues not resolved as part of this thesis
- Simplified and generalized Saros filesystem interfaces
Outline

Introduction

Thesis Topic

Approach

Identified Roadblocks

Results

Evaluation

Future Work
Evaluation

- Provided basic IDE cross-compatibility
- Open topics mainly pertain to improving the usability of Saros in cross-IDE setups
- IDE cross-compatibility still needs more testing
- Usability of UI was reduced due to major changes in central Saros design concepts
Outline

Introduction
Thesis Topic
Approach
Identified Roadblocks
Results
Evaluation

Future Work
Future Work

▶ Short-term goals:
  ▶ Add missing system tests (STF) for new sharing model
  ▶ Resolve remaining roadblock and issues
  ▶ Do more IDE cross-compatibility testing

▶ Long-term goals:
  ▶ Improve Saros UI to better reflect new sharing model
    ▶ Already resolve some of the usability issues related to creating projects
  ▶ Finish first stable release of Saros/I


Thank you for listening!

For more information, you can have a look at my thesis page: http://www.inf.fu-berlin.de/w/SE/ThesisSarosCrossIDE
Implementing Line Ending Normalization

- Implemented a line-ending-agnostic text positioning logic
  - Defines a text position using the line number and in-line offset

- Adjusted operational transformation logic in Jupiter algorithm for new text positioning scheme
  - Also extended test-suite to cover all new transformation cases

- Introduced line ending normalization
  - replaces all line endings with ‘\n’ when reading from an editor
  - re-substitutes the original line endings when writing content back to an editor
Preparing for Filesystem Rework

- Removed partial sharing
- Reworked resource transport logic
- Simplified filesystem interfaces
  - Removed unnecessary method parameters
  - Removed unnecessary methods
  - Introduced enum for resource types
  - Added documentation for all filesystem interfaces
Implementing an IDE-Independent Filesystem Model

- Implemented reference-point-based filesystem model for Saros/I
- Adjusted Saros/I UI to match new sharing model
- Cleaned up Saros Core
- Implemented reference-point-based filesystem model for Saros/E
- Adjusted Saros/E UI to match new sharing model
- Adjusted STF for new sharing model
Other Resolved Issues

- Implemented a unified handling for non-text editors
- Fixed a previously undiscovered fault in the operational transformation logic
- Implemented a unified handling for file moves
- Implemented a unified handling for character encodings
- Introduced improved local resource mapping suggestions
Open Topics & Issues

- Resolving Roadblock 3 — Implementing a new system to determine what not to share
- UI improvements for Saros/E
- Extending existing character encoding handling
- Introducing a unified BOM handling