CrossDoc

Team: Octo-Docs

Team Members:
Garrison Smith
Peter Huettl
Kristopher Moore

Client/Mentors

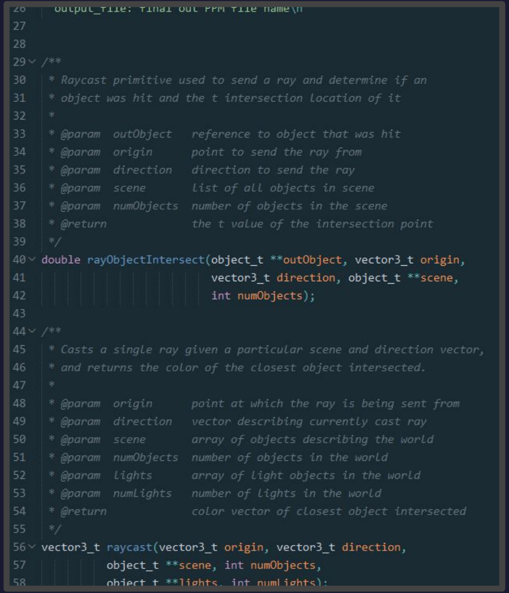
- Dr. James Palmer
  - Associate Professor at NAU - SICCS
- Dr. John Georgas
  - Associate Professor at NAU - SICCS
- Nakai McAddis
  - Graduate Professor

NORTHERN ARIZONA UNIVERSITY

The General Problem

Software/documentation interdependence

- Documentation gets buried
- Disorganized code base
- Comments are reliant on code

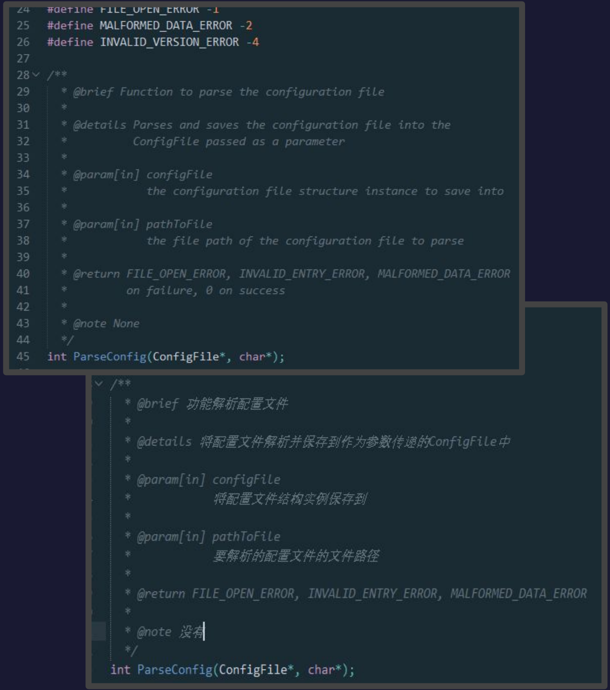


```c
 output_file: final out PPM file name\n


/**
 * Raycast primitive used to send a ray and determine if an
 * object was hit and the t intersection location of it
 *
 * @param  outObject   reference to object that was hit
 * @param  origin      point to send the ray from
 * @param  direction   direction to send the ray
 * @param  scene       list of all objects in scene
 * @param  numObjects  number of objects in the scene
 * @return             the t value of the intersection point
 */
double rayObjectIntersect(object_t **outObject, vector3_t origin,
                          vector3_t direction, object_t **scene,
                          int numObjects);

/**
 * Casts a single ray given a particular scene and direction vector,
 * and returns the color of the closest object intersected.
 *
 * @param  origin      point at which the ray is being sent from
 * @param  direction   vector describing currently cast ray
 * @param  scene       array of objects describing the world
 * @param  numObjects  number of objects in the world
 * @param  lights      array of light objects in the world
 * @param  numLights   number of lights in the world
 * @return             color vector of closest object intersected
 */
vector3_t raycast(vector3_t origin, vector3_t direction,
          object_t **scene, int numObjects,
          object_t **lights, int numLights);
```

Problem: Specific Example

- Large companies and large projects
  - Culturally diverse developers
  - Language barrier

- Software and Documentation
  - Misunderstood documentation
  - Software making it hard to find comments
  - Disorganized codebase



```c
#define FILE_OPEN_ERROR -1
#define MALFORMED_DATA_ERROR -2
#define INVALID_VERSION_ERROR -4

/**
  * @brief Function to parse the configuration file
  *
  * @details Parses and saves the configuration file into the
  *          ConfigFile passed as a parameter
  *
  * @param[in] configFile
  *            the configuration file structure instance to save into
  *
  * @param[in] pathToFile
  *            the file path of the configuration file to parse
  *
  * @return FILE_OPEN_ERROR, INVALID_ENTRY_ERROR, MALFORMED_DATA_ERROR
  *         on failure, 0 on success
  *
  * @note None
  */
int ParseConfig(ConfigFile*, char*);
```

```c
/**
  * @brief 功能解析配置文件
  *
  * @details 将配置文件解析并保存到作为参数传递的ConfigFile中
  *
  * @param[in] configFile
  *            将配置文件结构实例保存到
  *
  * @param[in] pathToFile
  *            要解析的配置文件的文件路径
  *
  * @return FILE_OPEN_ERROR, INVALID_ENTRY_ERROR, MALFORMED_DATA_ERROR
  *
  * @note 没有
  */
int ParseConfig(ConfigFile*, char*);
```

CrossDoc: The General Solution

- A flexible, external comment storage system
  - Comment **references**
  - Independent of codebase
  - Flexible comment editing interfaces

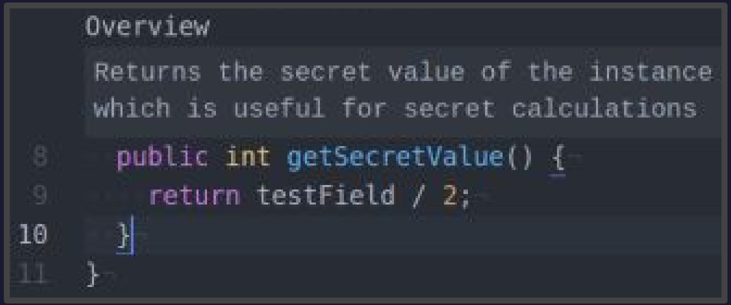
- Extend comment functionality
  - Hierarchical comment stores
  - Modularized comment sets
  - Language agnostic comment hotkeys

- Introduction of CRUD to commenting
  - Create, Read, Update, Delete
  - Extension of persistent storage into commenting

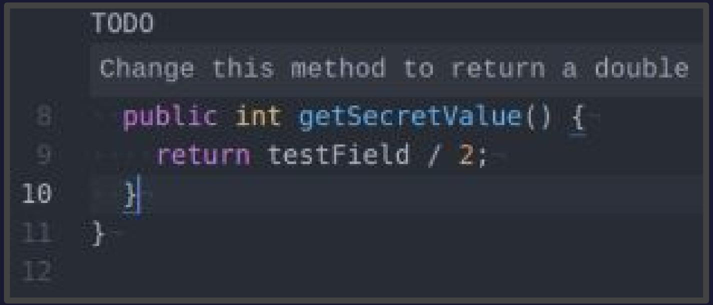
CrossDoc: Applied to Specific Example

- Large Companies / Products:
  - Provides tools for diverse developers to organize comments
  - Distinct comment sets, can be created for Language barriers

- Software Documentation:
  - Separation of comments into categories (Debug, Development, TODO, etc.)
  - Reduces misunderstanding of documentation
  - Provides organized structure for effortless searching



```
Overview
Returns the secret value of the instance
which is useful for secret calculations
public int getSecretValue() {
  return testField / 2;
}
}
```



```
TODO
Change this method to return a double
public int getSecretValue() {
  return testField / 2;
}
}
```

Requirements Elicitation

Implemented several techniques to acquire CrossDoc Requirements:

- Interviews with Client and Sponsor.
- Analysis of client goals, expectations, and probable use cases.
- Examination of existing code and comment bases of potential end-users.

Key Requirements

- Provide **unique** commenting functionalities
- **Intuitive** and easily **adoptable** system
- System that supports a **team environment**
- Improve developmental **workflow efficiency**

Focused Requirement:

System that supports a **team environment**

Supporting a Team Environment

Why this domain?

- Key aspect of CrossDoc
- Directly addresses our problem
- Wide array of requirement types

How to Support a Team

- **Scalability**
  - Expand alongside the company
  - Handle large codebases

- **Portability**
  - Support globalized development teams
  - Work from project-to-project

How CrossDoc Supports Teams

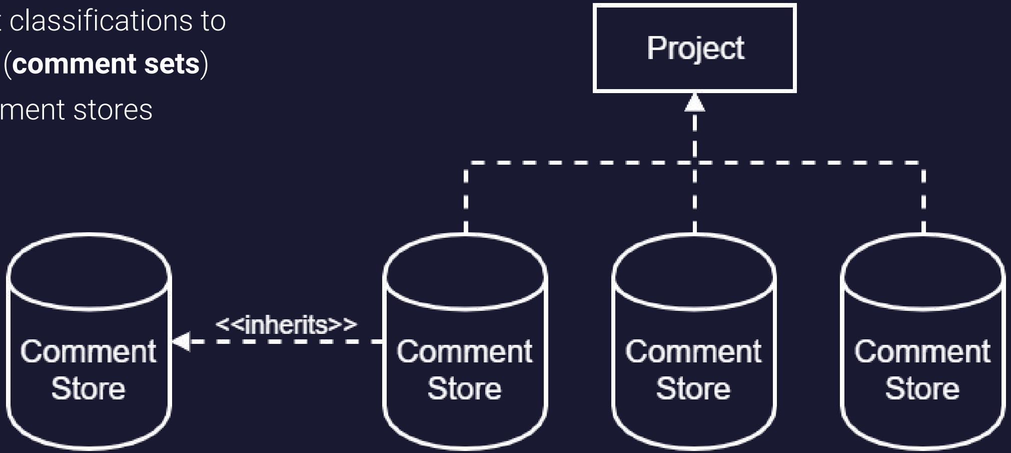
High-Level Requirements:

- Flexible comment storage system
    - Scalability

- Shared data access of comments
    - Portability

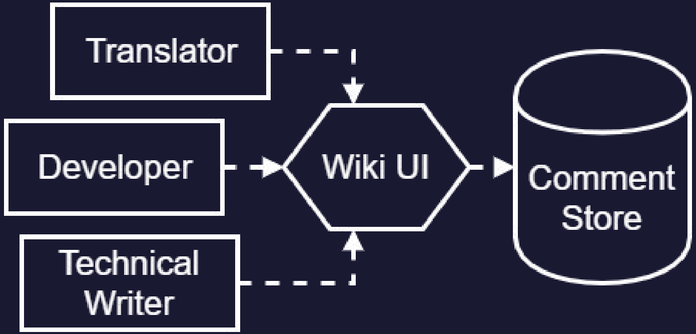
Flexible Comment Storage

- **Functional Requirements**
  - Implement support for **multiple** comment stores
  - Provide comment classifications to group comments (**comment sets**)
  - **Hierarchical** comment stores



Shared Data Access

- **Functional Requirements**
  - Support wiki based comment stores
  - Develop project-specific configuration file format



- **Environmental Requirements**
  - Git version control integration

- **Non-functional Requirements**
  - Portability of our local and server based comment storage

Requirements Summary

- **Supporting Teams**
  - Targeted requirements to provide **scalability**, and **portability**
  - Outlined clear system functions
  - Presented **actionable** and **meaningful** requirements

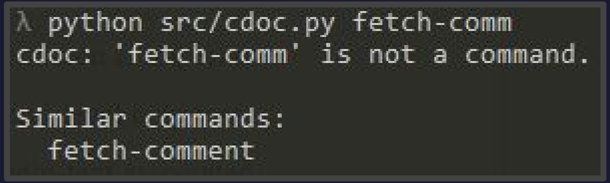
- **Project Requirements**
  - Ever-changing
  - Prepared to update our plan
  - Requirements baseline

Project Risks

- Competing product is released
- Lack of user adoption
- Miscommunication of product purpose

Risk Management

- Easily accessible and universal
  - Available across all text editors
  - Easy to create for users
- Package management system
  - PIP - Python Package Index
- The product will be well documented
  - Action level help text
  - Meaningful error messages

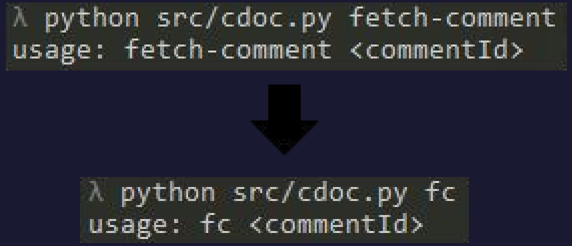


```
λ python src/cdoc.py fetch-comm
cdoc: 'fetch-comm' is not a command.

Similar commands:
  fetch-comment
```

Status Update

These are the objectives that we have completed so far:

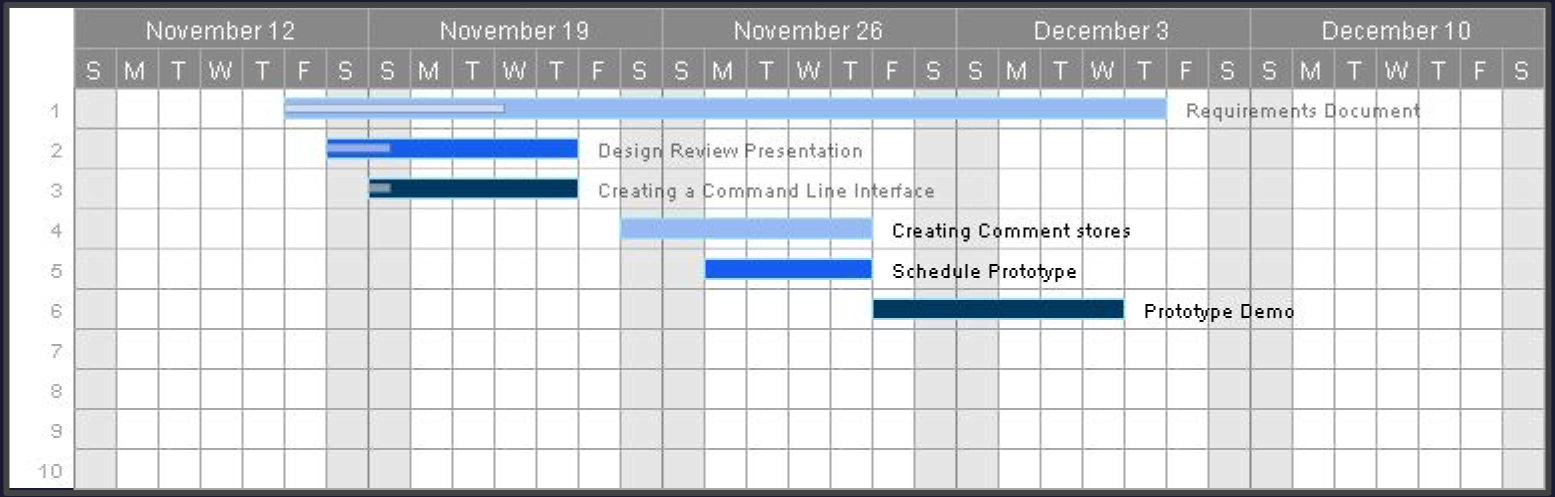
- Requirements Document draft
- Structured outline of the command line interface
- Understanding what lies ahead for our project



```
λ python src/cdoc.py fetch-comment
usage: fetch-comment <commentId>
```

```
λ python src/cdoc.py fc
usage: fc <commentId>
```

Schedule


November 12
November 19
November 26
December 3
December 10
S M T W T F S
1 Requirements Document
2 Design Review Presentation
3 Creating a Command Line Interface
4 Creating Comment stores
5 Schedule Prototype
6 Prototype Demo
7
8
9
10

Conclusion

- Problem: Current state of commenting is inflexible
- Solution: CrossDoc provides extended functionality to the commenting system
- Elicitation: Interviews and Data Analysis
- Requirements: Support teams with Scalability and Portability
- Risk Management: Easily accessible and installable through package manager
- Plan: Finalizations of interface, Prototype of CrossDoc