# Relate - **SCM, Version control, Jira, and Agile Development**

While reading this paper, I kept thinking back to the beginning of my internship with Clearwater and comparing/contrasting SCM with the methods that Clearwater uses. Clearwater practices its own blend of Agile development, and while it’s not explicitly an implementation of SCM, there are a lot of similarities.

For example, we implement SCIs, DOs, and ECP’s using an Atlassian program called Jira and couple that with a version control system, SVN. Together, these two programs act as our PSL/CCC, and Software Configuration Status Accounting.

At the beginning of a project we create an Epic case were we detail the requirements of the case and do early design work. As we design and discuss the project we revise this epic, and Jira keeps a history of each revision. This is not unlike an SCI. It differs that instead of creating a new SCI/Epic for each revision, we keep a single Epic that is revised and stores a history.

For each feature, improvement, technical task or bug fix we want to implement, we create a Jira case detailing what change we want to make. This is similar to ECPs. Our desired changes are documented before any work is begun.

Every week at the beginning of the week we hold a sprint planning where we as a team (including developers, our project manager, and our team lead), become the SCCB for the project. We review each case, deciding whether or not to accept it and begin working on it. For those we accept, we discuss the difficulties of the case, adding additional details to the case as required.

Once we have one week’s worth of work approved, we begin working on the cases. Every change we make is committed to a central repository where every change to the code is recorded. The Jira cases are updated with links showing the differences between the code at the beginning of the case and its current state. This is similar to the ongoing SCI’s in SCM.

Taken together, these steps implement each important aspect of Software Configuration Management.

# 2 Disagreement

“(The dividends of software auditing) include the avoidance of costly retrofits resulting from problems such as the appearance of new requirements and the discovery of major design flaws.” [1]

I disagree that Software auditing prevents the appearance of new requirements. As a developer, I sometimes wish I could completely lock down the projects I work on to prevent my software from suffering from the changes of opinion and requirements from the software’s stakeholders. While this system provides good pushback against whimsical changes, the truth of the matter is that requirements will unavoidably change during the course of development. This truth is evidenced by the creation of Agile development methodologies which focus on welcoming and adapting to changing requirements from stakeholders [2]. One of the principles behind the Agile Manifesto is

“Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.” [2]

[1] E. Bersoff, "Elements of software configuration management," *IEEE Trans. on Softw. Eng.*, vol. SE-10, no. 1, pp. 79-87, Jan. 1984,   
[Online] Available: [http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5010202&isnumber=5010187](https://content.byui.edu/items/8bcbc45e-012a-48e0-800d-7082dd962f15/1/?.vi=file&attachment.uuid=ebf74848-b4af-4336-8a5b-c5f1c7aab5fc)

[2] Fowler, Martin, and Jim Highsmith. "The agile manifesto." *Software Development* 9, no. 8 (2001): 28-35. [Online] Available: http://agilemanifesto.org/principles.html