**Specifications for Elevator solution v.1**

To run the solution, you need MS SQL Server 2005 or greater. The content and test scripts are in scripts.sql and test.sql files.

1. Task:

Imagine that you have a friend called Ivan. He invented the hardware for modern elevators for his new startup. The lifts support the most popular programming languages and also have all kind of sensors (e.g. weight, speed, temperature, literary whatever you can think of). Ivan asks you to write the first version of the lift movement software

Test scenarios:

- When there is new elevator request, the most optimal elevator accepts it (You should state what optimal is for your implementation)

- When the elevator is full, it shouldn’t stop to take new passengers

- When the elevator is on a floor that is the stop of some passenger, the elevator stops

- When the elevator is overloaded it doesn’t function until it is freed

1. Used programing language: T-SQL
2. Description of the solution:

* Elevators: 3 pieces {ID 1 – 3}
* Floors – 50 {ID 0 – 49}
* Max. weight – as per specification in dbo.Elevators.Max\_Weight
* External Buttons – 2 per each floor for each direction, properties send when pressed

{ID of the floor int, requested\_directions string}

* Statuses of the elevators – direction up, direction down, stop (loading/unloading people), idle (elevator is not executing any request)

1. Workflow – as per Workflow.xml (please use <https://www.draw.io/> if any tool is not available).
2. Content – tables in dbo scheme:

* Dbo.Elevator
* ID (int) – the identity of the elevator
* Model varchar(30) (Standard and Extra Load)
* Max\_Weight smallint – the load capacity of the elevator
* Current\_state varchar(30) – keeps the current state of the elevator
* Dbo.Active\_Queue
* ID int (identity of the request)
* Elevator\_ID int – the elevator that will serve the request, a reference to the elevator table (no foreign key will be attached, due to the simplicity of the implementation)
* Request\_floor tinyint – keeps the floor that the request was sent from
* Requested\_floor tinyint – keeps the floor requested to go or the floor that the last query of the elevator will be handled to on external queries
* Estimated\_Direction varchar(10) – up or down, defined by request\_floor – requested\_floor) = if negative “up”, if positive – down
* Transported\_Load (int) – 0 if there are no queries to handle and the elevator is Idle
* Dbo.Archive
* ID (int) of the query handled
* Traveled floors
* Delivered\_Weight (int) 0 if the starting position is idle

1. Optimization criterias for choosing elevator to handle a query

* Get the nearest elevator going the same direction
* Get an Idle elevator if the above failes
* Get an elevator that has the fastest ETA to the requested floor

1. Stored procedures:

* SP\_log\_request

@p\_floor\_id int,

@p\_direction varchar(10) = null (for internal requests)

@p\_is\_external bit = 1,

@p\_requested\_floor int = null, (for external queries)

@p\_request\_id int = 0, for forwarded request refused to stop due to overload

* SP\_Stop

@p\_request\_id

* SP\_GO

@p\_request\_id,

@p\_current\_weight,

@p\_input\_weight int,

@p\_output\_weight