A Real-World SLA Use Case in ContractLog

Adrian Paschke

SUBMITTED TO W3C RULE INTERCHANGE WORKING GROUP
DECEMBER 2005

http://www.w3.org/2005/rules/wg/wiki/Rule_Based_Service_Level_Management_and_SLAs_for_Service_Oriented_Computing

1 DESCRIPTION

The SLA defines three monitoring schedules, “Prime”, “Standard” and “Maintenance”:

<table>
<thead>
<tr>
<th>Schedule</th>
<th>Time</th>
<th>Availability</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime</td>
<td>8 a.m. - 18 p.m.</td>
<td>98% (99%)</td>
<td>100% ; pinged every 10s</td>
</tr>
<tr>
<td>Standard</td>
<td>18 p.m. - 8 a.m.</td>
<td>95% (97%)</td>
<td>99%; pinged every min.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>0 a.m. - 4 a.m.*</td>
<td>20% (50%)</td>
<td>80%; pinged every 10 min</td>
</tr>
</tbody>
</table>

During prime time the average availability has a low value of 98%, a median of 99% and a high value of 100% and a response time which must be below 4s. The service metrics are calculated via a ping every 10 seconds. During standard time the average availability is {high: 99%; low: 95%; median: 97%} and response time {high: 10 sec.; low: 16 sec.; median: 14 sec.} monitored via a ping every minute. Maintenance is permitted to take place between midnight and 4a.m. During this period the average availability is {high: 80%; low: 20%; median: 50%} monitored every 10 minutes. Response time will not be monitored in this case. Further the SLA defines a “bonus-malus” policy:

<table>
<thead>
<tr>
<th>Price</th>
<th>Base</th>
<th>Bonus</th>
<th>Malus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime</td>
<td>( p_{\text{prime}} )</td>
<td>( p_{\text{prime}} + (x_{\text{high}} - x_{\text{median}}) \times b_{\text{bonus}} % )</td>
<td>( p_{\text{prime}} - (x_{\text{median}} - x_{\text{low}}) \times b_{\text{malus}} % )</td>
</tr>
<tr>
<td>Standard</td>
<td>( p_{\text{standard}} )</td>
<td>( p_{\text{standard}} + (x_{\text{high}} - x_{\text{median}}) \times b_{\text{bonus}} % )</td>
<td>( p_{\text{standard}} - (x_{\text{median}} - x_{\text{low}}) \times b_{\text{malus}} % )</td>
</tr>
<tr>
<td>Maintenance</td>
<td>( p_{\text{maintenance}} )</td>
<td>( p_{\text{maintenance}} + (x_{\text{high}} - x_{\text{median}}) \times b_{\text{bonus}} % )</td>
<td>( p_{\text{maintenance}} - (x_{\text{median}} - x_{\text{low}}) \times b_{\text{malus}} % )</td>
</tr>
</tbody>
</table>

According to the monitoring schedules a differentiated base price is defined if the service levels are met. If the service levels are exceeded (median to high) a dependent bonus is added and if they fall below the agreed upon service levels (median to low) a discount is deducted. The bonus and malus are defined as a percentage value of the base price. If a service level is missed, i.e. the actual value falls below the low service level (<low) an additional penalty has to be paid which increases exponentially with the number of incidents during the accounting period. In case of outages/incidents the SLA defines two escalation levels:

<table>
<thead>
<tr>
<th>Level</th>
<th>Role</th>
<th>Time-to-Repair (TTR)</th>
<th>Rights / Obligations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process Manager</td>
<td>10 Min.</td>
<td>Start / Stop Service</td>
</tr>
<tr>
<td>2</td>
<td>Quality Manager</td>
<td>Max. Time-to-Repair (MTTR)</td>
<td>Change Service Levels</td>
</tr>
</tbody>
</table>

Each escalation level defines clear responsibilities in terms of associated roles which have certain rights and are obliged to do certain remedial actions in case of incidents which initiate the respective escalation level. In the SLAs’ escalation level 1 the process manager is obliged to restart an unavailable service within 10 minutes. Accordingly, she has the right (permission) to start and stop the service. If she fails to do so, escalation level 2 is triggered and the
quality manager is informed. The quality manager has more rights, e.g. the right (permission) to adapt/change the SLA management systems respectively the service levels. The quality manager might discuss the time needed to repair with the process manager and extend it up to a maximum time to repair level (change request). In case of very critical incidents the system might directly proceed to escalation level 2 and skip level 1. The basic SLA described here might be further extended with various other rules adding further SLA decision logic. For example the SLA might include compliance rules for reporting and auditing according to regulations such as the Sarbanes-Oxley act or more service levels such as maximum downtime, failure frequency or process level performance values according to ITIL such as time between loss and replacement (incident mgt.), number of repeated disturbances (problem mgt.), or number of untreated changes (change mgt.). An overview of further SLA and ITIL metrics can be found in [Paschke,A, Schnappinger-Gerull, E.: A Categorization Scheme for SLA Metrics, Multi-Conf. Information Systems, 2006]

2 Formalization in ContractLog

This section shows the formalization of a selected subset in ContractLog, namely the monitoring schedules, the escalation levels and the associated roles, as well as the following SLA rules:

"The service availability will be measured every t_schedule according to the actual schedule by a ping on the service. If the service is unavailable and it is not maintenance then escalation level 1 is triggered and the process manager is informed. Between 0-4 a.m. the process manager is permitted to start servicing which terminates any escalation level. The process manager is obliged to restart the service within time-to-repair, if the service is unavailable. If the process manager fails to restore the service in time-to-repair (violation of obligation), escalation level 2 is triggered and the chief quality manager is informed. The chief quality manager is permitted to extend the time-to-repair interval up to a defined maximum value in order to enable the process manager to restart the service within this new time-to-repair. If the process manager fails to restart the service within a maximum time-to-repair escalation level 3 is triggered and the control committee is informed. In escalation level 3 the service consumer is permitted to cancel the contract."

The formalization in ContractLog is as follows:

% service definition
service(http://ibis.in.tum.de/staff/paschke/rbsla/index.htm).
% role model and escalation levels
initially(escl_lvl(0)). % initially escalation level 0
% if escalation level 1 then process manager
role(process_manager) :- holdsAt(escl_lvl(1),T).
% if escalation level 2 then chief quality manager
role(chief_quality_manager) :- holdsAt(escl_lvl(2),T).
% if escalation level 3 then control committee
role(control_committee) :- holdsAt(escl_lvl(3),T).
% time schedules standard, prime, maintenance and monitoring intervals
% before 8 and after 18 every minute
schedule(standard, Service):-
systime(datetime(Y,M,D,H,Min,S)),
less(datetime(Y,M,D,H,Min,S),datetime(Y,M,D,8,0,0)),
interval(timespan(0,0,1,0),datetime(Y,M,D,H,Min,S)),
service(Service),
not(maintenance(Service)). % not maintenance
schedule(standard, Service):-
systime(datetime(Y,M,D,H,Min,S)),
more(datetime(Y,M,D,H,Min,S),datetime(Y,M,D,18,0,0)),
interval(timespan(0,0,1,0),datetime(Y,M,D,H,Min,S)),
service(Service),
not(maintenance(Service)). % not maintenance
% between 8 and 18 every 10 seconds
schedule(prime, Service):-
systime(datetime(Y,M,D,H,Min,S)),
lessequ(datetime(Y,M,D,H,Min,S),datetime(Y,M,D,18,0,0)),
moreequ(datetime(Y,M,D,H,Min,S),datetime(Y,M,D,8,0,0)),

% between 0 and 4 if maintenance every 10 minutes
schedule(maintenance, Service) :-
    lessequ(sysTime(datetime(Y, M, D, H, Min, S)),
            datetime(Y, M, D, 4, 0, 0)),
    schedule(maintenance, Service)

% initiate/terminate maintenance if permitted
% initiate(startServicing(Service), T),
% terminates(stopServicing(Service), maintenance(Service), T). % stop maintenance
happens(startServicing(Service), T):-
    happens(requestServicing(Role, Service), T),
    holdsAt(permit(Role, Service, startServicing(Service)), T).
% initiate unavailable state if outage event happens
initiates(outage(Service), unavailable(Service), T).
terminates(restart(Service), unavailable(Service), T).
% define time-to-repair deadline and trigger escalation level 2 if deadline is elapsed
time_to_repair(tdeadline). % relative time to repair value
trajectory(escl_lvl(1), T1, deadline, T2, (T2 - T1)). % deadline function
derivedEvent(elapsed).
happens(elapsed, T) :-
    time_to_repair(TTR), valueAt(deadline, T, TTR).
    terminates(elapsed, escl_lvl(1), T). % terminate escalation level 1
    initiates(elapsed, escl_lvl(2), T). % initiate escalation level 2
% trigger escalation level 3 if (updated) time-to-repair is > max time-to-repair
happens(exceeded, T) :-
    happens(elapsed, T1), T = T1 + ttrmax.
    terminates(exceeded, escl_lvl(2), T). % oblige process manager
derivedEvent(exceeded) % derive obligation to start the service if service unavailable
derived(oblige(process_manager, Service, restart(Service))).
happens(oblige, T) :-
    holdsAt(unavailable(Service), T).
    derived(oblige(process_manager, Service, restart(Service))).
% service consumer is permitted to cancel the contract in escl_lvl3
derivedEvent(cancel) % derived(permit(service_consumer, contract, cancel)).
happens(permit, T) :-
    holdsAt(permit(service_consumer, contract, cancel), T).

% ECA rule: “If the ping on the service fails and not maintenance then trigger escalation level 1 and notify process manager, else if ping succeeds and service is down then update with restart information and inform responsible role about restart”.
eca(schedule(T, S), not(available(S)), not(maintenance(S))), escalate(S), _, restart(S)). % ECA rule
available(S) :- WebService.ping(S). % ping service
maintenance(S) :- sysTime(T), holdsAt(maintenance(S), T).
escalate(S) :-
    sysTime(T),
    not(holdsAt(unavailable(S), T)), % escalate only once
    update('outages', "happens(outage(_0, _1), [S, T]), % add event
            role(R), notify(R, unavailable(S)). % notify
restart(S) :-
    sysTime(T), holdsAt(unavailable(S), T),
    update('outages', "happens(restart(_0, _1), [S, T]), % add event
            role(R), notify(R, restart(S)), % update + notify
% initiate unavailable state if outage event happens
initiates(outage(S), unavailable(S), T).
terminates(restart(S), unavailable(S), T).
% initiate escalation level 1 if outage event happens
terminates(outage(S), escl_lvl(0), T), initiates(outage(S), escl_lvl(1), T).
% terminate escalation level 1/2/3 if restart event happens
terminates(restart(S), escl_lvl(0), T), initiates(restart(S), escl_lvl(1), T).
terminates(restart(S), escl_lvl(2), T).
terminates(restart(S), escl_lvl(3), T).
% initiate escalation level 1 if outage event happens
terminates(outage(S), escl_lvl(0), T), initiates(outage(S), escl_lvl(1), T).
% initiate/terminate maintenance if permitted
% initiate(startServicing(Service), maintenance(Service), T), T). % stop maintenance
happens(startServicing(Service), T):-
    happens(requestServicing(Role, Service), T),
    holdsAt(permit(Role, Service, startServicing(Service)), T).
% permit process manager to start servicing between 0-4 a.m.
holdsAt(permit(process_manager, Service, startServicing(Service)), datetime(Y, M, D, H, Min, S)):=-
    lessequ(datetime(Y, M, D, H, Min, S), datetime(Y, M, D, 4, 0, 0)).
% else forbid process manager to start servicing.
holdsAt(forbid(process_manager, Service, startServicing(Service)), datetime(Y, M, D, H, Min, S)):=-
    more(datetime(Y, M, D, H, Min, S), datetime(Y, M, D, 4, 0, 0)).
% else forbid process manager to start servicing.
holdsAt(forbid(process_manager, Service, startServicing(Service)), datetime(Y, M, D, H, Min, S)):=-
    more(datetime(Y, M, D, H, Min, S), datetime(Y, M, D, 4, 0, 0)).
% derive obligation to start the service if service unavailable
derived(oblige(process_manager, Service, restart(Service))).
holdsAt(oblige(process_manager, Service, restart(Service)), T) :-
    holdsAt(unavailable(Service), T). % oblige process manager
derivedEvent(elapsed) % deadline function