**Scrum Team Audit Questions**

adopted from "Labirynty Scruma" by Jacek Wieczorek

| Problem | Identification |
|---|---|
| **Scrum Roles** | |
| Scrum Master lacks experience | The SM has this role for the first time, has little experience, and has never seen a well-implemented Scrum. SM doesn't understand the key notions (e.g. servant leadership, self-organization, change management models) and doesn't use tools (e.g. moderating, information visualization, timeboxing) which are useful in daily work. |
| Scrum Master behaves like a secretary | The SM functions as a scribe: takes the notes for the team, creates paper User Stories, writes down what has been arranged at meetings, updates Scrum charts and graphs. When the data must be entered in electronic tools (e.g. Confluence, JIRA), the SM does it. What's more, the SM takes responsibility for organizing meetings, sending invitations, booking, preparing, and ordering rooms after meetings. The SM is the team's runner/courier. |
| Scrum Master behaves like a project manager | The SM manages the team as if s/he was a project manager: assigns tasks, manages the product scope, organizes status meetings for team members. The SM takes care of 100% "resource utilization", as some people call it, making sure everyone has a task at hand. The SM assumes the role of SPOC (single point of contact), being the only point of entry and acquisition of crucial information relating to stakeholders, clients, or users of the product. The SM doesn't perform tasks that are usually associated with this function. The SM perceives product development as a project, focusing mainly on the scope, time, and budget of the initiative. |
| Scrum Master behaves like a coach | The SM answers literally EACH question asked by the team with another question. |

| | |
|---|---|
| Scrum Master works in several teams at once | The SM fulfills his/her role in multiple – at least three – Scrum Teams. The context is important in this trap: they are not mature, experienced teams but rather, beginner teams. |
| Scrum Master works in several roles at once | The SM undertakes additional tasks resulting from other functions, for example being simultaneously a SM, line manager, PO, customer care liaison, tester, etc. |
| Scrum Master is a temporary role | The SM is a rotational role in the team – changing after each fixed period, e.g. after every Sprint. There is no regular SM. |
| Scrum Master acts as a go-between | The SM is like a messenger carrying all sorts of information. Team members don't communicate directly with interested parties (the PO, stakeholders, or other colleagues) but instead use the SM to do so. |
| Scrum Master is not supported by the organization | The SM has no actual impact on the company. S/he cannot find authorized partners in management roles ready to solve the organization's real problems. The employees don't understand the meaning of the SM's role and don't know what to expect from it. When it comes to change management, the SM works with the status quo and receives no or insufficient support to continue the change process. The company's CEO, director, or managers remain silent when they should speak up, take a stand, and authorize changes in the organization. |
| Product Owner lacks experience | The PO has this role for the first time, has little experience, and has never seen a well-implemented Scrum. The PO doesn't know the key notions (e.g. process discovery, MVP, iterative product development) and doesn't use the tools (Story Mapping, MoSCoW method, Product Backlog Refinement) which are useful in daily work. |

| | |
|---|---|
| Product Owner does not present a clear vision | The PO doesn't know the vision of the product s/he is building or doesn't communicate it clearly enough. Nobody can say why the product is being developed or what the expected result of the work is. The PO doesn't use the vision during meetings with stakeholders and the development team nor in their daily work. The PO is unable to suggest a sensible Sprint Goal because s/he has no idea where the product is heading or why. |
| Product Owner is not decisive | The PO doesn't make any independent decisions and forwards most of the development team's questions to other people, pushing the decision onto them. This might result from the current structure of the company, a lack of trust in the PO, a lack of understanding of the PO's role, a lack of product knowledge or proper tools, or simply be due to a personality trait. |
| Product Owner does not talk about business context | Talking about the product, the PO says what s/he wants in the product but not why this is important. The Product Backlog elements in the form of User Stories only describe the user and the functionality (using the phrase "I want…"). The business context is not presented during Scrum events and in daily work . |
| Product Owner has no time for the team | The PO has no time for the team. When crucial questions concerning the product arise, the development team cannot contact the PO. The PO keeps postponing Scrum events or tries to avoid taking part in them, so they often happen without his/her presence. |
| Product Owner is too technical | The PO takes decisions related to the product's architecture and technical aspects. This may be a result of previous experience as a programmer, tech lead, or architect. |
| Product Owner does not speak with users | The PO doesn't hold a dialogue with users, making decisions concerning the product basing solely on his/her impressions. The context of user needs rarely appears in talks about the product. |

| | |
|---|---|
| Development Team members do not cooperate with each other | Each member of the team works on a separate element of the Product Backlog. There are only occasional instances of cooperation among them. |
| Development Team has internal skill silos | Members of the team are specialized in narrow fields and never go beyond them, saying things like, “I’m a programmer, not a tester”, “This a job for an analyst”, “It’s Tom who does CSS here”. |
| Development Team does not automate tests | There are only manual tests. The team doesn’t create automated unit, integration, or user interface tests. |
| Development Team members do not organize their work on their own | All the decisions about who is going to do a task and how are made by only one person, for instance one of the developers, the PO, or the SM. Worse still, the decisions are made by someone who is not a member of the Scrum Team: a line manager, project manager, or architect. |
| Development Team members are only partially available | Members of the development team belong to other Scrum Teams. Their availability is expressed in terms of percent per Sprint. Even worse, there is a frequent rotation of team members – some persons permanently leave the team, others join it. |
| Development Team works behind Product Owner’s back | The PO knows only about part of the work done by the team. They perform additional work commissioned by the client, stakeholders, other development teams, or generated by themselves. |
| Development Team is too big | The meetings of team members are not effective. The decision-making process concerning products, in particular their technical aspects, is very slow. Even though moderated by the SM, Scrum events are noisy and disordered. |
| **Scrum Events** | |

| | |
|---|---|
| Scrum Events are not regular | Scrum events – Daily Scrum, Sprint Planning, Sprint Review, Sprint Retrospective – are not regular. In a worse version of this trap, some events (Daily Scrum or Sprint Retrospective) don't take place at all. |
| Development Team realizes Sprint Zero | The team realizes a Sprint – or worse, a couple of Sprints – is not resulting in a product that can be potentially released. The Sprint is comprised of tasks related to workplace organization, work environment configuration, architecture preparation, and to collecting stakeholders' expectations. At the end of the Sprint, there are no visible results in the form of even a small component of a functioning product. |

| | |
|---|---|
| Tests are not performed during Sprints | The team alternates Sprints – sometimes referring to development, sometimes to testing. During development Sprints, team members don't perform any tests, only create code. During test Sprints, they identify errors and fix them. It's not transparent what developers do during test Sprints and what testers do during development Sprints. Another variant of this trap is realizing Sprints in which testers start testing only when the given element of the Product Backlog has been written by the developers. This system of working is like returning to the waterfall model disguised as Scrum. Very frequently, due to bad planning, this happens on the last day of the Sprint, so testing gets automatically put off until the next Sprint or Sprints. |
| The work planned for a Sprint is chronically endless | The team regularly fails to deliver many features chosen from the Product Backlog for a particular Sprint. For example, they only complete 2 or 3 elements out the planned 10. Every Sprint, a large number of elements is unfinished and automatically moved to the following Sprint. Sometimes these elements are postponed for several Sprints. |

| | |
|---|---|
| Sprints are prolonged according to needs | The Sprints don’t have any fixed predictable length. According to their needs, team members prolong the Sprints to finish the planned works. |
| Development Team does not use metrics | Team members don’t use metrics. They don’t collect, analyze, or use data such as the Team Pace, Planning Predictability, or the number of recognized unsolved errors in the product. |
| Development Team realizes a project, not a product | Sprints are treated as containers with predefined project requirements. The main focus is on realizing specific goals, especially on delivering the promised scope of works before the deadline. There is no adaptive thinking, discussing the value for the end user, or quick and clean verification of whether the product meets the expectations placed on it. |
| Too many tasks in progress | A lot of tasks are processed simultaneously. Each team member carries out a couple of tasks at the same time. However, only a few tasks are actually completed and ready for implementation. |
| Development Team members answer the three questions mechanically | One by one, team members respond to the three questions indicated in the Scrum Guide. Sometimes, their replies are very general and don’t provide any valuable information. There is no additional interaction between team members going beyond the above mentioned questions. |
| Development Team members do not track work progress | Team members don’t track the progress of works during Daily Scrums. Discussions about progress are enigmatic, very general, and don’t refer to the future actions of the team. The team cannot specify clearly how much work is left to do or whether the Sprint Goal has been reached. |

| | |
|---|---|
| Development Team members over-report the state of the work | Team members give detailed reports about the Sprint to the SM, the PO, the client, the manager, or another person. The team is concentrated on meeting the needs of this person, neglecting their own needs when it comes to updating the increment delivery plan. This is often related to maintaining eye contact with the person who is receiving the report. The physical orientation of the team towards that person is also visible: the team occupies one side of the room, and the recipient of the report is on the other side. In a worse scenario, the recipient must ask the team about the status of particular tasks, becoming in fact the leader of the meeting. |
| Development Team members do not concentrate on completing tasks | Team members talk about what they are doing or what they are going to do. They don't talk about the tasks they've finished or will finish soon. In the worst case, some tasks drag on for the whole Sprint or even for several Sprints. The team is focused on discussing the works in progress and not on what's been completed or how much work is left to reach the Sprint Goal. |
| Development Team members stray from the main subject of the meeting | Team members stray from the main subject of the Daily Scrum: i.e. work synchronization and planning the following 24 hours. Digressions occur and some or all team members follow them. |
| Product Increment Delivery Plan is not updated | During Daily Scrums, the team doesn't update the plan of reaching the Sprint Goal. The discussion doesn't influence the actual tactical plan of works for the following the day. The plan might not even have been created in the course of Sprint Planning. |
| Meetings start at random hours | The team has not settled on a regular time for Daily Scrums. Meetings begin when everyone gets to work. Sometimes there is no fixed hour, only a vague period, e.g. between 10 and 11 a.m. |

| | |
|---|---|
| Meetings are carried out through text messages | Team members don't talk during Daily Scrums. Their communication is based on text, using tools like Skype, Slack, or HipChat. |
| Not everyone actively participates in meetings | Some people don't take part in Daily Scrums. Some people don't say anything about the work they've done since the last meeting or speak in very general terms. |
| Product Backlog is refined instead of planning the work | During Sprint Planning, team members do things typical for the refinement of the Product Backlog: they create new elements listed in the Product Backlog, they expand and estimate them, adding new acceptance criteria or splitting them into smaller parts. |
| There is no Sprint Delivery Plan | The team carries out Sprint Planning but there is no tangible Sprint Delivery Plan. Selected features of the Product Backlog are changed into working software. |
| Sprint Delivery Plan is too rigid and detailed | The team plans the realization of particular functionalities of the Product Backlog in great detail. Team members spend a lot of time carefully devising each step of the Sprint. All of them have precise tasks for each day of the Sprint without any room for change (buffer) in case something goes wrong. |
| There is no Sprint Goal | Team members select the features of the Product Backlog they are planning to deliver in the upcoming Sprint, but the Sprint Goal is not defined so they have no clue what the final outcome of the iteration should be. |
| Sprint Goal is too general | The Sprint Goal is vague, both for the team and for stakeholders. It's so general that it proves impossible to determine what the result of the team's work should be. |
| Development Team plans by intuition | During Sprint Planning, the team ignores historical data concerning their efficiency and availability forecasts. The decisions regarding the choice and number of the Product Backlog features to be delivered are taken by intuition. |

| | |
|---|---|
| Only programmers take part in planning | Only the team members with programming skills take an active part in Sprint Planning. Other members, e.g. testers, UX, analysts, or DevOps are passive in meetings. |
| Stakeholders do not take part in meetings | Stakeholders don't take part in Sprint Reviews. Persons interested in the product development are missing (e.g. end users, management, other teams' representatives). The only people participating in the meeting are the development team, the SM, and the PO. |
| Only the demo session is carried out | Only the presentation part of the meeting is held. In the worst case, team members speak only about what they've done ("I've done task X", "I've done task Y") or present the completed functionalities of the product. Then the meeting is over so there's no chance for discussion about further development of the product. |

| | |
|---|---|
| Manual testing is performed | Such a session is like a software testing meeting. Team members present a large number of test scenarios, scrupulously verifying the quality of the created app. Stakeholders ask them to run detailed tests, e.g. "Enter a value with a comma in the field for the postal code". There's a lot of emphasis on checking whether the Product Backlog features have been realized correctly; there's no discussion about a broader context of the created product, though. |
| There is no value for users | The presented Sprint result brings no value to the project users. Technical aspects of the product are demonstrated: database charts, frameworks, classes, code fragments. To show the outcome of the current Sprint, unclear technical terms are used, understandable only to the development team. |

| | |
|---|---|
| Meetings have no schedule | The meeting has a chaotic structure. There's no clear schedule of events. Team members find it difficult to present product components fluently. A lot of time is lost in irrelevant activities (connecting computers with projectors or displays, preparing test data, switching between development environments, or deciding which member of the team is going to present the results of the Sprint). |
| Sprint Review is headed by Product Owner | The PO chairs the whole Sprint Review – welcoming the participants, presenting the agenda, speaking about the product increment, adding details, answering questions. In the meantime, the development team remains passive and silent. |
| Development Team does not present the actual product | During the Sprint Review, the created product is not presented. The focus is on presenting the Product Backlog features or low-level technical tasks. The main part of the meeting is devoted to viewing the electronic Product Backlog, e.g. in Jira or Excel. The meeting rather resembles a project status meeting. |
| Meetings have no structure | The meeting has no definite structure or agenda. There is no moderator or the moderator doesn't run the meeting properly. Timeboxes aren't used. The discussion has no cause-and-effect construction. Many topics are spoken about at the same time, but none is analyzed well enough to draw conclusions for the future. |
| The progress of intended works is not monitored | The team doesn't introduce changes in the process established during retrospectives. The progress of planned works is not monitored systematically. It is unclear which forecasted tasks have been completed and which haven't. Retrospectives don't refer to the planned activities. |

| | |
|---|---|
| Development Team hold over important issues until the Retrospective | The Scrum Team doesn't solve essential problems as they appear. Even if the problem is important and urgent, it must wait for the nearest Retrospective: in the case of long Sprints (two or three weeks) this takes up a lot of time. |
| Development Team ignores the elephant in the room | Team members talk about insignificant subjects, while the serious issues – even though everyone knows what they are – are omitted. |
| One person dominates during meetings | The meeting is dominated by one person who is speaking throughout most of the Retrospective. Other participants hardly say anything and are unable to break through with their ideas. |
| Planned works are defined in too general terms | The actions planned as outcomes of the Retrospective are very general and unmeasurable. |
| Most planned tasks are assigned to the Scrum Master | The majority of the planned tasks is assigned to the SM. In the worst scenario, the SM actually implements the tasks without any help. |
| Meetings result in unrealistic task lists | The Retrospective results in a lengthy list of improvements to implement. Eventually, the Scrum Team doesn't enact them. |
| Product Owner does not take part in meetings | The PO is absent from the Retrospective. This is a common occurrence. |
| Scrum Master is only a moderator | The SM doesn't take part in the discussion with the Scrum Team during the Retrospective. The SM's presence is limited to moderating the meeting and making sure it's effective. |
| **Scrum Artifacts** | |
| There is no increment by the end of the Sprint | Team members don't show a functioning product during the Sprint Review. They declare they've been working but they are unable to present complete functionalities or they can only demonstrate technicalities which are often unclear for stakeholders. |

| | |
|---|---|
| The increment does not fulfill the Definition of Done | The created functionality looks finished but team members disagree. You can hear them suggesting that there is something left to do: “It’s ready but we need to wait for tests”, “We need an hour more to let this functionality go to the production”, “It’s done but it must go to the repository”. In extreme cases, the team doesn’t respect the DoD at all: each member chooses what must be done before the element of the Product Backlog can be considered as finished. During the Sprint Review, the team is surprised by the errors that turn up. |
| Product increments are implemented too rarely | Product increments are seldom implemented or delivered to users. The implementations take place once every couple of months or even once a year. In the worst scenario, the whole work is stored and implemented in one go. |
| The product increment is built according to architecture layers | The team divides works according to architecture layers (API, graphic interface, database layer) that don’t bring much value from the user’s perspective and say little to stakeholders. |
| The current Sprint increment is not integrated with the rest of the product | The functionalities listed in the Product Backlog produced by the team work independently but are not integrated with the existing part of the product. In other words, the product increment works only locally within the team. |

| | |
|---|---|
| Sprint Backlog does not exist | The team doesn’t prepare the Sprint Backlog during Sprint Planning. Planning is based on a general choice of the Product Backlog elements to realize during a Sprint. The team’s forecasts are limited to large elements of the Product Backlog, mostly in the form of User Stories, and they don’t refer to technical tasks. There are no real concepts of how to realize the Sprint. |
| Sprint Backlog is invisible | It is unclear what work is in progress, who is responsible for each given task, and how much work remains to be done. |

| | |
|---|---|
| Sprint Backlog is a stiff, unchangeable plan | The works planned in the Sprint Backlog during Sprint Planning remain unchanged during the course of the Sprint. At Daily Scrums, the team reports the stages of the plan which have been completed, but they don't consider updating the plan to reflect the present situation. |
| Sprint Backlog's scope is a permanent obligation on the team | The team is obliged to deliver exactly the same features of the Product Backlog that are present in the Sprint Backlog. |
| Product Backlog does not exist | The Product Backlog doesn't exist and it remains unclear where to look for it. If it exists, it is fragmented and the pieces scattered across many different locations: e.g. e-mails or the PO's files. Sometimes it only exists as an oral version. |
| Product Backlog is not ready for working | The Product Backlog doesn't include any relevant information concerning the product. Its elements, especially the features planned for the upcoming Sprints, are messy, unclear, poorly defined, and non-estimated. The Backlog doesn't present a plan for further development of the product. |
| Product Backlog is disorganized | The Product Backlog's elements are randomly arranged. The functionalities planned for the nearest Sprint are not placed at the top. The product development plan is not reflected in the Product Backlog. |
| Product Backlog is of low-quality | The Product Backlog contains fragmentary information about the functionalities: they are described in general terms and it is unclear what must be done. The acceptance criteria for some elements are vague. The components don't make up a larger separate complete functionality or a specific version of the product. |
| Product Backlog does not mention the purpose of the functionality | The description of functionalities only mentions what must be done and how, it doesn't say why they should be realized. In User Stories, only the "As a…, I want to…" part appears, while the "So that…" part is left out. |

| | |
|---|---|
| Product Backlog is a copy of the documentation | The Product Backlog is filled with passages copied from template documentation prepared by someone outside the Scrum Team. The elements are written down as a wall of text. |
| Product Backlog is a list of rigid requirements | The Product Backlog is not a live artifact: the change dynamics among the elements is little (few components are added or removed). The original requirements are still there, patiently waiting for their turn. User feedback concerning the product is not recorded – and even if it is, there is no mention of that in the Product Backlog. |
| Product Backlog is managed by intuition | There are no metrics defined for the product. Decisions are taken by intuition and speculation. No specific numbers are mentioned during product discussions. |
| **Additional** | |
| There is no Refinement | The Product Backlog Refinement sessions don't actually take place. The Product Backlog is not ready to use – some elements don't exist at all or are vaguely described, disorganized, and there are no estimates. The Product Backlog doesn't reflect the product vision. Its features aren't understandable to stakeholders and the team. Team members often report a lacks of product knowledge. |
| Product Owner is not ready for meetings | The PO doesn't know the answers to the development team's questions. |
| Development Team is not ready for meetings | The team is unable to take an active part in the discussion about the Product Backlog. Developers often say "I don't know", "I'd have to check", or "It depends". |
| The long-term plans for works are not discussed | The Product Backlog is prepared ad hoc. The maximum range of the prepared elements is one Sprint ahead. It makes it impossible to define a plan for the product development. During Sprint Planning it is necessary to undertake works which will improve the Product Backlog features so that they are sufficiently ready to be included in the following Sprint. |

| | |
|---|---|
| Product Backlog elements are only discussed once | The Product Backlog's elements are only discussed once, usually at the moment of their creation. The Scrum Team refers back to them only during Sprint Planning. From the very beginning, the elements seem to be ready to use in the Sprint. |
| Product Backlog elements are only discussed at low levels | The team takes a lot of time to discuss the low-level details of the product. Discussing one element of the Product Backlog can take up a whole session of Refinement. The technical methods of developing a functionality are planned, even at the level of division into technical tasks (e.g. modification in the database, adding a class). More distant functionalities (not planned within the following two months) are also prepared in every detail. |
| Only programmers take part in the discussion | Only programmers take part in the Refinement session and only technical aspects of the product are discussed. Other members of the team (e.g. testers, analysts, graphic designers) don't speak out or, even worse, are absent. |
| The scribe is a bottleneck | The whole team is staring at an electronic device (e.g. using Jira or Trello). The meeting is not dynamic. Everyone is looking at the screen as one person jots down the conversation in the electronic device. The rest of the participants must wait until the scribe finishes writing to continue the discussion. |
| Only the Product Owner updates the Product Backlog | The PO notes down all the elements of the Product Backlog. Nobody else physically modifies, adds, or removes the elements. The team knows little about the content of the Product Backlog. |