

Automated Impact Analysis

Calculating Cost of Downtime by Simulation of Failure

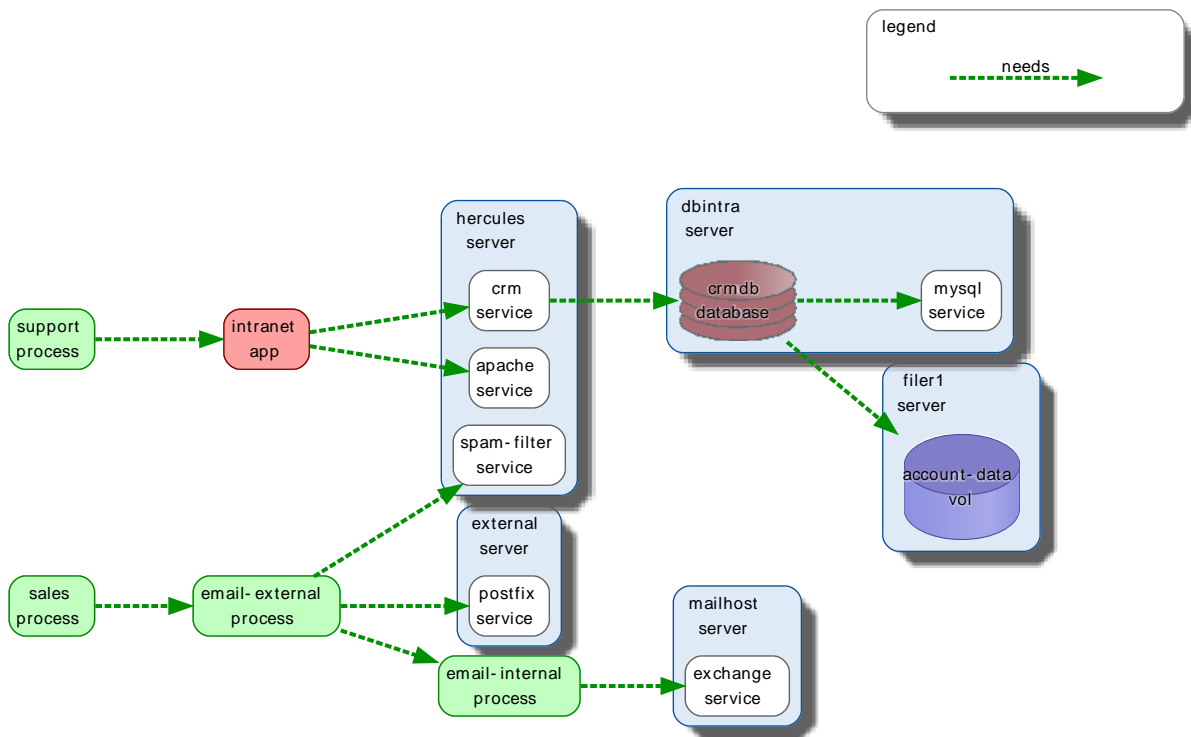
Cost of Downtime Method 1
 Modeling and Simulating of Failure 2
 Discussion of Simulation Results 4
 Determining Restoration Order 5

Cost of Downtime Method

This document describes an approach to using Blueprints™ for automated impact analysis through a method of calculating *cost of downtime* for every component that supports important business processes. Traditionally, an IT component's cost is recorded as its book value or replacement cost. More useful to disaster recovery and business continuity however, is valuing components by the potential cost of lost productivity and lost reputation due to failure. Cost of downtime metrics help answer difficult questions such as: where to focus resources to improve reliability, and what's a good order to restore services after failure.

In the example and discussion that follows, we demonstrate how to model dependencies, and use that model as the basis for automated failure simulations which calculate cost-of-downtime metrics to the component level. To conduct the impact analysis, we first build a model of the business process to systems dependencies using Blueprints™, then associate a *util* with each high level business process to quantify the utility the company realizes from the process.

Figure 1. Dependency Model



Rather than attempt to assign utility metrics to every IT component, we instead assign them only to a limited set of high level processes, and let Blueprints™ analyze the model's unique dependency configuration to compute cost-of-downtimes for all underlying services and hardware devices.

Although an organization may have thousands of IT components, they need only identify the much smaller set of high level business processes, and focus on assigning reasonable utility metrics for those. In the real world, IT configuration is constantly in flux as servers and applications are changed but fortunately the utility of high level business processes is relatively static. As the model is updated with IT changes such as moving a software component or virtualization of physical servers, it continues to provide current and reasonably accurate component level cost-of-downtime estimates simply by conducting a new series of simulations with the adjusted model.

Blueprints™ provides two key capabilities that make the creation and maintenance of this analytic capability possible: easy systems modeling, and automated calculation through simulation. Although the work involved in modeling a company's IT infrastructure is non-trivial, the benefits are worth the effort and go far beyond the ability to calculate cost of downtime metrics. This paper's focus, however, is limited to discussing only downtime costing.

Note: All diagrams and tables in this article were generated automatically by Blueprints™ from an actual business process and IT dependency model.

Modeling and Simulating of Failure

In the following example, we assign a *util* for certain high level processes which reflect a dollar value estimate of utility that our company derives from a process per hour. The planning specialist is expected to facilitate the collection of this information from members of the company's management teams who are knowledgeable about various business processes, and their relative value to the business.

Blueprints™ uses these *util* values, and an understanding of the systems' dependencies to calculate the value of lost utility from any given individual IT component failure. This is accomplished by conducting a simulation of failure for each service, and identifying which direct services would be impacted. The simulation then propagates that failure up the dependency tree to identify additional services also expected to fail. When the simulation has finished, the utility of all failing objects is summed, and that dollar value is assigned to the initial failing component which began the simulation. That sum is recorded as the component's cost-of-downtime (*_cod*). In other words, the cost-of-downtime is the lost utility the company experiences when that component fails.

This approach enables Blueprints™ to score every component with it's own specific downtime cost to the company. With this information, managers and IT specialists can better understand which IT components are more critical than others, and manage them accordingly.

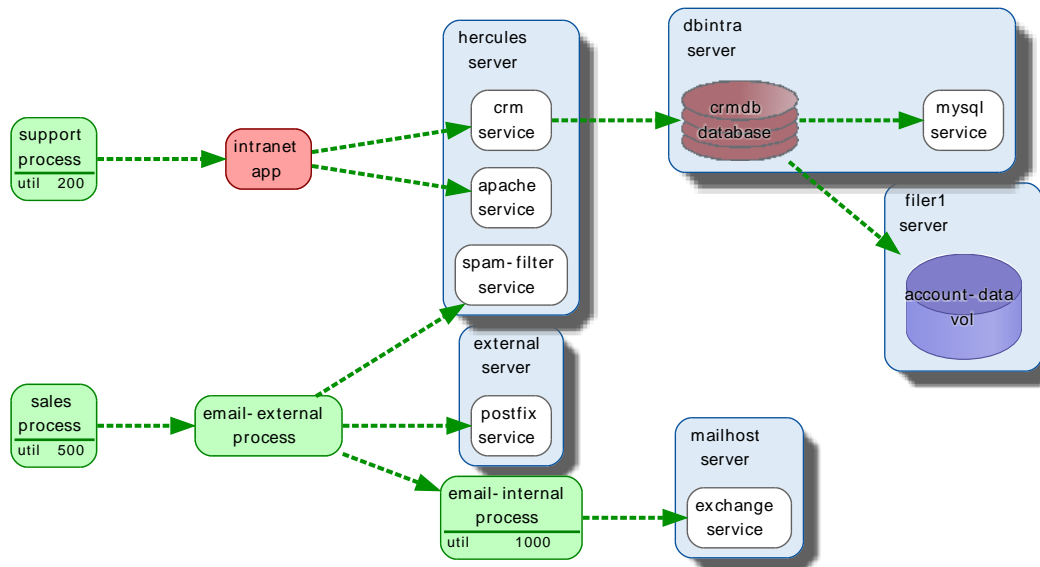
In Figure 1 on page 1, we have modeled the dependencies of four business processes *sales*, *support*, *email-external*, and *email-internal*. There are a total of five servers (the blue boxes) with various services and databases running on these servers supporting the higher level services. The dashed green lines denote a *needs* relation between two objects. For example, *intranet* needs *apache*.

The direction of the relation is important; in this example, we see that a failure of *apache* will cause *intranet* to fail as well. Some readers may be more familiar with the relations modeled in the opposite direction, where *apache* is shown as an *input* to *intranet*, however we've chosen to model from the perspective dependencies rather than inputs and outputs since the dependency is exactly what matters for this analysis regardless of data flow.

Higher level services appear to the left; supporting components appear to the right. Notice how sending mail externally needs all the functionality of internal email capability, plus some additional services.

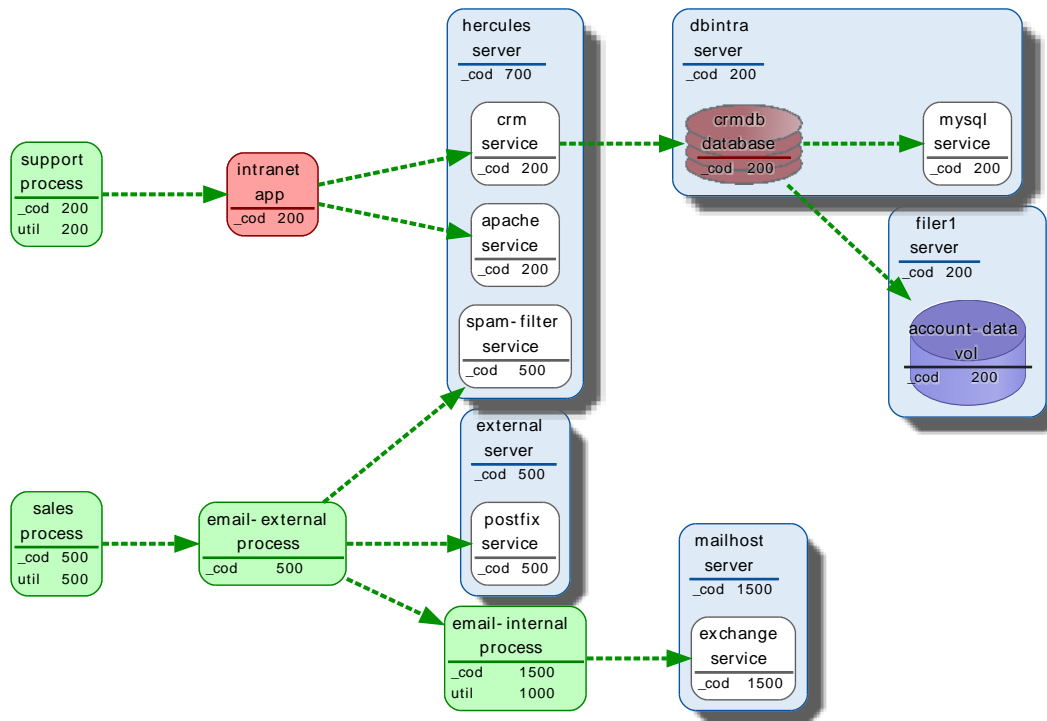
Let's take another look at the model, this time annotating certain high level processes with the *util* values we've asked managers of the company to declare which represent the relative worth of these processes to the company.

Figure 2. Dependency Model



With this additional information in the model, Blueprints™ can now run failure simulations for each object to determine the total lost utility to the company experiences when that object fails. Since our model contains 18 objects, 18 simulations will be conducted, each one supposing that one particular object is failing. Once the simulations are conducted, the model is annotated with the cost-of-downtime attributes (*_cod*) calculated from each of the simulations.

Figure 3. Dependency Model



Discussion of Simulation Results

Notice how each object now has a *_cod* attribute. We can think of this as the dollar value cost of downtime for that object. If the *util* attributes represent dollar value of utility per hour, then the *_cod* represents dollars lost per hour due to failure of that component.

See how *hercules*'s *_cod* is 700 dollars per hour because when *hercules* fails, it adversely impacts two business processes with a declared utility. Both *sales* and *support* would be impacted, and the sum of their lost utility is 700.

For another slightly more complex example, let's look closely at the *mysql* service and verify the cost-of-downtime (*_cod*) was computed as we expect it should be. A failure of *mysql* should cause the *crmdb* database to fail, causing the *crm* service to fail, causing a failure of the *intranet* application, and ultimately the *support* process thus causing 200 units (dollars) of lost utility. In other words, the cost of downtime is 200 dollars per hour when *mysql* fails. For much the same reason, when the server *dbintra* fails, it too will impact *support* and cause the same lost utility of 200 dollars.

But notice how the *_cod* for *dbintra* is not simply a sum of the *_cod*'s of it's hosted services! If this were the case, *dbintra* would have a *_cod* of 400 dollars. That however, over-represents the impact, since although both *crmdb* and *mysql* would indeed fail when *dbintra* fails, the source of lost utility (*support*) is the same for each component, so the impact of *dbintra*'s failure should only include *support*'s utility once. This is why our cost-of-downtime calculation is done by simulating failures, and not with a less sophisticated, recursive sum of impacted services--it enables us to add better logic to the calculations.

Armed with this new information (cost of downtime metrics), we can see IT resources in a new light--as the potential threat to business function each constitutes. Although this paper demonstrated

this approach with a small set of IT resources, the method is conducted the same way for much larger IT configurations. An IT infrastructure of only tens of servers may not merit this approach to analyzing impact; intuition might be sufficient. But consider an environment with hundreds or thousands of components. Blueprints™ enables you to gather and document the dependency knowledge initially, keep up with changes, and employ that model to make better recovery and continuity planning decisions.

Determining Restoration Order

Finally we'll conclude the discussion with some ideas on how these cost-of-downtime metrics benefit the disaster recovery effort directly when used in a strategy to prioritize system restoration. What if all servers in this example were lost due to a fire in the datacenter? What would be best order for system administrators to attack the restoration project? Knowing which servers have a greater cost-of-downtime helps determine the order they should be restored.

Suppose all servers were lost...

Table 1. Simulated Failure of Devices

Objects
external, hercules, mailhost, dbintra, filer1

Simulate the failure and determine the lost utility...

Table 2. Impacted Processes, Servers, and Apps

Object	Type	Util
email-internal	process	1000
support	process	200
sales	process	500

Restore the most valuable servers first...

Table 3. Optimized Restoration Order

Obspec	Type	_failing	_impacted	_cod	_restore_order
mailhost	server	yes	5	1500	1
hercules	server	yes	8	700	2
external	server	yes	4	500	3
filer1	server	yes	6	200	4
dbintra	server	yes	6	200	5

Here Blueprints™ has determined the order the servers should be restored to ensure services boot correctly and cost to the company is minimized.

For more information regarding Blueprints™, please see <http://pathwaysystems.com>.