Single-Responsibility Versus Needless Complexity by Ray Houston

At Pablo's Day of TDD, we were discussing the Single-Responsibility Principle (SRP) while working through one of the labs and a question came up about a piece of code. The code in question looked something like the following (warning: this is over simplified code to show a point):

```
public bool Login(string username, string password)
{
   var user = userRepo.GetUserByUsername(username);

   if(user == null)
      return false;

   if (loginValidator.IsValid(user, password))
      return true;

   user.FailedLoginAttempts++;

   if (user.FailedLoginAttempts >= 3)
      user.LockedOut = true;

   return false;
}
```

This was from the LoginService class and this was its only method. The question was whether or not this violates SRP. It appears to have multiple responsibilities, it is in charge of incrementing FailLoginAttempts as well as locking the user out after 3 failed attempts. I believe we answered the question with a "depends", but it bothered me that we didn't have a better answer. Personally, I wouldn't have busted this up into another class, but I didn't have a good argument to stand on.

Today I went searching through Agile Principle, Patterns, and Practices in C# looking for a better answer. In the chapter on SRP, the book gives an example of an interface of a modem that can Dial/Hang-up and Send/Receive. The former is connection management and the later is data communication. The book asks the question as to whether or not these responsibilities should be separated. The answer is

"That depends on how the application is changing."

It then gives an example to how a change might violate SRP, but then states:

"If, on the other hand, the application is not changing in ways that cause the two responsibilities to change at different times, there is no need to separate them. Indeed, separating them would smell of needless complexity."

Ah, there's the backup wisdom that I needed to validate my gut feeling. Here's one final quote from the book:

"There is a corollary here. An axis of change is an axis of change only if the changes occur. It is not wise to apply SRP (or any other principle, for that matter) if there is no symptom."

I think applying SRP is about using good judgment. You certainly don't want to wait until you have to make a change before you think about SRP. You also don't want to over do it either and end up with classes with one method, each having only a couple lines of code.