# CS314 Software Engineering

## User Stories

---

Dave Matthews

---

# CMMI for Development Model

<table>
<thead>
<tr>
<th>Maturity</th>
<th>Process</th>
<th>Project</th>
<th>Engineering</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>• Organizational Performance Management</td>
<td></td>
<td>• Requirements Development</td>
<td>• Causal Analysis and Resolution</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>• Quantitative Project Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>• Organizational Process Definition</td>
<td>• Integrated Project Management</td>
<td>• Requirements Development</td>
<td>• Decision Analysis and Resolution</td>
</tr>
<tr>
<td></td>
<td>• Organizational Process Focus</td>
<td>• Risk Management</td>
<td>• Technical Solution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Organizational Training</td>
<td>• Product Integration</td>
<td>• Product Integration</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Verification</td>
<td>• Verification</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Validation</td>
<td>• Validation</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>• Requirements Management</td>
<td></td>
<td></td>
<td>• Configuration Management</td>
</tr>
<tr>
<td></td>
<td>• Project Planning</td>
<td></td>
<td></td>
<td>• Measurement and Analysis</td>
</tr>
<tr>
<td></td>
<td>• Project Monitoring and Control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Supplier Agreement Management</td>
<td></td>
<td></td>
<td>• Process and Product Quality Assurance</td>
</tr>
</tbody>
</table>
User Story

A brief statement of intent that describes something the system needs to do for the user.
User Story

- As a `<role>`, I can `<activity>` so that `<business value>`
  - `<role>` represents who performs the activity or receiving the value, even another system.
  - `<activity>` represents the action to be performed
  - `<business value>` represents the value achieved by the activity
- Stories appear in the product backlog
- Acceptance criteria are associated with a story
- Tasks are associated with a story

User Story models

```
BacklogItem

Is One Of

UserStory 1 implemented by Task 1..*

done when passes

1..*

AcceptanceTest
```

```
UserStory
+role : String
+activity : String
+businessValue : String
```
User Story

- Negotiable expression of intent, not detailed requirements
- Short, easy to read, understandable to all
- Represent small increments of valued functionality that can be developed in a short period
- Relatively easy to estimate
- Organized in easily arranged lists, rearranged as needed
- Elaborated on as needed, avoiding early specificity
- Require little or no maintenance, safely discarded when done
- Serve as inputs to documentation

User Story Example

- As a consumer, I want to be able to see my daily energy usage so that I can lower my energy costs and usage.
- Acceptance Criteria
  - Read Decawatt meter data every 10 seconds and display on portal in 15-minute increments and display on in-home display every read.
  - Read KiloWatt meters for new data as available and display on the portal every hour and on the in-home display every read.
  - No multiday trending for now.
INVEST model

- Independent – self contained, no inherent dependencies
- Negotiable – not contracts, leave space for discussion
- Valuable – delivers value to the stakeholders
- Estimable – must be able to estimate the size
- Small – possible to plan/task/prioritize with some certainty
- Testable – provides necessary information to create tests

Splitting User Stories

- Workflow steps – implement in incremental stages
- Business rule variations – break into several stories to handle complexity
- Major effort – break into several parts, with most effort in the first part
- Complex/simple – break complex stories down, ask what’s the simple version
- Data variations – build the simplest version, add variations and sources
- Data entry methods – build the simplest UI first, richer UI later
- Defer system qualities – do simple first, add other “-ilities” later
- Operations – split vague operations (manage, control) into multiple operations
- Use-case scenarios – split complex user to system and system to system interactions
- Break out a spike – split to gain better technical or functional understanding
Tasks

• Smallest unit of work necessary for completion of a story.
• Multiple task associated with a single story
• Have an owner and an estimate
• Assures teams understand the work to be done and can meet their commitments

Acceptance Tests

• Functional tests
• Automated whenever possible
• Confirm the story has been implemented correctly
GitHub / Git Usage

• GitHub
  – On the internet, holds the master repo and issues
  – Should always run with no problems
  – No changes are made here
  – Proposed changes from remote reviewed, approved, merged

• Git
  – On a remote/local machine
  – Holds a clone of the master
  – Changes made in branches pushed back to the GitHub master

GitHub Activity Diagram
GitHub – Graphs / Network

GitHub – select issue to work on

ProTip! Exclude your own issues with -author:davematt.csu.
Git (on local system)

- # install git on your local system and configure it (username, password)
- # make a directory for your local report and/or cd to it
- git clone [masterURL]  # first time
- git pull # get latest before creating a new branch
- git branch [newbranchname]
- git checkout [newbranchname]  # never master!
- git branch # verify which branch you are in
- # add directories, modify files, build and test
- git pull # merge changes from master into local
- git status # verify
- git add .  #
- git status # verify
- git pull # merge changes from master into local
- # make sure it still builds and tests
- git commit –m “closes #999”
- git push origin [newbranchname]

GitHub Pull Request

- Create a pull request using the branch you just pushed
GitHub Pull Request

- View pull request
- Add comments
- Address comments
- Merge pull request

GitHub Confirm Merge
GitHub
Done