Internet Applications Design and Implementation

(Lecture 9 - Part 3 - An App end-to-end)

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)
João Leitão (jc.leitao@fct.unl.pt)

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Example for today: Shared Task List

- A project management app where users have a backlog, can create tasks, divide them into sprints, assign them to users, mark them as completed, and see a variety of statistics about tasks and sprints.

Roadmap

1. User stories define a user centred application
2. IFML specifications to specify the (user stories) interactions and components.
3. React components implement each one of the IFML components and views.
4. Interface Mockups define the style and final composition of the UI.

User stories (I)

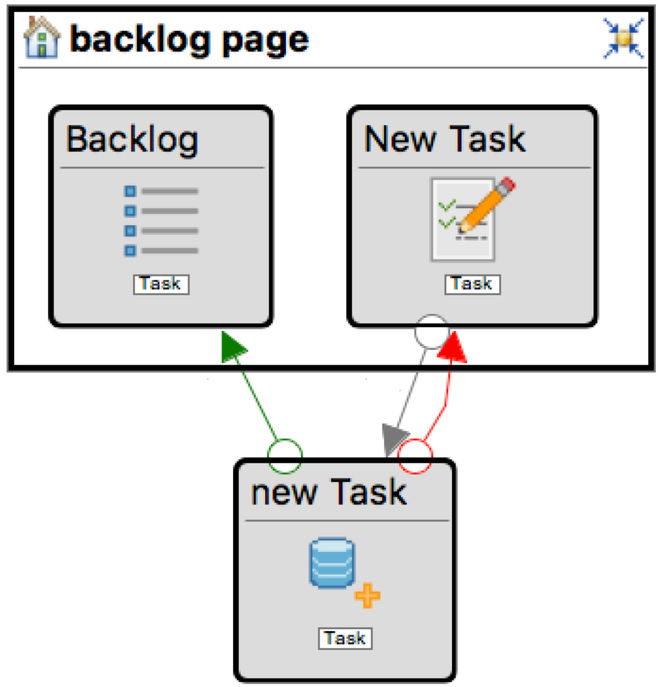
1. A user creates a new **task** given a name, a description, and due date, and sees the result in the **backlog**.
2. A user creates a new **sprint** given a name, a description, and a due date, and sees the result in the list of sprints.
3. A user selects a task (from the backlog or a sprint), assigns it to a (another) sprint and sees its name in the sprint task list.
4. A user selects a task (from the backlog or a sprint), and sees its name, description, and dates.
5. A user selects a task (from the backlog or a sprint), assigns it to a (another) user and sees the result in the details of the task and on the original list.
6. A user selects a task (from the backlog or a sprint), completes it (in a given date) and sees the end date in original list of tasks.

User Stories (II)

7. A user opens its home page, selects a sprint, and sees the sprint details, statistics and task list.
8. A user opens its home page, selects a user, and sees the user details, statistics and tasks list.
9. A user opens its home page, selects the backlog and sees the task list.
10. A user opens its home page, searches the backlog using a text and task properties, and sees the task list filtered by the given criteria.

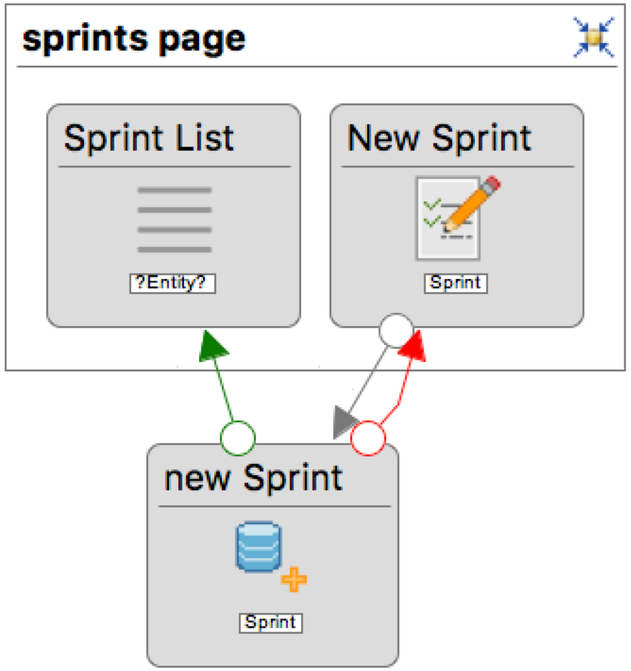
IFML Specification

- A user creates a new **task** given a name, a description, and due date, and sees the result in the **backlog**.



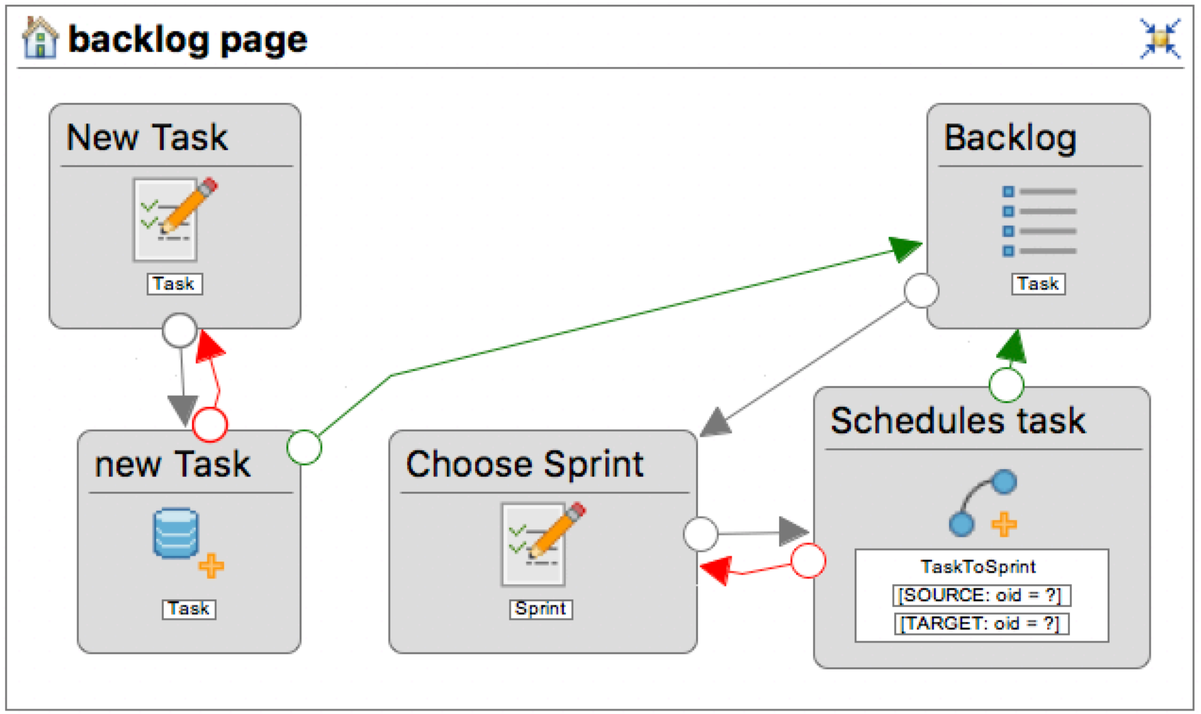
IFML Specification

- A user creates a new **sprint** given a name, a description, and a due date, and sees the result in the list of sprints.



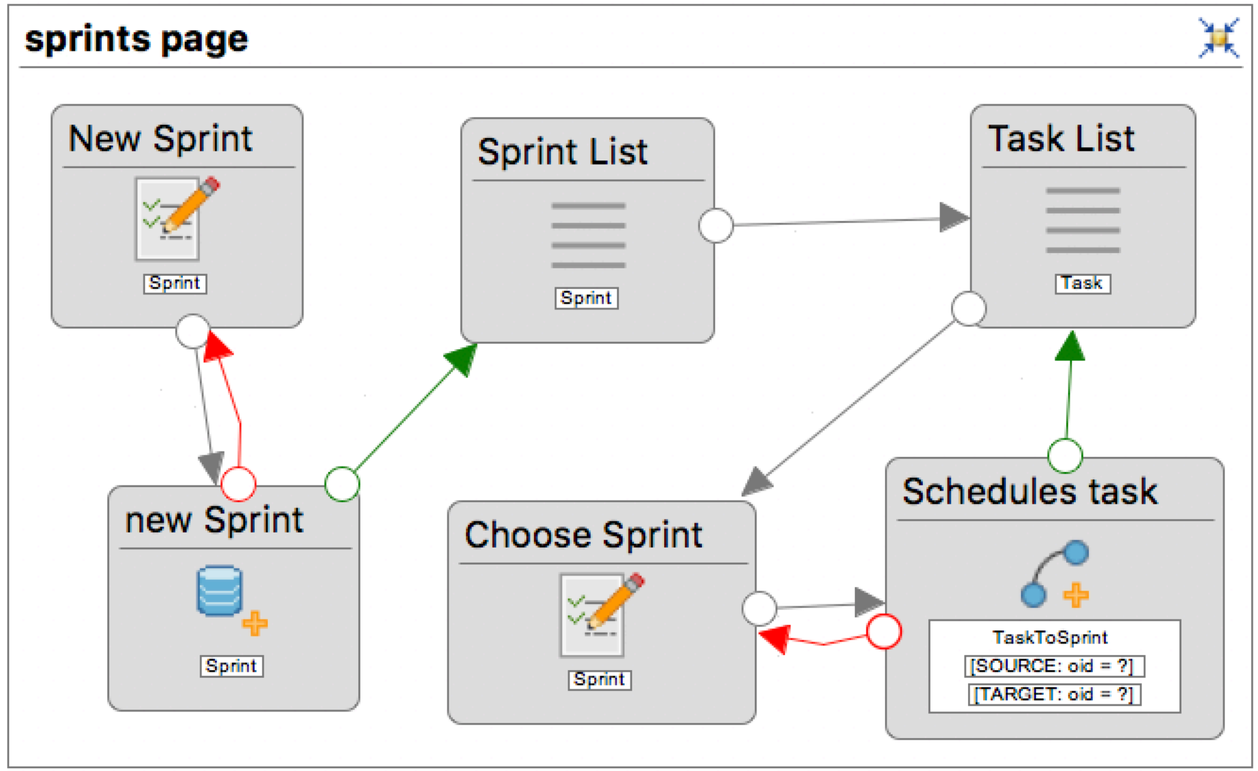
IFML Specification

- A user selects a task from the backlog, assigns it to a (another) sprint and sees its name in the sprint task list.



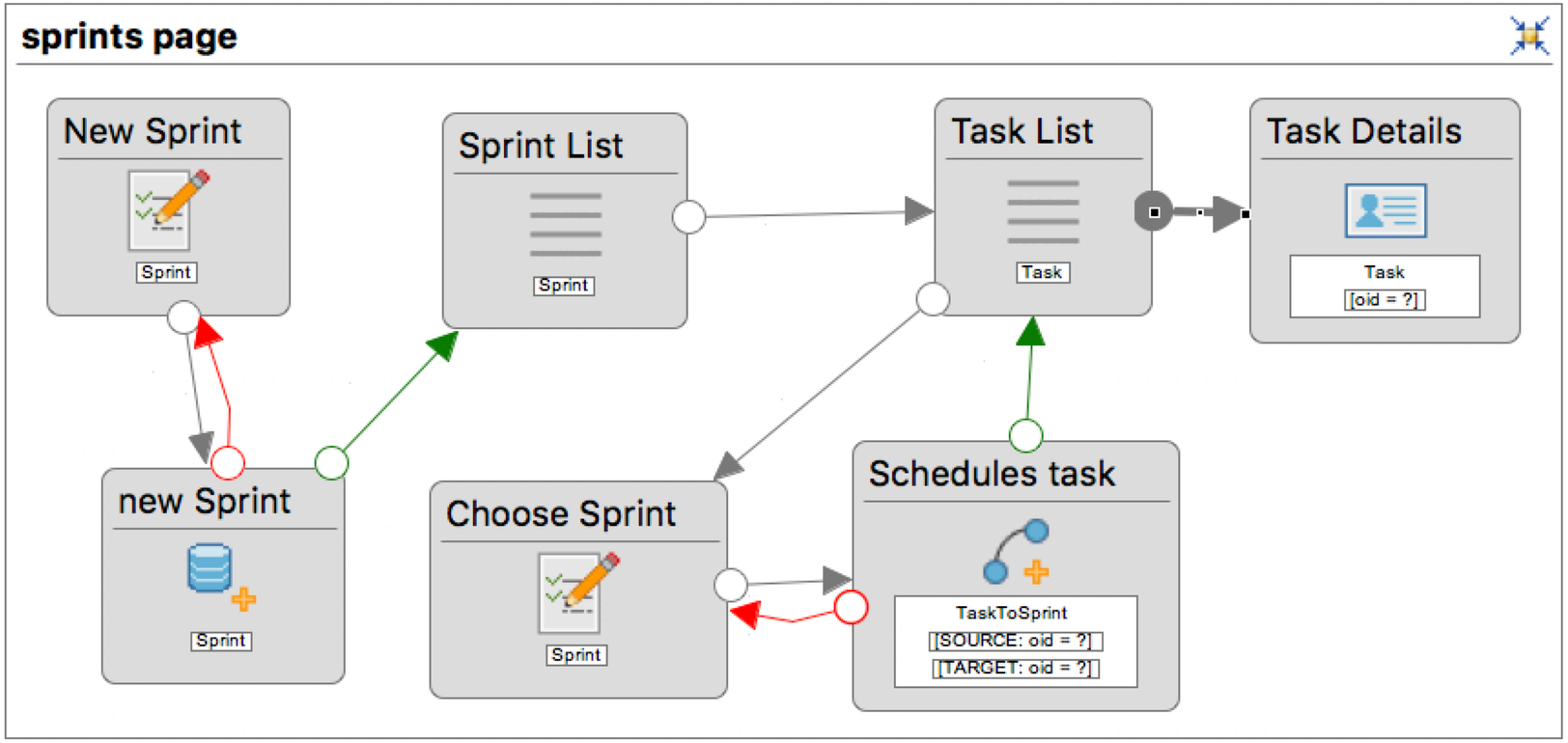
IFML Specification

- A user selects a task from a sprint, assigns it to a (another) sprint and sees its name in the sprint task list.



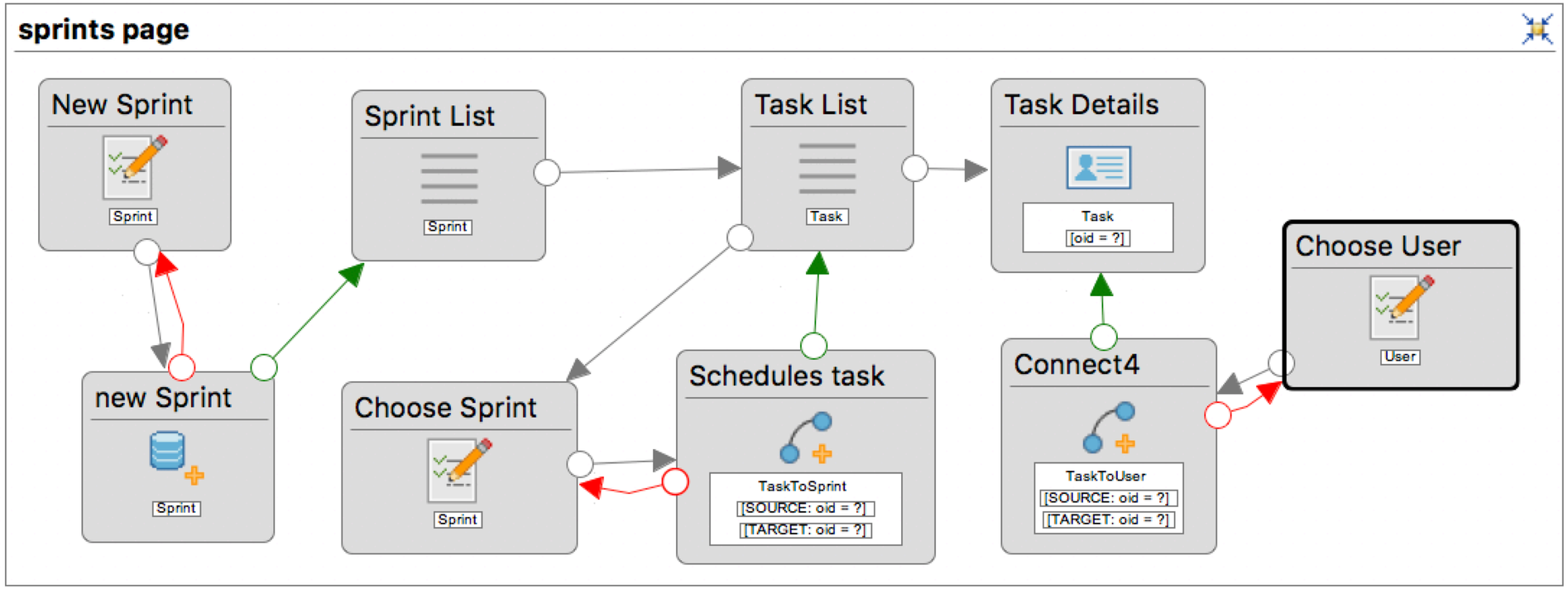
IFML Specification

- A user selects a task from a sprint, and sees its name, description, and dates.



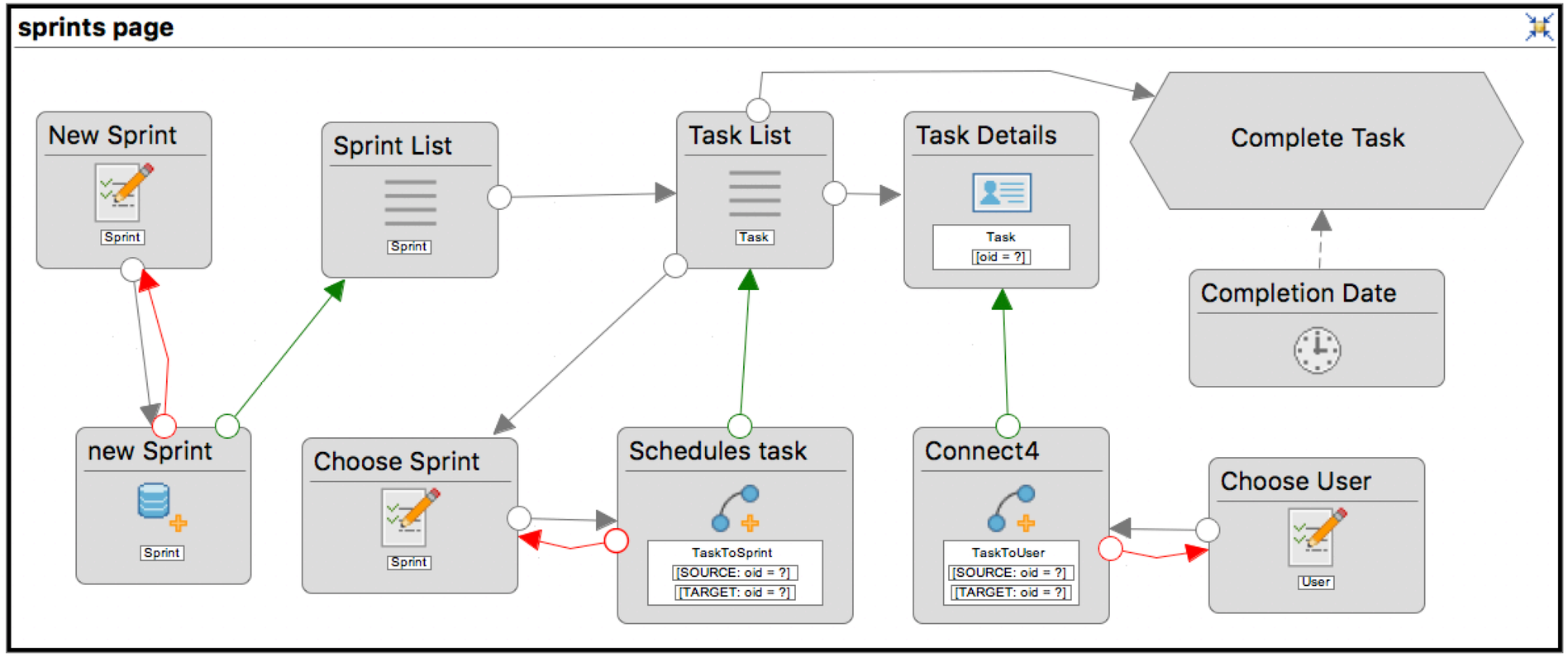
IFML Specification

- A user selects a task from a sprint, assigns it to a (another) user and sees the result in the details of the task and on the original list.
-



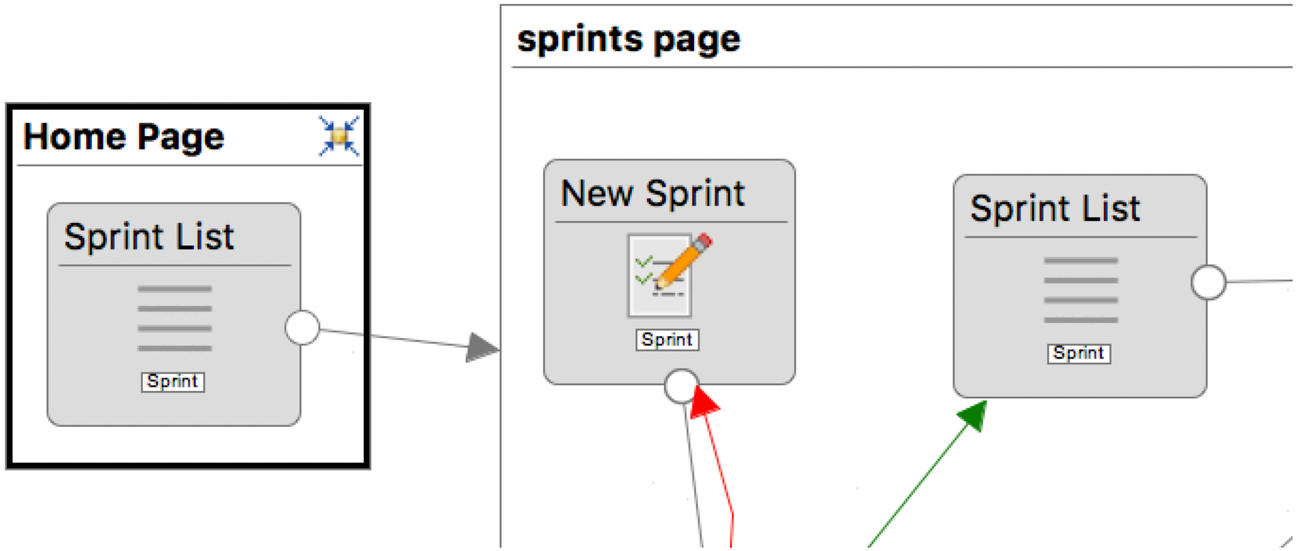
IFML Specification

- A user selects a task from a sprint, completes it (in a given date) and sees the end date in original list of tasks.



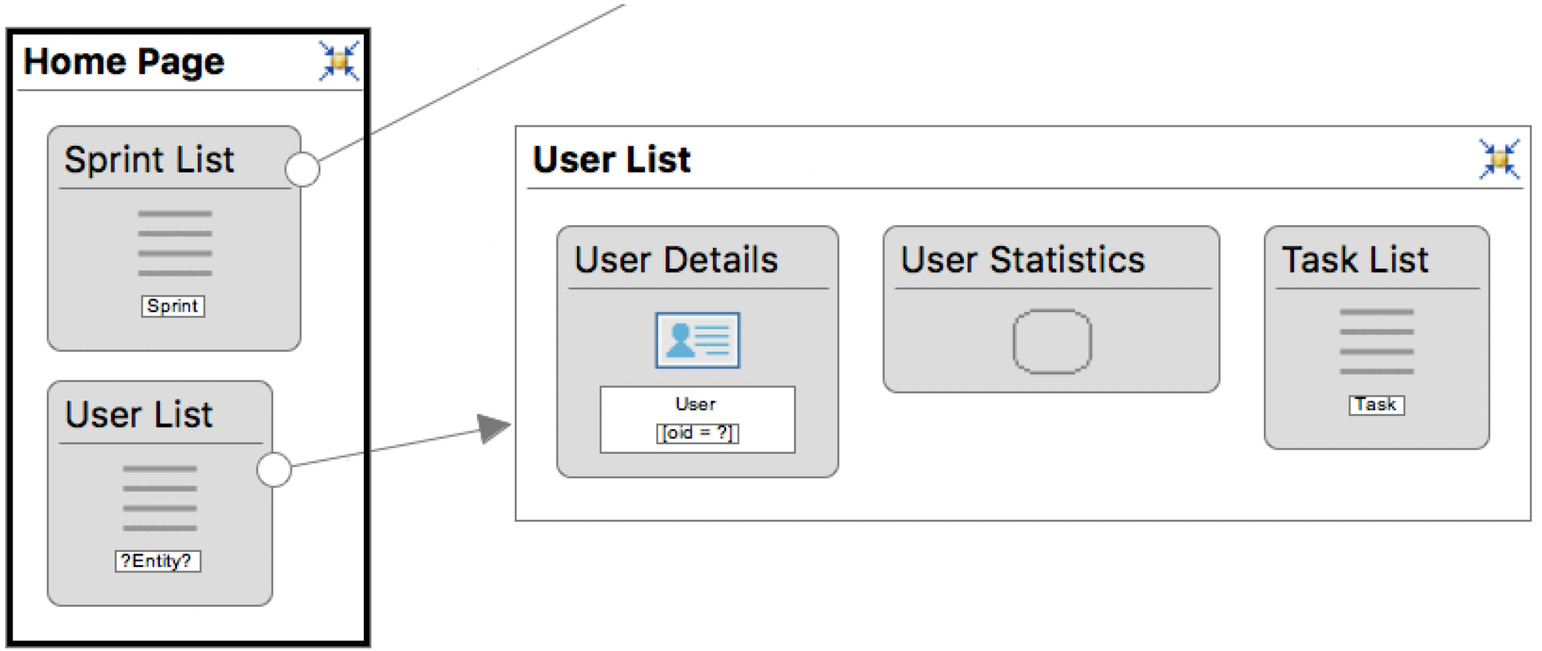
IFML Specification

- A user opens its home page, selects a sprint, and sees the sprint details, statistics and task list.



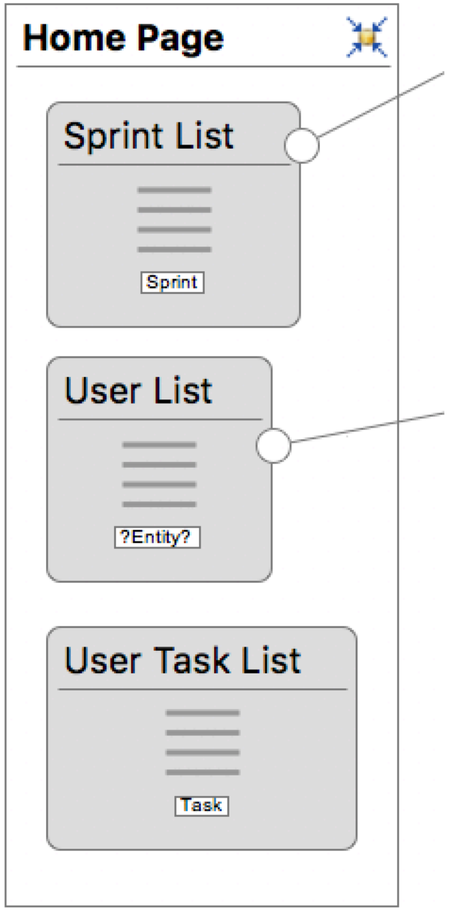
IFML Specification

- A user opens its home page, selects a user, and sees the user details, statistics and tasks list.



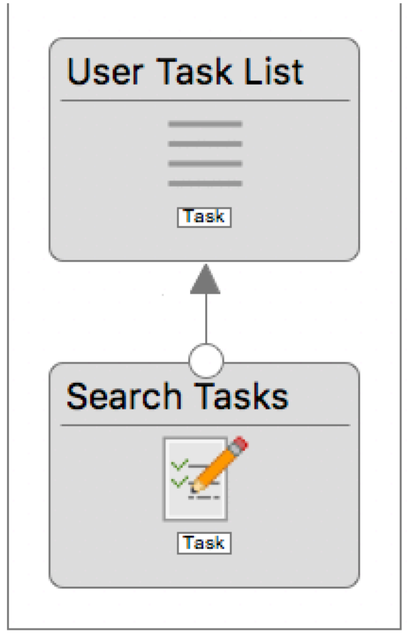
IFML Specification

- A user opens its home page, selects the backlog and sees the task list.

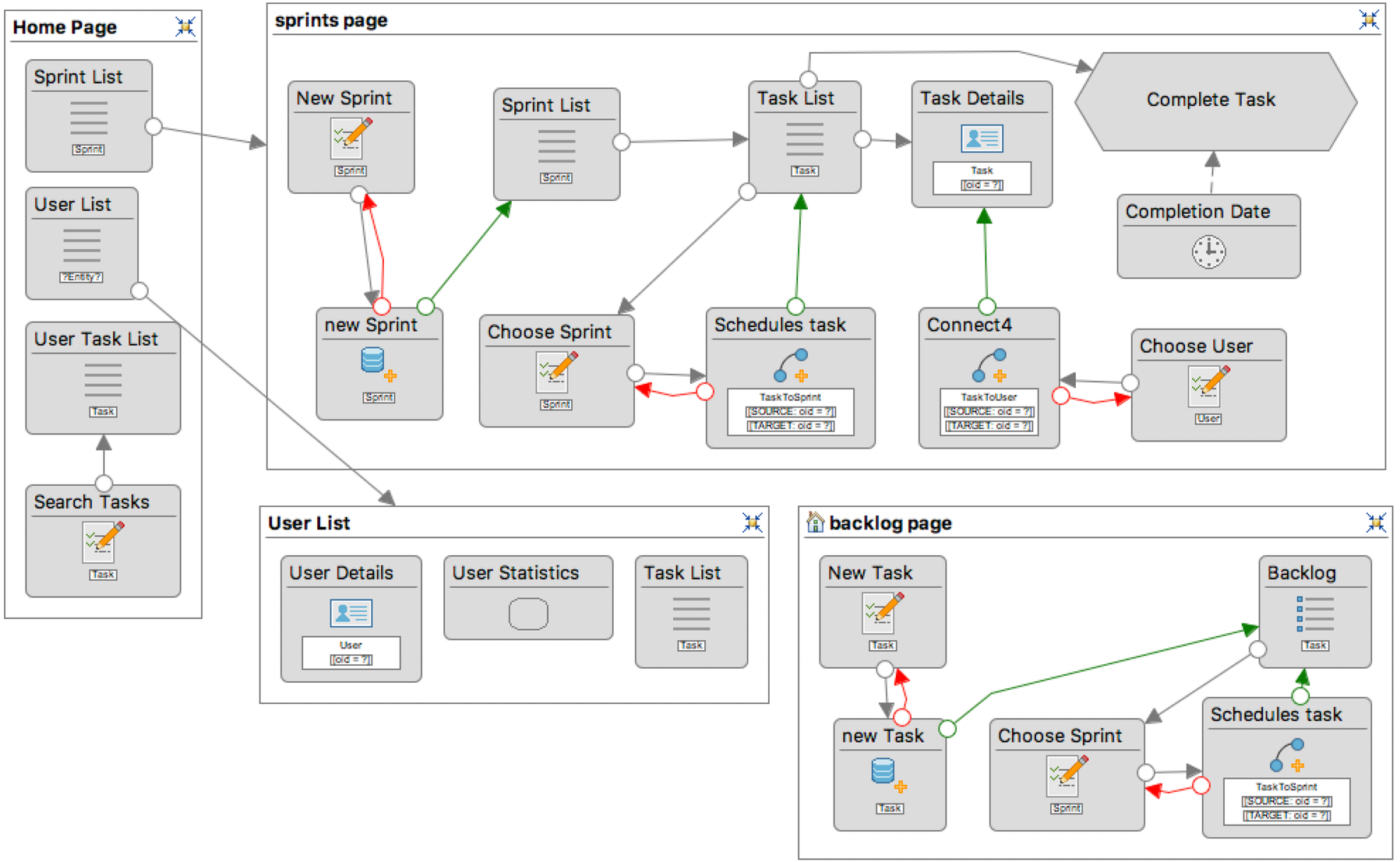


IFML Specification

- A user opens its home page, searches the backlog using a text and task properties, and sees the task list filtered by the given criteria.



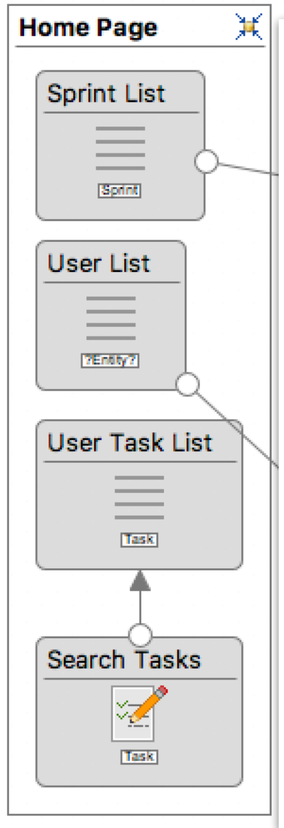
IFML Specification: The full picture



React structure

```
import * as React from "react";
import { BrowserRouter as Router, Route, Link } from "react-router-dom";

export interface HelloProps { compiler: string; framework: string; }

export const Dashboard = () => <div>Dashboard</div>

export const BackLog = () => <div>BackLog</div>

export const Sprints = () => <div>Sprints</div>

export const Users = () => <div>Users</div>

export const App = () =>
    <Router>
        <div className="container">
            <h1>Task List</h1>
            <div>
                <ul>
                    <li><Link to="/"> Home </Link></li>
                    <li><Link to="/backlog"> BackLog </Link></li>
                    <li><Link to="/sprints"> Sprints </Link></li>
                    <li><Link to="/users"> Users </Link></li>
                </ul>
            </div>

            <div>
                <div>
                    <Route exact path="/" component={Dashboard}/>
                    <Route path="/backlog" component={BackLog}/>
                    <Route path="/sprints" component={Sprints}/>
                    <Route path="/users" component={Users}/>
                </div>
            </div>
        </div>
    </Router>
```

React structure




```jsx
const SearchCriteria = () =>
    <form>
        <input type="text" name="search"/>
        <input type="submit" value="Search"/>
    </form>

const SeacheableUserTaskList = () =>
    <div>
        <TaskList/>
        <SearchCriteria/>
    </div>

const Dashboard = () =>
    <div>
        <h1>Dashboard</h1>
        <SprintList/>
        <UserList/>
        <SeacheableUserTaskList/>
    </div>
```

```jsx
const UserList = () =>
    <div>
        <h2>Users</h2>
        <ul>
            <li>User 1</li>
            <li>User 2</li>
            <li>User 3</li>
            <li>User 4</li>
        </ul>
    </div>

const SprintList = () =>
    <div>
        <ul>
            <li>Sprint 1</li>
            <li>Sprint 2</li>
            <li>Sprint 3</li>
            <li>Sprint 4</li>
        </ul>
    </div>

const TaskList = () =>
    <div>
        <h2>Tasks</h2>
        <ul>
            <li>Task 1</li>
            <li>Task 2</li>
            <li>Task 3</li>
            <li>Task 4</li>
        </ul>
    </div>
```

React structure

```
export class App extends Component<any,any> {

    constructor(props: any) {
        super(props);
        this.state = {
            sprints: [],
            tasks: []
        }
    }

    render() {
        return (
            <Router>
                <div className="container">
                    <h1>Task List</h1>
                    <div>
                        <ul>
                            <li><Link to="/"> Home </Link></li>
                            <li><Link to="/sprints"> Sprints </Link></li>
                            <li><Link to="/backlog"> BackLog </Link></li>
                            <li><Link to="/users"> Users </Link></li>
                        </ul>
                    </div>

                    <div>
                        <div>
                            <Route exact path="/"
                                   component={() => <DashboardPage sprints={this.state.sprints}/>}/>
                            <Route path="/sprints" component={Sprints}/>
                            <Route path="/backlog" component={BackLogPage}/>
                            <Route path="/users" component={UsersPage}/>
                        </div>
                    </div>
                </div>
            </Router>);
    }
}
```

Summary

Pros and Cons of using specs and frameworks

- Formal Design and Method
- Communication between stakeholders
- Documentation for future reference
  - It’s important to use tools that sync specs and code
- Maintenance and evolution
  - Allows the analysis of the impact of code evolutions