Kent Beck, Cynthia Andres (2004) *extreme programming explained - embrace change*. 2nd edition. p 67f

Single Code Base

There is only one code stream. You can develop in a temporary branch, but never let it live longer than a few hours.

Multiple code streams are an enormous source of waste in software development. I fix a defect in the currently deployed software. Then I have to retrofit the fix to all the other deployed versions and the active development branch. Then you find that my fix broke something you were working on and you interrupt me to fix my fix. And on and on.

There are legitimate reasons for having multiple versions of the source code active at one time. Sometimes, though, all that is at work is simple expedience, a micro-optimization taken without a view to the macro-consequences. If you have multiple code bases, put a plan in place for reducing them gradually. You can improve the build system to create several products from a single code base. You can move the variation into configuration files. Whatever you have to do, improve your process until you no longer need multiple versions of the code.

One of my clients had seven different code bases for seven different customers and it was costing them more than they could afford. Development was taking far longer than it used to. Programmers were creating far more defects than before. Programming just wasn't as fun as it had been initially. When I pointed out the costs of the multiple code bases and the impossibility of scaling such a practice, the client responded that they simply couldn't afford the work of reuniting the code. I couldn't convince the client to even try reducing from seven to six versions or adding the next customer as a variation of one of the existing versions.

Don't make more versions of your source code. Rather than add more code bases, fix the underlying design problem that is preventing you from running from a single code base. If you have a legitimate reason for having multiple versions, look at those reasons as assumptions to be challenged rather than absolutes. It might take a while to unravel deep assumptions, but that unraveling may open the door to the next round of improvement.