CrossDoc

Team: Octo-Docs

Team Members:
Garrison Smith
Peter Huettl
Kristopher Moore
Brian Saganey

Client/Mentor

- Dr. James Palmer
  - Associate Professor at NAU -SICCS
- Dr. John Georgas
  - Associate Professor at NAU -SICCS
- Nakai McAddis
  - Graduate Professor

NORTHERN ARIZONA UNIVERSITY

Problem Statement

General Problem

Software/documentation interdependence

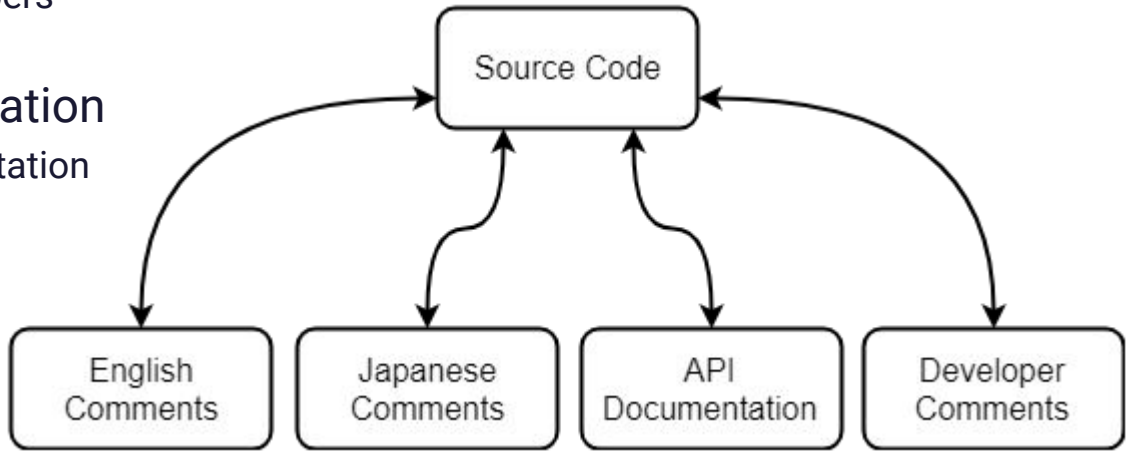
- Documentation is buried in software
- Software/documentation tightly coupled

```
int parseCamera(camera_t *camera, char *line);

/**
 * English
 * Helper function used to parse light properties from string.
 *
 * Swedish
 * Hjälperfunktionen användes för att analysera
 * ljusegenskaper från strängen.
 *
 * TODO
 * Consider adding support for spot lights
 *
 * Design Insight
 * The light pointer is passed in as a parameter to allow
 * the actual return value to denote the error status of
 * the function call. If successful, the memory address
 * pointed to by the light parameter is populated.
 *
 * @param  camera  pointer to output light
 * @param  line    string containing light data to parse
 * @return         error status of parsing
 */
int parseLight(light_t *light, char *line);
```

Specific Problem

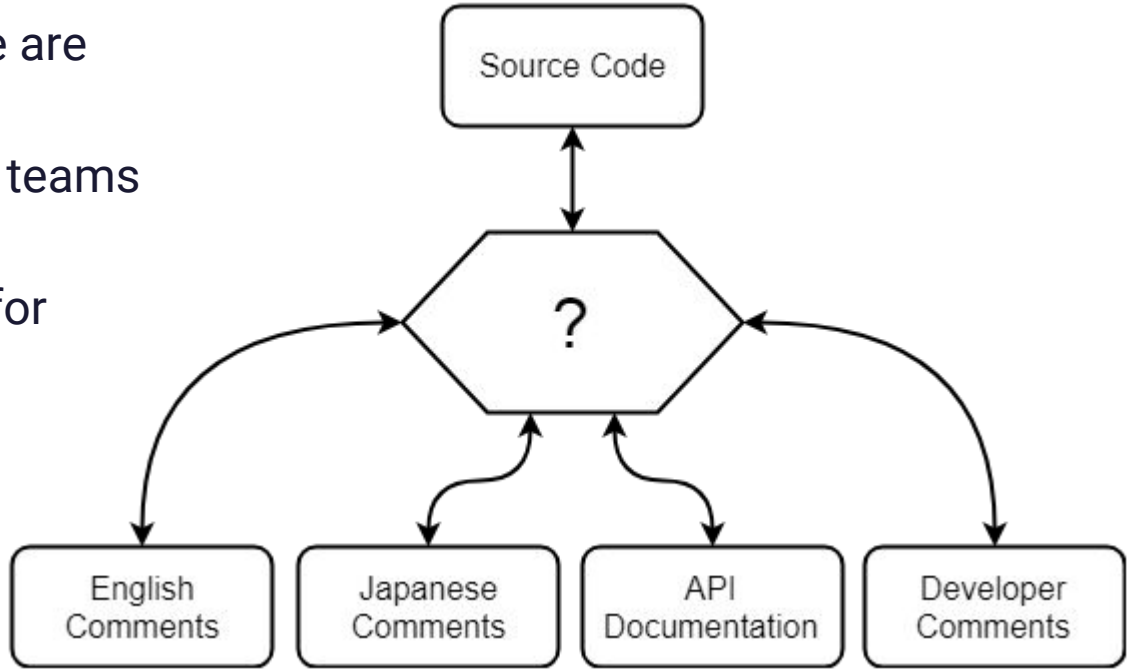
- Large companies with large projects
  - Culturally diverse developers
  - Language barrier
- Software and Documentation
  - Misunderstood documentation
  - Comments mismatched
  - within the codebase



Solution Statement

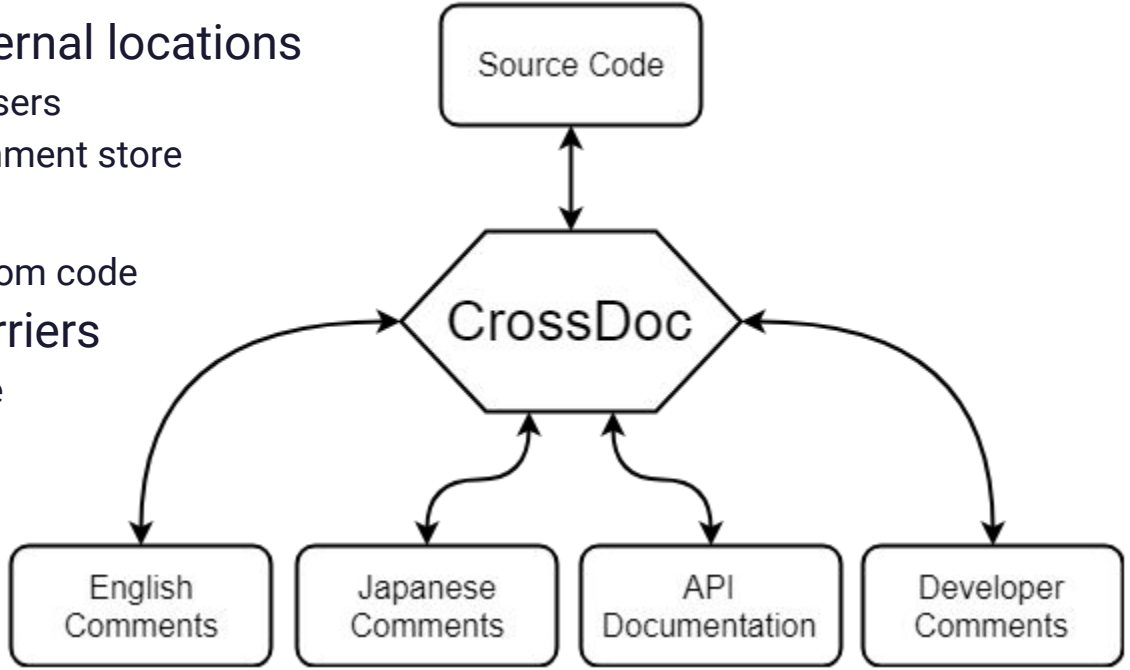
Improved Commenting System

- Comments and software are decoupled
- Easily scalable for small teams and large organizations
- Comments are specific for developers
  - Specific to cultural
  - Specific to language



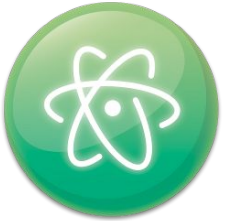
The Solution: CrossDoc

- Comments stored in external locations
  - Easily accessible for all users
  - Editable in code or in comment store
- Scales alongside teams
  - Expands independently from code
- Breaks down cultural barriers
  - Easily store and reference comments in different languages



CrossDoc Implementation

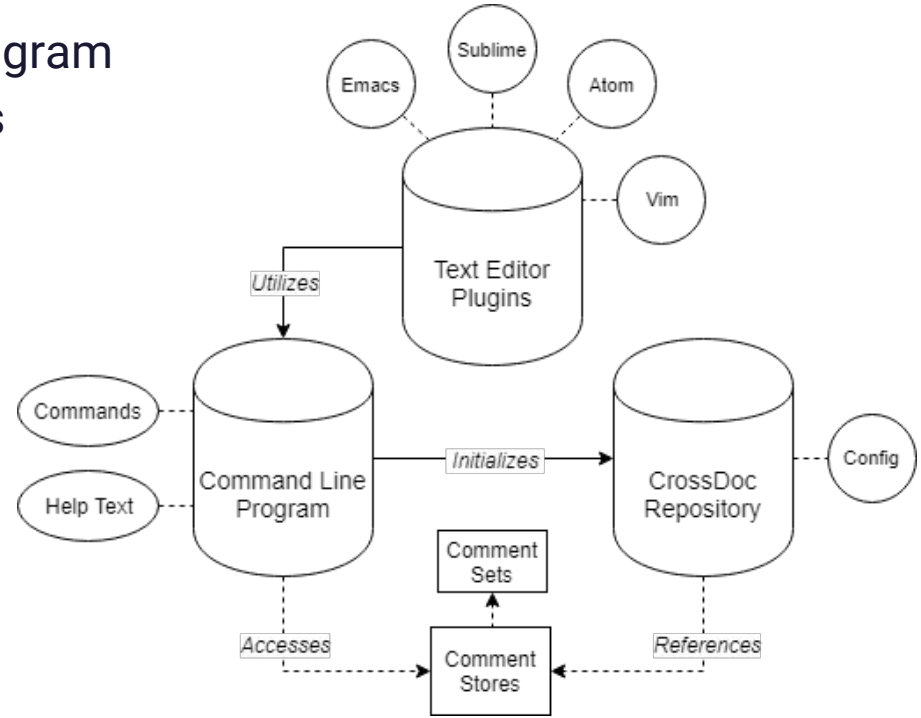
- Key Requirements
  - Simple setup for teams and organizations of any size
  - All Comments stored in one location
  - Simple editing of comments for non-programmers
  - Users can simply create, delete, and update comments within text-editors
    - Atom
    - Emacs
    - Sublime
    - Vim



Architectural Overview

High Level Overview

- Back-end command line program
- Front-end text editor plugins
- CrossDoc repository



Command Line Program

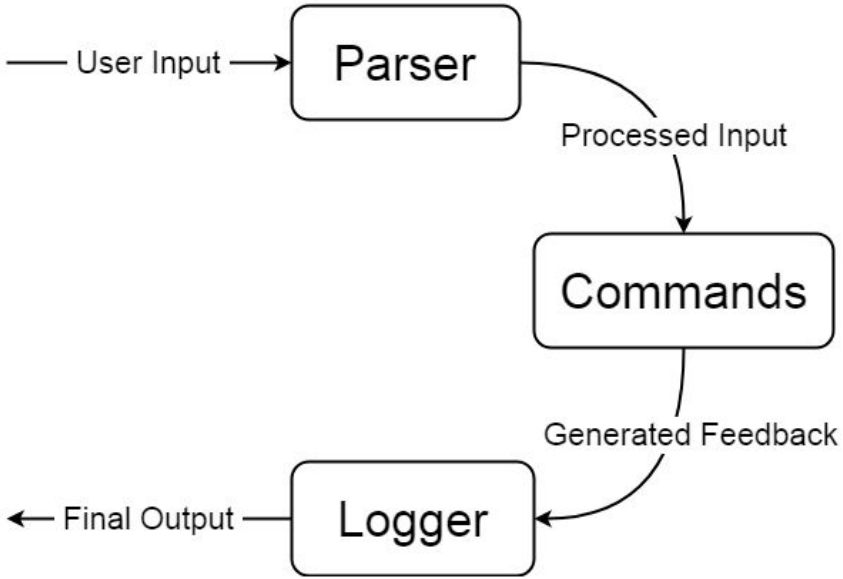
- Implements core functionality
  - Create comments
  - Read comments
  - Delete comments
  - Etc..
- Provides API to interact with tool
- Text editor agnostic

```
λ cross-doc --help
usage: cross-doc <command>

All CrossDoc commands:
  init
  create-comment
  generate-anchor
  fetch-comment
  delete-comment
  update-comment
```

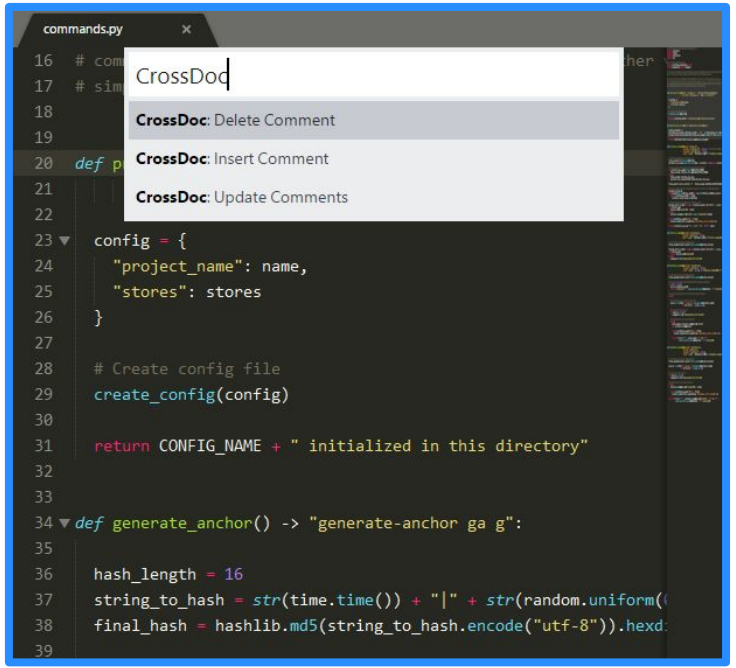
Command Line Program

- Parser
  - Reads input
  - Delegates to commands
- Commands
  - Implements CrossDoc functionality
- Logger
  - Provides concise output
  - Outputs help text where necessary



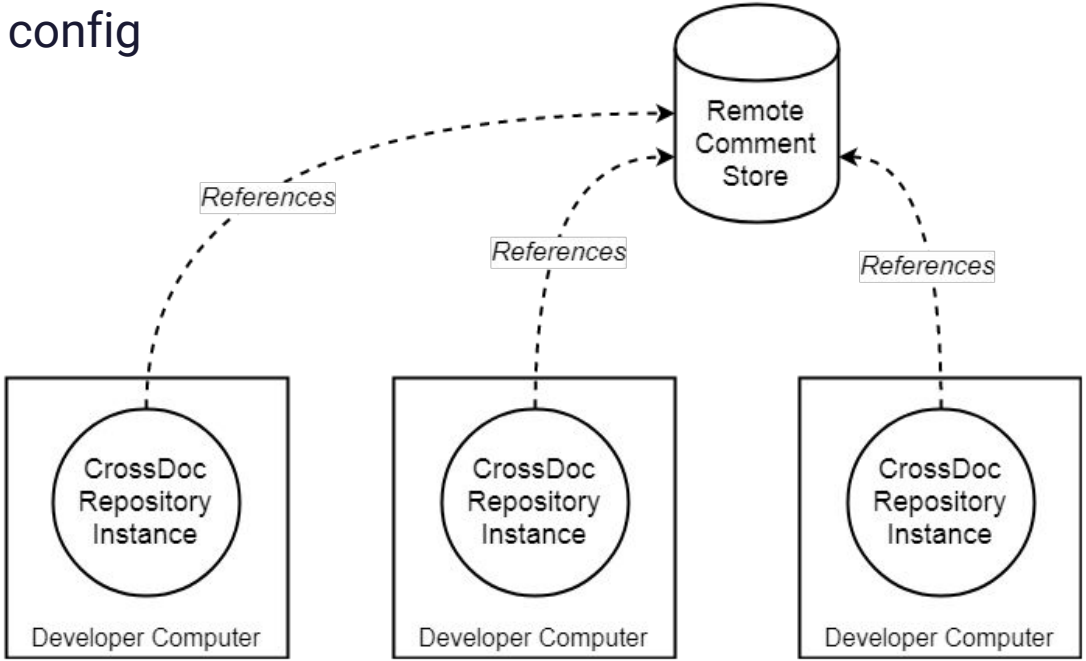
Text Editor Plugins

- CrossDoc user interface
- Intuitive commands and hotkeys
- Support for multiple text editors
  - Atom
  - Emacs
  - Sublime
  - Vim



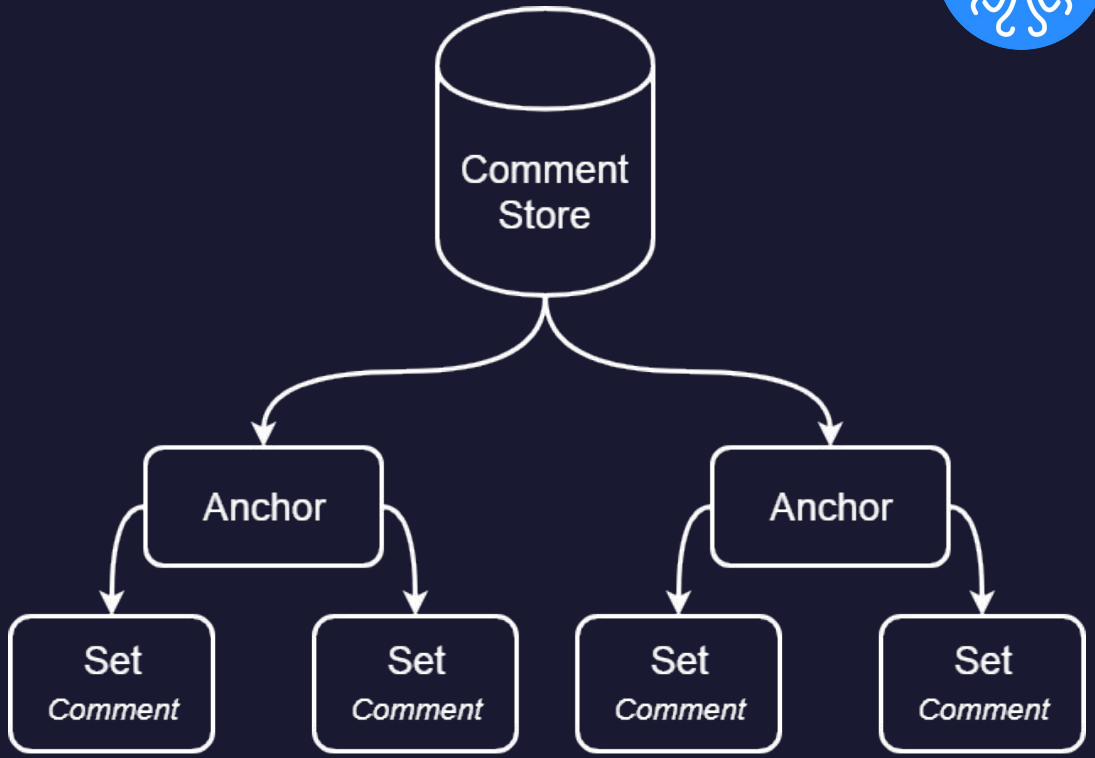
CrossDoc Repository

- Identified by a custom config file *(cdoc-config.json)*
- Stores references to comment stores
- Persistent meta-data storage



Comment Storage

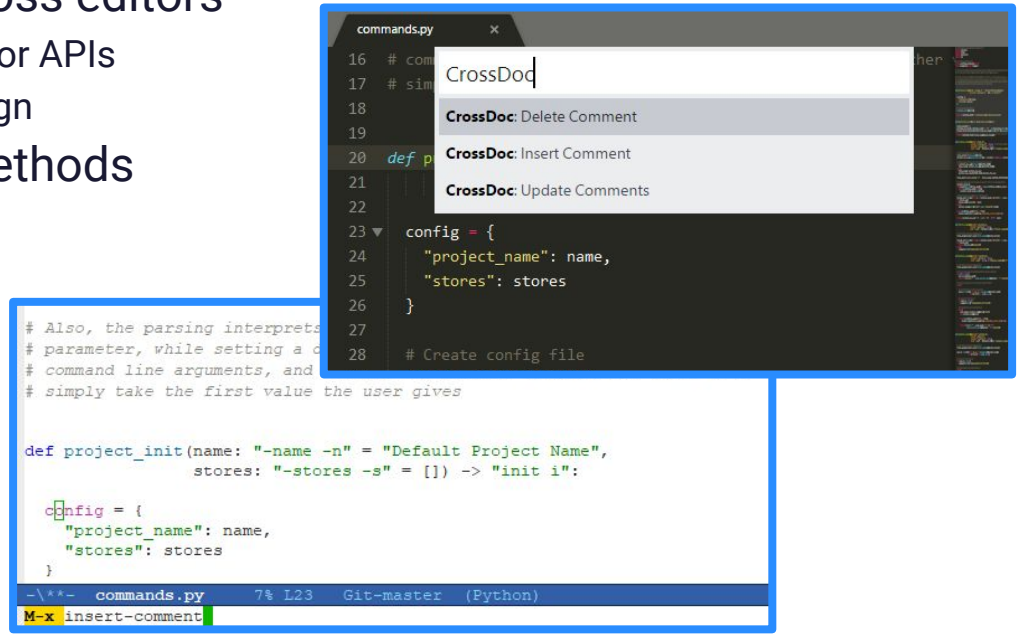
- Comment stores
  - Directory of anchors
  - Local and remote
- CrossDoc anchors
  - Comment identifier
- Comment sets
  - Distinct categories
  - Stores comment text



Implementation Challenges

Development Challenges

- Consistent functionalities across editors
  - Managing limitations of text editor APIs
  - Developing a consistent UX design
- Managing multiple storage methods
  - Remote and local storage
  - Comment validation
  - Using OS agnostic methods

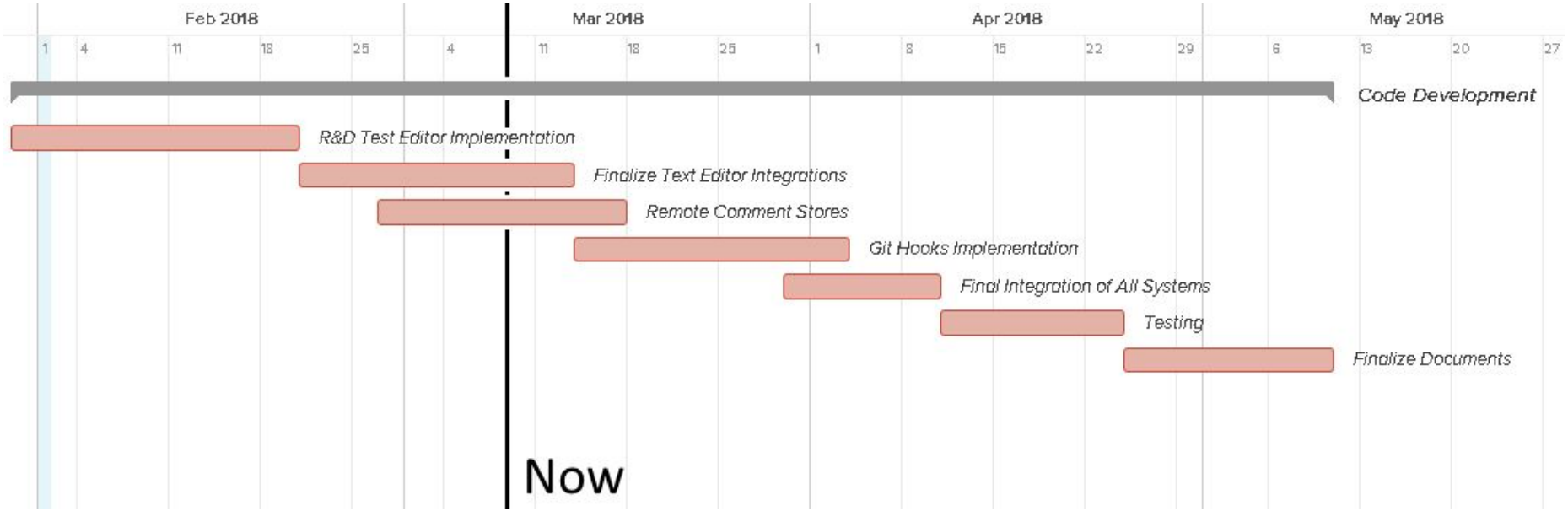


Development Solutions

- ~~Consistent functionalities across editors~~
  - Designed Cdoc command format as adaptable for all editors
  - Language **agnostic** and **extendable** to Atom, Emacs, Sublime, and Vim APIs
  - Simple format, with recognizable commands for ease of use
- ~~Managing multiple storage methods~~
  - Implemented CrossDoc CL-Parser to adapt to flexible storage inputs
  - Validation of comments and stores, preventing issues when anchor is clipped

Development Schedule

Gantt Chart


Feb 2018
Mar 2018
Apr 2018
May 2018
1
4
11
18
25
4
11
18
25
1
8
15
22
29
6
13
20
27
Code Development
R&D Test Editor Implementation
Finalize Text Editor Integrations
Remote Comment Stores
Git Hooks Implementation
Final Integration of All Systems
Testing
Finalize Documents
Now

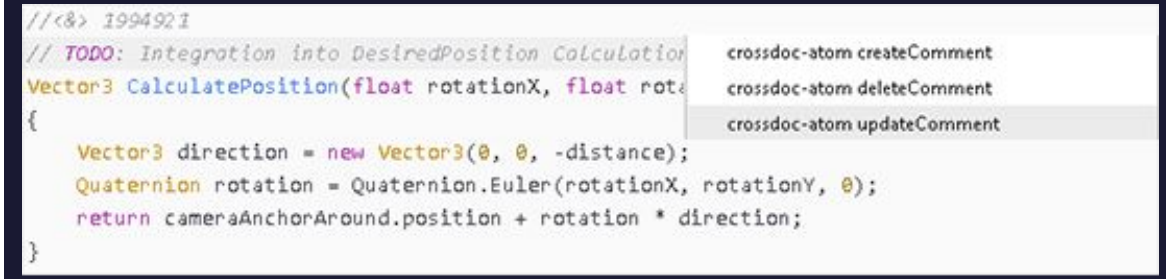
Development Milestones

Command Line Parser:

The interface of CrossDoc, tool extended by other Subsystems.

Text Editor Plugin Extensions:

- Atom
- Emacs
- Sublime
- Vim



Conclusion

Summary: Problem

- Documentation is buried, too reliant on Codebase

```
/* Returns a vector calculation position given Smoothed Mouse Y/X axes and Distance
  スムーズマウスY / X軸と距離を指定し てベクトル計算位置を返します
  إرجاع موقف حساب متجه نظرا محسوس الماوس Y / X والمسافة المحاور
  Gibt eine Vektorberechnungsposition zurück, für die die Y / X-Achsen der geglätteten
  Maus und die Entfernung angegeben wurden
  Renvoie une position de calcul vectorielle donnée pour les axes Y / X lissés et Distance
  Возвращает векторную расчетную позицию, заданную сглаженными осями Y / X мыши и расстоянием
  Trả lại vị trí tính toán véc tơ cho các trục Y / X Chuột Mịn và Khoảng cách
*/
Vector3 CalculatePosition(float rotationX, float rotationY, float distance)
{
    Vector3 direction = new Vector3(0, 0, -distance);
    Quaternion rotation = Quaternion.Euler(rotationX, rotationY, 0);
    return cameraAnchorAround.position + rotation * direction;
}
```

Summary: Solution

- Provide a better way to comment with CrossDoc!

- Scalable, Exterior Storage, and Enhanced Comment Functionalities.

```
//<&> 1994921
// Overview: Returns a vector calculation position given Smoothed Mouse Y/X axes and Distance
Vector3 CalculatePosition(float rotationX, float rotationY, float distance)
{
    Vector3 direction = new Vector3(0, 0, -distance);
    Quaternion rotation = Quaternion.Euler(rotationX, rotationY, 0);
    return cameraAnchorAround.position + rotation * direction;
}
```

```
//<&> 1994921
// LocalizationJapanese: スムーズマウスY / X軸と距離を指定し てベクトル計算位置を返します
Vector3 CalculatePosition(float rotationX, float rotationY, float distance)
{
    Vector3 direction = new Vector3(0, 0, -distance);
    Quaternion rotation = Quaternion.Euler(rotationX, rotationY, 0);
    return cameraAnchorAround.position + rotation * direction;
}
```

The Path Ahead

- Remote Comment Stores
  - Accessible remote versions of Local Comment Store system
  - Extension of CL-Parsers functionality
- Git Hooks Implementation
  - Incorporate Git-Hooks pre/post commit system to allow CrossDoc to remove comments from the official commits