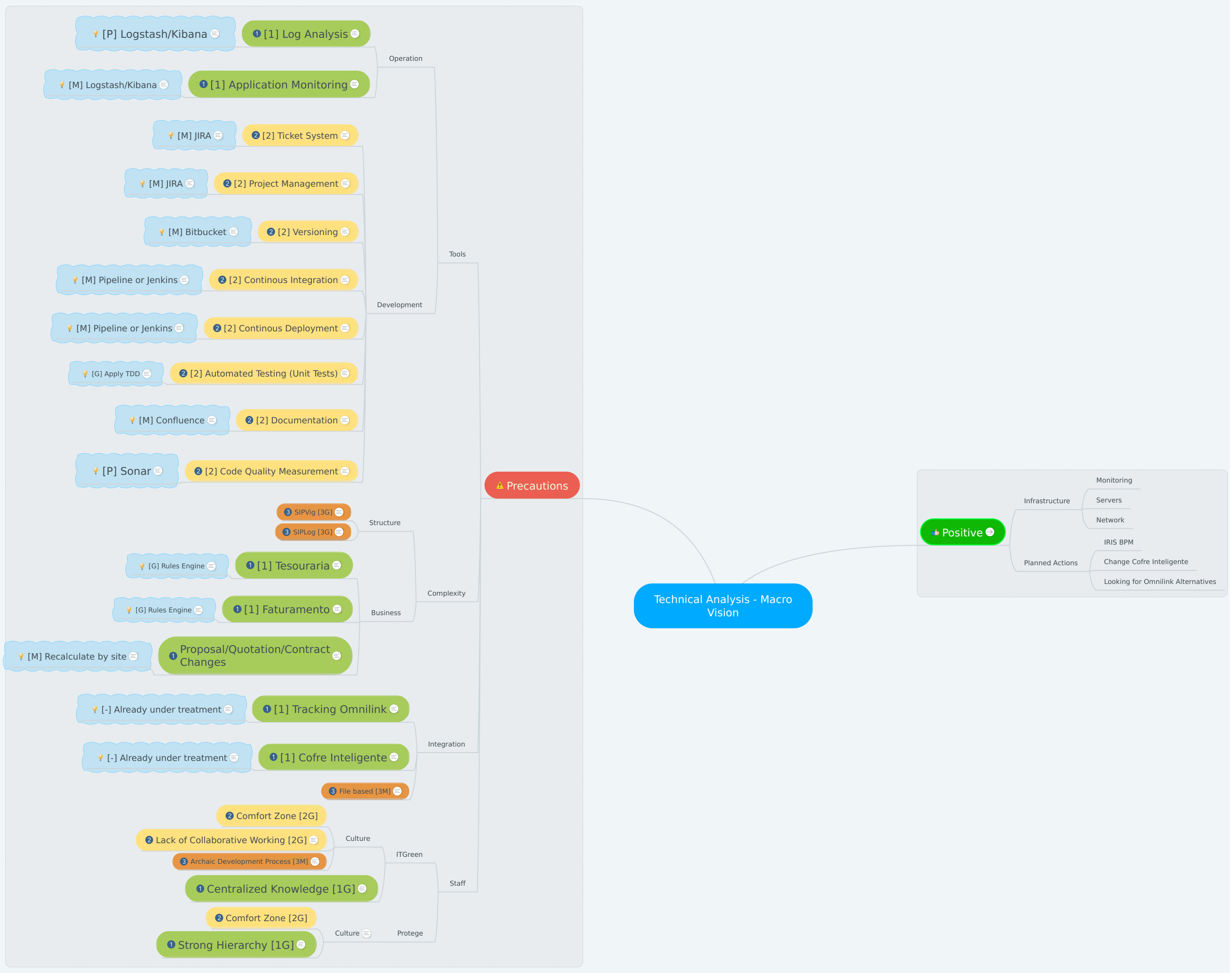
Technical Analysis - Macro Vision
Positive
Infrastructure
Monitoring
Servers
Network
Planned Actions
IRIS BPM
Change Cofre Inteligente
Looking for Omnilink Alternatives
Precautions
Tools
Operation
[1] Log Analysis
[P] Logstash/Kibana
[1] Application Monitoring
[M] Logstash/Kibana
Development
[2] Ticket System
[M] JIRA
[2] Project Management
[M] JIRA
[2] Versioning
[M] Bitbucket
[2] Continous Integration
[M] Pipeline or Jenkins
[2] Continous Deployment
[M] Pipeline or Jenkins
[2] Automated Testing (Unit Tests)
[G] Apply TDD
[2] Documentation
[M] Confluence
[2] Code Quality Measurement
[P] Sonar
Complexity
Structure
SIPVig [3G]
SIPLog [3G]
Business
[1] Tesouraria
[G] Rules Engine
[1] Faturamento
[G] Rules Engine
Proposal/Quotation/Contract Changes
[M] Recalculate by site
Integration
[1] Tracking Omnilink
[-] Already under treatment
[1] Cofre Inteligente
[-] Already under treatment
File based [3M]
Staff
ITGreen
Culture
Comfort Zone [2G]
Lack of Collaborative Working [2G]
Archaic Development Process [3M]
Centralized Knowledge [1G]
Protege
Culture
Comfort Zone [2G]
Strong Hierarchy [1G]

Technical Analysis - Macro Vision

1. Positive

Link: https://www.mindmeister.com/1190903291#

1.1. Infrastructure

1.1.1. Monitoring

1.1.2. Servers

1.1.3. Network

1.2. Planned Actions

1.2.1. IRIS BPM

1.2.2. Change Cofre Inteligente

1.2.3. Looking for Omnilink Alternatives

2. Precautions

2.1. Tools

2.1.1. Operation

2.1.1.1. [1] Log Analysis

> Today, logs are only analyzed in case of incidents. Instead of being reactive, should be analyzed frequently to preview eventual problems before the user becomes aware.

2.1.1.1.1. [P] Logstash/Kibana

> Proposal: For extensive almost real time analysis and centralized log access, we recommend to apply for tools like this

2.1.1.2. [1] Application Monitoring

It was complained that applications cannot be monitored sufficiently (Zombie Service).

2.1.1.2.1. [M] Logstash/Kibana

Proposal: For extensive almost real time analysis and centralized log access, we recommend to apply for tools like this

2.1.2. Development

2.1.2.1. [2] Ticket System

There is a gap between the both systems used to control the issues (Cherwell vs Redmine)

2.1.2.1.1. [M] JIRA

Proposal: Using a central Ticket platform is vital for better communication between Protege and its service providers. That way consistent project status can be gathered easily. JIRA, as one of the most well known development project management tool, integrates with a lot of useful tools, like Bitbucket (git like CVS), Pipeline (CI/CD tool), Confluence (Documentation) and much more.

2.1.2.2. [2] Project Management

Apparently, the projects a managed with different tools, which leads to difficulties in terms of visibility and consistency.

2.1.2.2.1. [M] JIRA

Proposal: JIRA is really good at organizing development task. More sophisticated project management features, like budget control and timeline, is not really inbuilt, but a lot of peripheral products integrates with it. JIRA could be a good starting point for project management.

2.1.2.3. [2] Versioning

Use SVN as VCS. But is not used consistently (commented code). Having special Build Masters (for merging and generating executables) indicates that the process is not optimized.

2.1.2.3.1. [M] Bitbucket

Proposal:Bitbucket is a git CVS (sldo Mercurial) that integrates natively with JIRA and other Atlassian tools. Git simplifies drastically version control as it makes branching and merging much less painless than solutions like SVN.

2.1.2.4. [2] Continous Integration

Simply does not exist!

2.1.2.4.1. [M] Pipeline or Jenkins

Proposal:Pipeline is a modern CI/CD tool that integrates natively with JIRA and other Atlassian tools. Jenkins is an well-known CI/CD tool which offers more flexibility, especially for legacy systems.

2.1.2.5. [2] Continous Deployment

Simply does not exist!

2.1.2.5.1. [M] Pipeline or Jenkins

Proposal:Pipeline is a modern CI/CD tool that integrates natively with JIRA and other Atlassian tools. Jenkins is an well-known CI/CD tool which offers more flexibility, especially for legacy systems.

2.1.2.6. [2] Automated Testing (Unit Tests)

No unit tests at all. Everything is tested manually. Testcomplete is used for some Screen Testing of desktop application (coverage at 40%), but takes several hours to run. Is considered as obligation. With this scenario Change Requests and/or refactorings are cumbersome and error prone. An extensive test suite makes changes easier as eventual side-effects can be detected during development. Furthermore, an automated unit test suite can be integrated in the deployment process, hence reduces the risk of errors significantly.

2.1.2.6.1. [G] Apply TDD

Proposal: First of all, it is essential to introduce automated tests. Test automation can be integrated in an automated integration or deployment process, reducing the risk of errors in production. Having a high test coverage (>60%) also reduces risks of side-effects in case of changes, and makes the developer feel more confident when changing or even refactoring the code. Furthermore, Test Driven Development affects the way how modules are coupled. That is, the code tends to be broken in small testable units, which raises the overall code quality, i.e. makes code more maintainable.

2.1.2.7. [2] Documentation

Missing and/or non consistent documentation complicates boarding or contextualization ("...at least 6 months to get aboard")

2.1.2.7.1. [M] Confluence

Proposal:Confluence centralizes documentation in a wiki-like style. As it is part of the Atlassian portfolio, seamless integrations with JIRA etc is guaranteed.

2.1.2.8. [2] Code Quality Measurement

There are no metrics about code quality. Actually, code grew wild...no sense of "clean code"

2.1.2.8.1. [P] Sonar

Proposal: Using a static code analysis tool helps to detect problems with maintainability of the code, or even security issues. Sonar is a sophisticated tool that can be integrated in CI/CD and provides a good overview of the code bases quality

2.2. Complexity

2.2.1. Structure

2.2.1.1. SIPVig [3G]

Outdated technology does not favor modern development style, like TDD. This module is still maturing and tends to altered/patched frequently.

2.2.1.2. SIPLog [3G]

Already matured, and suffers less corrections, but more feature addition. Outdated technology does not really favor modern development methods, like TDD.

2.2.2. Business

2.2.2.1. [1] Tesouraria

Complex Rules without any backed unit tests. Working on this requires experienced co-workers.

2.2.2.1.1. [G] Rules Engine

Proposal: Due to the nature of changing business, it could be useful to apply for a rules engine. That way cumbersome recompile and redeploy is unnecessary, and eventually the rules can be defined by the business itself, instead being redirected to a specialist. There's a lot of potential to reduce lead times and issue quantity

2.2.2.2. [1] Faturamento

Complex Rules without any backed unit tests. Working on this requires experienced co-workers.

2.2.2.2.1. [G] Rules Engine

Proposal: Due to the nature of changing business, it could be useful to apply for a rules engine. That way cumbersome recompile and redeploy is unnecessary, and eventually the rules can be defined by the business itself, instead being redirected to a specialist. There's a lot of potential to reduce lead times and issue quantity

2.2.2.3. Proposal/Quotation/Contract Changes

Currently, eventual changes of existing contracts require, for some reason, recalculation of all company sites. E.g. Bradesco changes a contract for a single site, so *all* need to be recalculation, which is an extremely time-consuming and even error prone process.

2.2.2.3.1. [M] Recalculate by site

Proposal: As suggested by an experienced ITGreen developer, it would make much more sense to calculate only for the changed site. The calculation would be a question of seconds - instead of hours - and risk of interruption is almost neglible.

2.3. Integration

2.3.1. [1] Tracking Omnilink

Integration of Omnilink is not sufficiently flexible. A lot of unnecessary data, which cannot be filtered out affecting system stability. Omnilink is not willing to customize output. According to Protege, this item is being reviewed. Eventually, it's possible to work on this without dismissing OmniLink.

2.3.1.1. [-] Already under treatment

Protege is treating this issue already.

2.3.2. [1] Cofre Inteligente

Being said, that there are problems with integration. Not much details known yet. Already under review by authorities of Protege.

2.3.2.1. [-] Already under treatment

Protege is treating this issue already.

2.3.3. File based [3M]

File based integrations require running jobs. Each job augments complexity and needs additional monitoring to see if it works.

2.4. Staff

2.4.1. ITGreen

2.4.1.1. Culture

2.4.1.1.1. Comfort Zone [2G]

2.4.1.1.2. Lack of Collaborative Working [2G]

We had the impression that specialization is a favored concept. Specialists live in a bubble, which makes them on one hand valuable, but on the other hand they tend to keep their knowledge as a secret, unwilling to share with others. This kind of behavior promotes even competitive thinking and may lead to counter-productive attitudes.

2.4.1.1.3. Archaic Development Process [3M]

Archaic culture reflected through classic development process - kind of waterfall.

2.4.1.2. Centralized Knowledge [1G]

Almost all business knowledge is concentrated at ITGreen. Furthermore, ITGreen promotes specialization, no real allrounders. High risk of essential knowledge loss.

2.4.2. Protege

2.4.2.1. Culture

Apparently, a very rigid almost military culture.

2.4.2.1.1. Comfort Zone [2G]

2.4.2.1.2. Strong Hierarchy [1G]

A rigid almost military hierarchy affects motivation. People fear to think outside the box and innovation is blocked. No one wants to experiment, as eventual failures are punished.