O Artigo citado relata o “Dependency Hell” ocorrido no caso Jarbas, onde utilizando um arquivo “requeriments.txt que tinha a funcionalidade de checas se as bibliotecas listadas estão na sua versão mais recente, onde se as bibliotecas estiverem atualizadas abre-se um pull request no repositório atualizando todas as versões de dependências do arquivo.

No caso Jarbas após a realização de debugs ficou clara que o que tinha quebrado o código foi algo de fora do projeto, porem para descobri e confirmar as suspeitas seria necessário a reinstalação de um pacote anterior a versar 0.4.0, algo que acabou confirmando as suspeitas.

Neste caso claramente tornou-se possível facilmente acompanhar as mudanças realizadas no código através do versionamento, pois consiste em estabeleces por meio de numerações diferentes que permitiria aos programadores acompanhar as mudanças aplicadas e os usuários finais identificarem as novidades da versão mais recente.

Conforme o artigo citado aponta-se também a importância de um versionamento semântico que auxiliaria no tipo de correção, implementação ou atualização realizada no código que mantem um histórico de commits como o tipo de alteração considerando a versão definida no release e:

(O release teve *1 major*, *1 menor* e *4 patches*) v2.1.4 - Release 04

1. versão Maior(MAJOR): quando fizer mudanças incompatíveis na API,
2. versão Menor(MINOR): quando adicionar funcionalidades mantendo compatibilidade, e
3. versão de Correção(PATCH): quando corrigir falhas mantendo compatibilidade.

Rótulos adicionais para pré-lançamento(pré-release) e metadados de construção(build) estão disponíveis como extensão ao formato MAJOR.MINOR.PATCH.

De acordo com o artigo onde cita que o co-fundador do git hub chama o versionamento semântico de um conjunto de regras e boas práticas no desenvolvimento de software, o conjunto de boas práticas alinhado com ferramentas de versionamento tais como o próprio git hub podem trazem um enorme ganho em tempo de desenvolvimento e manutenção de software.