Sapient “Gotchas” | Common Pitfalls when moving to Agile

Moving your organization to Agile is akin to moving your organization to a new country with a new language and culture, with implicit and explicit differences from how you work today. No amount of training can prepare you for every challenge. We have compiled a list of the major “gotchas” that can derail a project with team members new to Agile. Work to proactively prevent these situations, and also look for these situations as they may occur.

Gotcha	Description
Lack of discipline	Although it may seem counter intuitive, Agile is an extremely disciplined approach to working. Agile does not equal sloppiness. Most people will have a difficult time adjusting to this. Tracking stories to closure, accounting for the velocity of an iteration, tracking one’s estimate to complete -- these are all things where every team will slip up the first few iterations. Without discipline Agile will not work. Setting good examples and continually following up are essential.
Business is not as usual	The business stakeholders involved must learn the process as it is very different for them. Often over-looked or avoided because the delivery team may also be ramping up on the methodology (and hence feel uncomfortable teaching it to others), if this group is not brought into the fold there will be major disconnects (in terminology, approach to situations, etc.). Once educated they will see the benefits of being able direct the development as it progresses. The business also needs to clearly understand the expectation that it will also be frustrating for them to see the “dirty laundry” being aired each iteration: unfinished work, team mistakes, and other issues that are often hidden in a methodology with long breaks between business review.
Iterative death spiral	People mis-interpret the term ‘iterative’ to mean that there is unlimited ability to revisit scope. In fact, Agile success hinges on incremental completion of scope throughout the project. Teams that do not effectively define their “exit criteria” for each work unit (known as “story”), will never get real closure on work in progress. This leads to endless cycles of revisions that are really scope changes but which are not labeled so because of the mis-perception, causing delays, overruns, and a demoralized team. Iterative does imply that the process can support multiple iterations if they are intended. If not planned, an additional “iteration” on a screen flow, for example, is a scope change.
Design tunnel vision	A common mis-interpretation of the Elaboration Release (or Executable Architecture Release) is that the technical design of the entire system can be split into silos and each of these developed sequentially (with the Elaboration release focusing on a couple of these). This is the XP model of “diving in” to coding, but causes major issues with most new systems because there needs to be a level of technical design of the entire application completed before development begins. The impact of not doing high-level design is major redesign and rework of previously-built modules, when any problems are encountered in the design of a new module. Or, worse, the team may proceed with a poorly architected solution due to time pressures.
Testing undervalued	Often the need to do continuous QA on an application is not appreciated, and the initial releases will see not enough emphasis on testing. An effective test plan must be created, and things like data creation, environment setup, etc., must all be addressed very early in the project. If these steps are not taken, then it becomes essentially impossible to execute the tests (which are part of the exit criteria for a story), and hence the stories cannot be completed and delivered. This undermines the entire premise of incremental delivery and evolutionary design, causing most of the benefits of agile to be lost.
Playing without a coach	In a typical project the pressures of learning the business, addressing technology, business, and team personality issues, will very quickly overwhelm even the most prepared person. Adding upon this a major change in delivery methodology (even if only in certain areas) will lead one to revert to known approaches. The volume of information is simply too great. The new techniques will be dropped and mis-interpreted. A major mitigation here is training and more importantly, effective support from someone with actual experience in the field.
Blame game	Agile can be a scapegoat for initiatives gone wrong. Because Agile is very good at surfacing issues very early in a project the first reaction can be to attribute these to Agile. Careful follow up and root cause analysis can mitigate this challenge.
Working style	Developers used to working in a solo mode will have a bit of a hump to get over when working in small teams to complete stories. This can manifest itself in stalled work, as communication reverts to inefficient mechanisms such as email (instead of sitting together, or picking up the phone).
Automated testing	Automated testing is fundamental to quality, short delivery cycles, and hence the agile model. Yet, there are many barriers: taking on a legacy application with no existing test suites; lack of tools for many aspects of an application; lack of team knowledge on how to do this. People will want to revert to more comfortable manual testing approaches.
TDD	Test-Driven Development will be a struggle to adopt. It is counter-intuitive unless tried; also, many software situations -- existing code base, WYSIWYG tools, etc. -- don’t really lend themselves to doing it.

Sapient®

Gotcha	Description
Metrics are hard	Metrics are essential to tracking the success of the methodology, from adoption to its downstream effects on quality, productivity, time to market, and others. You must understand what is working, where, and what is not working, so that you can focus improvements and/or support where it is needed. But successfully tracking and analyzing metrics is challenging. Even with a well-planned approach to minimize cost and data capture problems, the problem remains of rationalizing any cost or overhead to an individual project, because in most cases the true value is at the organizational level. A major mitigator is to provide tool support that effectively adds little or no overhead -- essentially giving ”free” metrics. Even with a tool, some metrics such as pre-live defects may simply be too difficult to track and hence are not worth it.
Taking off the blinders	Developers working in a very focused environment within a larger phase may find themselves struggling within a short iteration to get involved in a much broader range of skills: estimation, design, test design, development, testing, and finally demonstrating functionality to the business. This has tremendous upside in terms of learning new skills, etc., but can be very stressful.
Who needs tools?	Without tools, some fundamental aspects such as velocity tracking, estimating, and tracking ETC become much more labor-intensive. The steep learning curve is enough as it is -- with no tools, the adoption curve will be simply too steep (even if the team can see direct value for their project).
Project Lead Mindset shift	Given focus on transparency and on pushing responsibility to the team, the Project Lead will be less of a task manager, and more of a problem solver, and will have to “let go” of a lot of previously-held control. He/she will have constant unavoidable visibility into problems. There is an increased emphasis on honesty, openness, and trust that may feel very different. Operating in an adaptive planning environment will feel very uncomfortable to many.
Project Lead Skillset shift	Project Lead skills shift notably. The focus shifts to be more outward, to the business. Lead must have more polished skills of expectation management with business, as there is much more frequent interaction and transparency. Relationship skills are more important than being a hard task-master. The Lead will shift ownership to his team and focus less on “command and control,” and more on developers’ skills of working with the business
Developer role shift	Developers will work hand in hand with testers and cannot declare a piece of code “done” until the tester gives the green light. Also, because there may not be enough testers in the organization, developers need to step in and play that role (in terms of writing scripts, etc.).
Tester role shift	Testers who are used to having a more clearly specified set of tasks will struggle, as the Agile tester role involves more thought, planning, and business knowledge. The tester becomes a true partner with the developers in building the application.
Business Analyst role shift	Working just in time on requirements will feel very unnatural. Working hand in hand with developers to define requirements and then to help test them as a developer works will feel very foreign. This problem will be exacerbated by the dual reporting structures that exist between developers and BAs.
Technical Lead role shift	The difference between a waterfall Design and the Elaboration Release (or Executable Architecture Release) is notable -- in Agile the technical lead has a lot of freedom to decide the level of depth to go to for the project scope. Some technical leads may not have sufficient experience to comfortably operate in this more ambiguous situation. Technical Leads will also need to support their teams more with challenging concepts such as Test Driven Development.

Sapient®