AGILE SERVICE MANAGEMENT with Stabil-IT from gamechanginginsight.com

GAME CHANGING INSIGHT

Incident management

After receiving incidents through the service desk practice, incident management sets out to minimize the disruption caused to the service in line with the agreed service level. Incidents are classified as major or standard with major incidents requiring immediate action. The number of standard incidents permitted on a service is in line with the requirements specified by service level management. Once this number is exceeded, the standard incidents must be addressed to reduce the number back down to an acceptable level.

An incident is created when the stability of the service is less than the stability of the supporting technical platforms.

An incident can be either major (critical) or standard (non-critical).

See also: Service level management

Problem management

To secure a more reliable performance from your service you may choose to investigate in depth, one of the areas in which you have previously had an incident. This takes a concerted effort but results in a more stable and better understood service.

With problem management a node containing knowledge, can be further investigated to find the root cause and to prevent reoccurrence of the issue.

The Root cause analysis action allow you to perform this activity and results in added stability at the node plus reducing any instability in the corresponding technical platform

See also: Information management

Knowledge management

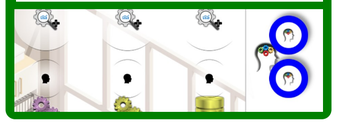
Knowledge management is an important element in the game. Over the course of the simulation you will be developing the knowledge within the company and leveraging that knowledge to be more effective in your activities. Knowledge management is also featured in the use of guilds within the simulation to highlight the need to further share knowledge with your peers.

Knowledge about a service can be published after solving an incident by taking a little extra time to document the incident and its resolution, making it a known error.

This knowledge can later be used to perform a root cause analysis as part of problem management.

Building up a good knowledge base also helps the players become more effective, allowing them to perform more activities in any given round.

When a technical platform life-cycles, some knowledge about the earlier platform version is lost.



Release management

Over the course of a game, each team is aiming to create two new releases of their service and collectively to create one new release of each of the shared services.

There are six scheduled release points during the game, where players can release their newly developed versions.

For each new version, there are three development steps and three testing steps. The three development steps must be completed for the new version to be released, however, releasing a version that has not been through all three testing steps will create instability.

See also: Service validation and testing

Service validation and testing

When the development phase of a release is completed it is still untested and if put into production may cause disruption to a number of nodes, reducing the stability of the associated technical platform. Initially, three technical platforms would potentially be affected but this can be reduced by continuing to perform a "Develop" action.

When releasing into production, the number of untested nodes will face disruption and lose stability.

See also: Release management

Relationship management

There are three main relationships at play in the simulation, the one between you as a service provider and your business users, the one between the various DevOps teams and the one with the IT management as a whole.

The former is played out with the engaged user representing the most important change requested by your user base. If you fail to satisfy this request you will lose business value.

The DevOps teams may help each other out by transferring effort in exchange for influence.

Within the IT management you have influence which you gain by working on the shared services.

There is also a real-world relationship in the game itself; between you and the other players.

Service level management

The expected performance of the service has been agreed with the customer, by assessing its impact on the business. This includes identifying what incidents are considered high priority and how quickly a resolution should be provided.

Actioning an incident is mandatory whenever there is a major (critical incident) or when the number of minor (non-critical) incidents exceeds the agreed quantity,

See also: Incident management

Continual improvement

Within the simulation there is a backlog item for improving the service. This may be done in answer to a requested change or as a way to increase the stability of a node.

Within the simulation, improvements add stability to your service but reduce the stability of the associated technical platform reinforce since, in your rush to improve you do not perform sufficient testing .

See also: Service request management

Service desk
Behind the scenes, the Service desk is receiving requests from the business covering a range of

Change control
Identified change requests are prioritized and the most significant request in each service becomes an emergency change request on a specific node.

This is carried out periodically over the course of the simulation.

Architecture management
The services are represented with 5 nodes, one for the application or UX/UI area and four for different supporting platforms. This is an overview of the architectural design of the solution.

The underlying technical platforms are life-cycled during the course of the game.

Risk management
The next technical platform to be life-cycled is known to the players and they have time to prepare in order to mitigate the disruption that will be generated. Risk of disruption is also present when deploying a release that has not been fully tested or when improving the service (an untested activity).

Software development and management
The simulation has, as a recurring element, the improvement of the software and the impact of testing . This is the background behind both the "Improve service" activity as well as the "Develop" activity (see release management).