

## WIT MSc IT Architecture Patterns Architecture Description (AD)

### WIT ITA Patterns Lab AD: 'RunKeeper Achiever' Analytics Dashboard

<< *Note: This Architecture Description (AD) template from the Open Group TOGAF™ 9.1 architecture framework has been adapted to the purposes of the WIT MSc module on IT Architecture Patterns.*>>.

## Table of Contents

<b>A)Problem Description.....</b>	<b>6</b>
A.1PROBLEM SCOPE.....	6
A.1.1PROBLEM SUMMARY.....	6
A.1.2DOMAIN GLOSSARY.....	9
A.2STAKEHOLDERS CONCERNS.....	9
A.3REQUIRED QUALITY PROPERTIES.....	9
<b>B)Solution Description.....</b>	<b>11</b>
B.1ARCHITECTURE VIEWS.....	11
B.2CONTEXT VIEW.....	12
B.2.1VIEW INTENT.....	12
B.2.2VIEW MODELLING ARTIFACTS.....	13
B.2.2.1SYSTEM CONTEXT MODEL.....	13
B.2.2.2SYSTEM FEATURES MODEL.....	19
B.3BEHAVIORAL VIEW.....	20
B.3.1VIEW INTENT.....	20
B.3.2VIEW MODELLING ARTIFACTS.....	20
B.3.2.1DATA PREPARATION MODEL.....	20
B.3.2.2DASHBOARD RENDERING MODEL.....	23
B.4INFORMATION VIEW.....	30
B.4.1VIEW INTENT.....	30
B.4.2VIEW MODELLING ARTIFACTS.....	34
B.4.2.1BATCH ANALYTICS DATA FLOW MODEL.....	34
B.4.2.2DASHBOARD RENDERING DATA FLOW MODEL.....	36
B.5FUNCTIONAL VIEW.....	38
B.5.1VIEW INTENT.....	38
B.5.2VIEW MODELLING ARTIFACTS.....	38
B.5.2.1INDIRECTION LAYERS MODEL.....	38
B.5.2.2DATA PROVISIONING MODEL.....	40
B.6ARCHITECTURE PERSPECTIVES.....	42
B.6.1PERFORMANCE PERSPECTIVE.....	42
B.6.1.1RECORDING of DESIGN DECISIONS / TRADE-OFFS.....	42
B.6.2SCALABILITY PERSPECTIVE.....	42
B.6.2.1RECORDING of DESIGN DECISIONS / TRADE-OFFS.....	42
B.6.3TESTABILITY PERSPECTIVE.....	43

B.6.3.1RECORDING of DESIGN DECISIONS / TRADE-OFFS.....	43
B.6.4FLEXIBILITY PERSPECTIVE.....	43
B.6.4.1RECORDING of DESIGN DECISIONS / TRADE-OFFS.....	43
B.7VIEW MAPPINGS.....	43
B.7.1For each VIEW MAPPING.....	43
<b>C)References.....</b>	<b>44</b>

## Document Information

<b>Project Name:</b>	RunKeeper “Achiever” Analytics Dashboard		
<b>Prepared By:</b>	WIT ITA Patterms Module Authors	<b>Document Version No:</b>	1.0
<b>Title:</b>	Architecture Description (AD)	<b>Document Version Date:</b>	2016-04-15
<b>Reviewed By:</b>	WIT ITA Patterms Module Authors	<b>Review Date:</b>	2016-04-13

## Distribution List

From	Date	Phone/Fax/Email
WIT ITA Patterms Module Authors	2016-04-15	N/A

To	Action*	Due Date	Phone/Fax/Email
WIT ITA Patterms Module Students	Review	2016-04-28	N/A

\* Action Types: Approve, Review, Inform, File, Action Required, Attend Meeting, Other (please specify)

## Document Version History

Version Number	Version Date	Revised By	Description	Filename
1.0	Apr. 13 <sup>th</sup>	WIT ITA Patterms Module Authors	This document files some of the Architecture Views describing the Lab of the IT Architecture Styles & Patterns module: Runkeeper's “Achiever” Analytics Dashboard.	{runkeeper.archimate}

---

# Document Overview

---

This Architecture Description (AD) template from the Open Group TOGAF™ 9.1 architecture framework has been adapted to the purposes of the WIT Msc module on IT Architecture Patterns.

The following document shows how to consolidate descriptive Architecture Views of the WIT ITA Patterns Lab on Architecture Styles & Patterns into a TOGAF Architecture Description.

Note: This instance isn't an instantiation of the Template, not the template it self. The complete template can be found on Moodle, as part of the Module Materials.


Note: This document isn't a complete AD, but shows to integrate a few Views, Perspectives and Patterns for the purposes of students' working assignments.

## A) Problem Description

### A.1 PROBLEM SCOPE

#### A.1.1 PROBLEM SUMMARY

This section to briefly describe here the problem you are solving for.

<b>Project Name</b>	Runkeeper “Achiever” Analytics Dashboard
<b>Industry Domain</b>	Sports / Fitness Tracking
<b>Problem Description</b>	<p>Runkeeper wants to provide superior insight to its Customers about their progression towards achieving challenges and training plans.</p> <p>The granularity of current data points captured by Runkeeper customers isn’t sufficient to create meaningful data insights.</p> <p>To create a data analytics dashboard, Runkeeper must collect more precise, more granular data about the physical activities and health indicators of its customer, 24/7.</p> <p>A partnership with ASICS will offer to a selected amount of premium Runkeeper customer to participate to a 1-year suscription plan named: Runkeeper ELITE.</p> <p>This new subscription plan will be proposed to 5,000 Customers, formely suscribers of the Runkeeper GO Plan for a minimum of 1-year.</p> <p>5,000 ASICS Alive M10 weareable (fitness-tracking) devices will be dispatched to existing suscribers, converted to ELITE members.</p> <div><p>A fitness as stylish</p><p>BREAKTHRU</p><p>PurePulse™ Heart Rate</p><p>On-Screen Workouts</p><p>Alive M10 Fitness Watch</p></div> <p>The wearable device captures enough detailed data points to generate meaningful data analytics, in turn enabling the creation of an Analytics Dashboard named: Achiever</p>

#### Analytics Dashboard.

It is expected that the amount of data generated by the sample of 5,000 ELITE Members over 1-year to be in the order of TB of data.

It is expected that processing all the data collected will require significant computing cycle.

It is expected that all data points collected will require new forms of data storage (i.e. different from classic RDBMS data storage technologies in use at Runkeeper.com to date).

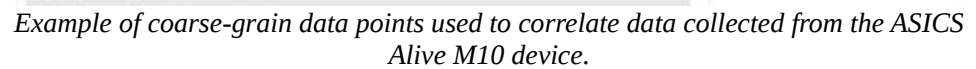
To make sense of all data points will be correlated, 2 other sources of information will used:

1. Customer Heath high-level data profile (obtained from off-premise data sources owned by Healthgraph.com)

Data	Week of May 17 <sup>th</sup>	Week of May 24 <sup>th</sup>	Week of June 1 <sup>st</sup>
Weight	180 lbs	182 lbs	176 lbs
Avg. Daily Calories	3,400	3,500	2,400
Avg. Daily Sleep	6 hrs.	5 hrs.	8 hrs.

*Example of coarse-grain data points used to correlate data collected from the ASICS Alive M10 device.*

2. Customer Activity high-level data profile (obtained from on-premise data sources owned by Runkeeper.com).





## Fort William Marathon Training

Click a date to see the run details

Choose a Pace: Cumulative



### Training Totals

Duration	Pace	Distance
<b>00:28:48</b>	<b>8:39</b>	<b>370.5</b>

### Jun 20

Duration	Pace	Distance	Elevation
<b>01:15:05</b>	<b>8:43</b>	<b>8.6 mi</b>	<b>2.0 ft</b>

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
April 2015	6	7	8	9	10	11	12
	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28	29	30			
May 2015					1	2	3
	4	5	6	7	8	9	10
	11	12	13	14	15	16	17
	18	19	20	21	22	23	24
	25	26	27	28	29	30	31
June 2015	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30					



7:00  11:00



60 ft  
40 ft  
20 ft  
0 ft

*Example of resulting Analytics Dashboard screen*



	<p>The first revision of this Analytics Dashboard will be provided for Web Desktop users ONLY. However, a Mobile-friendly version will follow, and so the proposed architecture must account for this.</p> <p>Comparative progress can be analysed using activity tracking indicators (i.e. elevation, distance, steps, etc.), and presented on a visual map with color codifications/legends.</p> <p>Comparative progress can also be slided &amp; diced using fitness health indicators (i.e. heart beat, calories burned, etc.), and presented on visual statistical charts.</p>
--	---

### A.1.2 DOMAIN GLOSSARY

This section to describe here key Terms & Definitions essential to the understanding the problem domain.

<b>ELITE Member</b>	A registered Runkeeper.com Customer paying a premium. ELITE Members are part of the ELITE Subscription Plan. This subscription plan provides its members with an ASICS Alive M10 wearable device. ELITE Members can track their activity (e.g. route, heart rate) using this device, and upload their activity feeds to Runkeeper.com via a Mobile device or PC workstation.
<b>ELITE Plan</b>	A 1-year MINIMUM premium subscription plan for Runkeeper users that have been Customers of the GO Plan for more than a year. The subscription plan includes free shipping of a n ASICS wearable device.
<b>ASICS Alive M10</b>	A (fictive) wearable device, shipped for a selected set of Runkeeper Customers measuring all-day steps, distance, calories burned, active minutes, hourly activity & stationary time. The ASICS Alive M10 is a prototype device shipped to Runkeeper ELITE Members ONLY.

## A.2 STAKEHOLDERS CONCERNS

This section is to identify the stakeholders of the target solution architecture – i.e. Stakeholders who need to review architecture Views and this AD document.

<b>Initiative Sponsors</b>	Runkeeper Chief Information Officer, Chief Marketing Officer
<b>Project Development Team</b>	Project IT Team implementating the IT Architecture described in this document. The project Team is composed of 2 data analysts/testers, and two developers.
<b>System Administrator</b>	Responsible for operating the solution and ensuring the new solution meets its Service Level Agreement to the same standard as any other Runkeeper application.

## A.3 REQUIRED QUALITY PROPERTIES

This section to list required quality propoerties of the solution architecture.

<b>Performance</b>	<p>Mandated performance profile:</p> <ul style="list-style-type: none"> <li>- Response time from User Stimulus requesting display of Analytics Dashboard MUST BE less than 8 seconds for 5,000 concurrent users..</li> <li>- Response time MUST BE consistently withing a bracket going from 4 to 8 seconds maximum.</li> <li>- NO TIMEOUT are allowed, regardless of the amount the data analysed and processed (i.e. regardless of the number of marathons, number of fitness data points, number of activty data points, etc.)</li> <li>- Performance MUST BE independent of processing volume.</li> <li>- Throughput of data MUST BE predicatble.</li> </ul>
<b>Scalability</b>	<p>Cost of required processing cycles and memory must be predictable, function of the number of concurrent users.</p> <p>Starting from 5,000 concurrent users, the supporting infrastructure MUST dynamically scale up 100,000 concurrent users by means of infrastructure configuration.</p>
<b>Testability</b>	<p>Beyond testing the UI features of the Analytics Dashboard, the architecture MUST allow of load testing so monitoring of service levels of the application can be automated.</p>
<b>Flexibility</b>	<p>The proposed architecture must be extensible to an Android/IOS version of the Analytics Dashboard with minimum rewrite of the core system logic.</p>

## B) Solution Description

This section to describe the key characteristics of your solution design.

### B.1 ARCHITECTURE VIEWS

This section to list all VIEWS composing your solution architecture.

VIEW NAME	PURPOSE	TARGET STAKEHOLDER(s)
<b>CONTEXT View</b>	<p>The purpose of this view is to outline the main elements at play in the design of our analytics dashboard:</p> <ol style="list-style-type: none"><li>1. Who will interact with the dashboard, performing what kind of actions</li><li>2. What will interact with the dashboard, performing what kind of actions</li></ol>	<p>Data Analyst, CIO</p> <p>Note: If the Context view was containing a Motivation Model(s), the CMO would be added to this list.</p>
<b>BEHAVIORAL View</b>	<p>List processing flows at play for the rendering RK's analytics dashboard. The purpose of this View is to outline:</p> <ol style="list-style-type: none"><li>1. How data gets collected and prepared for purposes building analytics data sets</li><li>2. How analytics data sets are served to users</li></ol>	<p>CIO</p> <p>Data Analysts</p>
<b>INFORMATION STRUCTURE View</b>	<p>List data flows play for the rendering of RK's analytics dashboard.</p> <p>The purpose of this View is to model:</p> <ol style="list-style-type: none"><li>1. Data flow of the asynchronous Batch Analytics data preparation process</li><li>2. Data flow of the synchronous request/response dashboard rendering process</li></ol>	<p>Data Analysts</p> <p>Developers</p>
<b>FUNCTIONAL View</b>	<p>List responsibilities of component groups, collaborating for the rendering of RK's analytics dashboard.</p> <p>The purpose of this View is to outline:</p> <ol style="list-style-type: none"><li>1. The structure of our design and grouping of key components</li><li>2. How presentation components are abstracted data sources</li></ol>	<p>Developers</p>
<b>INFRASTRUCTURE View</b>	<p><i>Not addressed for the purposes of this example document, but certainly should be in a real context. For examples of Infrastructure Models, refer to Lab 1.</i></p>	<p>System Administrators</p>

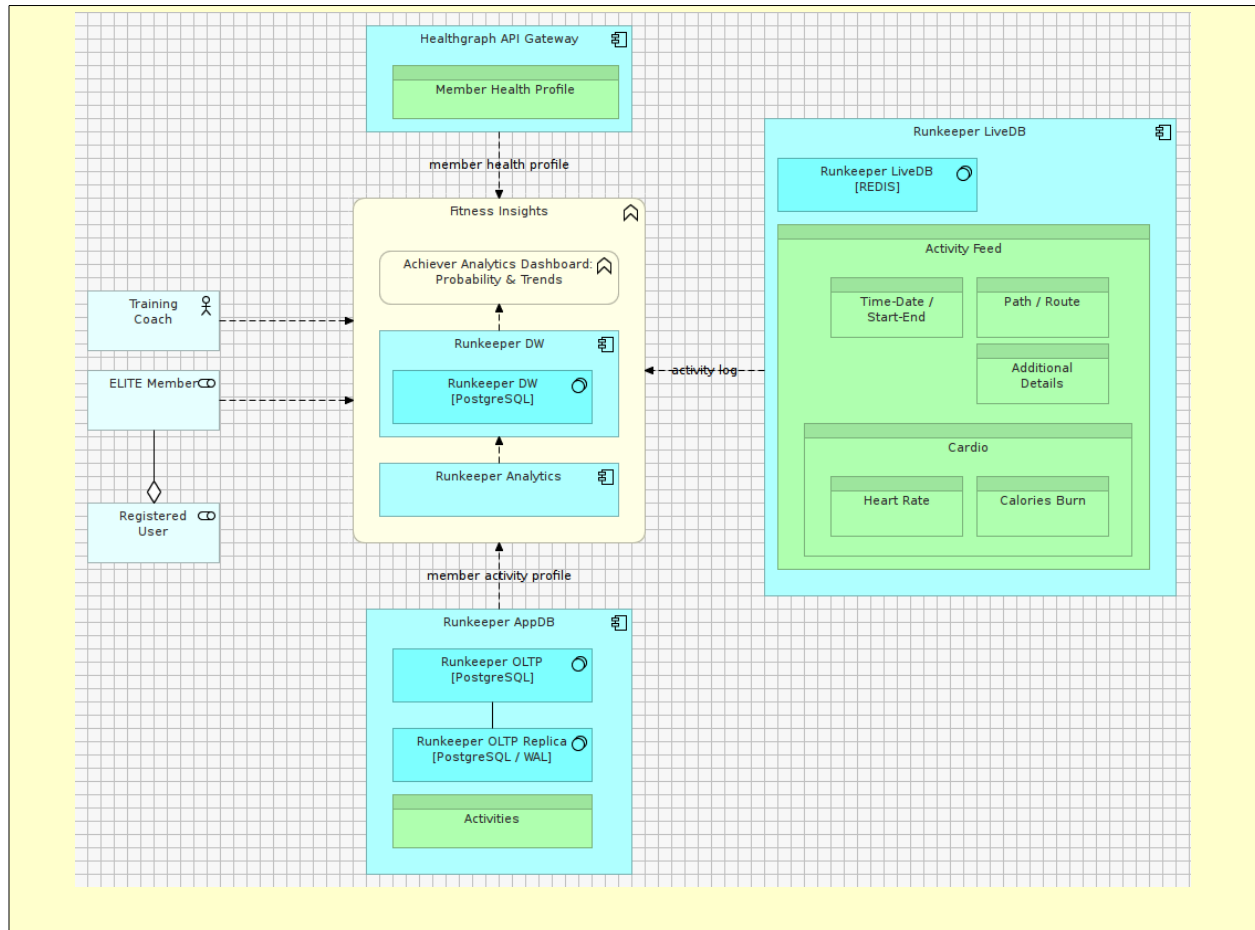
## B.2 CONTEXT VIEW

### B.2.1 VIEW INTENT

MODEL NAME	PURPOSE
<b>SYSTEM CONTEXT Model</b>	To the SCOPE of architecture solution in terms of interdependent / surrounding system components and actors.
<b>SYSTEM FEATURES Model</b>	To the SCOPE of architecture solution in terms of application capabilities and .

### B.2.2 VIEW MODELLING ARTIFACTS

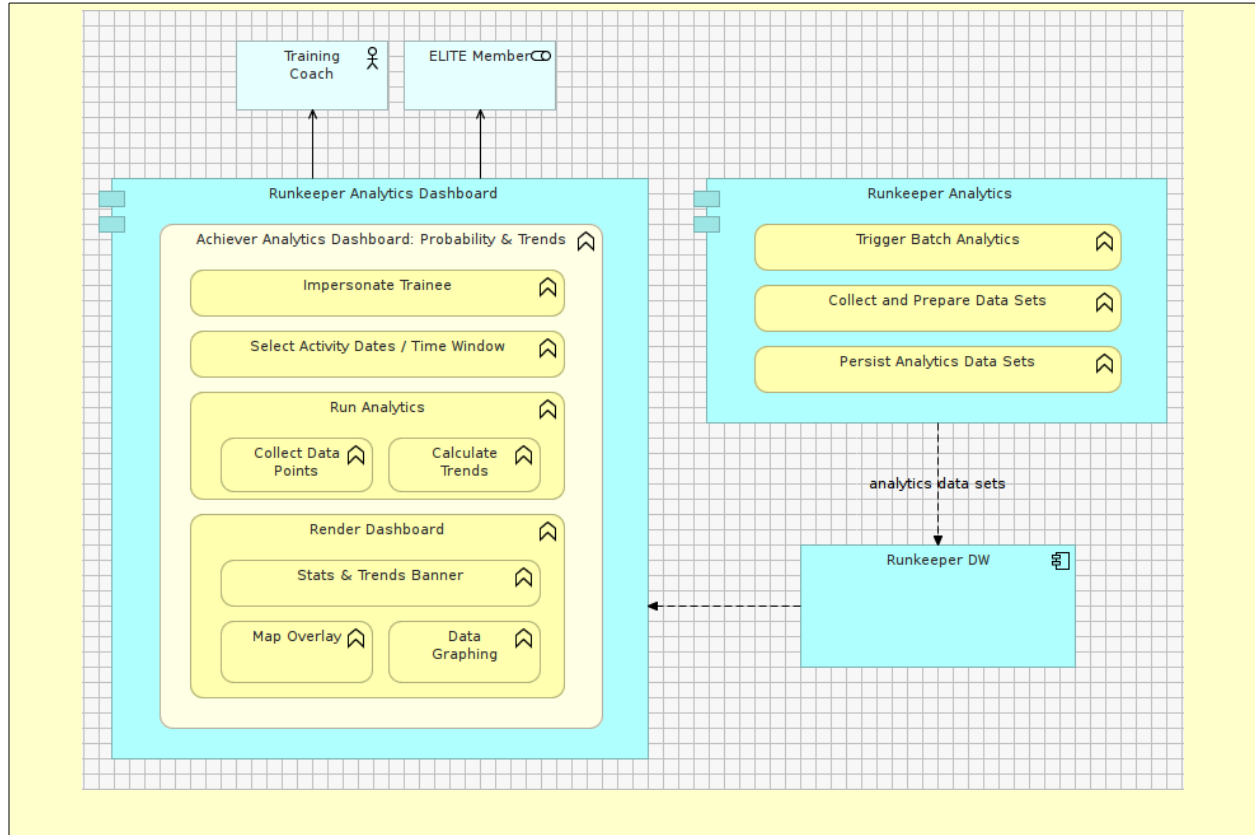
#### B.2.2.1 SYSTEM CONTEXT MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
<b>Context View Type</b>	To outline what/who will interact with the analytics platform

	architected.
<b>MODEL KIND USED</b>	<b>JUSTIFICATION / INTENT</b>
<b>System Context Modeling</b>	<p>This model shows the Achiever Dashboard Analytics system solutions in the center of the model, and its inputs and outputs from/to external factors at the periphery.</p> <p>All User data candidate for Analytics is collected and correlated from 3 distinct sources of information: PostgreSQL (User Activity / Workouts data), REDIS (User Geo-positioning / Health Tracking data points) and HealthGraph (User Health Profile data).</p> <p>Resulting datasets (prepared for Analytics Dashboard reporting) are then persisted in a single location which is the authoritative source for reporting, i.e. providing data that can be analyzed under a number of reporting dimensions.</p>
<b>STYLE/PATTERNS USED</b>	<b>JUSTIFICATION / INTENT</b>
<b>Data-driven Style: Data Factory (Data Warehousing)</b>	<p>User Data Analytics Views are prepared. Data coming from various data sources (relational and non-relational) is going through a number of preparation steps (e.g. cleansing, validation, mapping) with a view to create user-driven analytics datasets.</p> <p>The Runkeeper Data Warehouse stores and centralizes User Analytics datasets for purposes of reporting.</p> <p>These datasets are “ready” (suitable) for the Achiever Analytics Dashboard to directly consume information.</p>
<b>Domain Pattern: Shared Database</b>	<p>User Data Analytics are stored in a shared data repository, accessible from the Achiever Analytics Dashboard. Data can be pulled into the DW or pushed from external applications into the DW.</p> <p>This architecture is based on a Data Factory Style making use of a PULL mechanism (from data sources), referred as ETL (extract Transform load).</p>

### B.2.2.2 SYSTEM FEATURES MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Context View Type	To outline what are the main responsibilities for the components at play.
MODEL KIND USED	JUSTIFICATION / INTENT
System Feature Modeling	<p>This model shows that the Application is composed of two main components completely independent from one another: (1.) Runkeeper Analytics, (2.) Runkeeper Analytics Dashboard.</p> <p>(i.e. each module can be deployed and inter-changed completely independently from one another at runtime without having to redeploy the entire solution).</p> <p>More specifically, the model shows how features of the Analytics solution are decomposed and allocated to each of the main two components.</p> <p>These two components share a common database infrastructure component storing Pre-generated (i.e. prepared) User Analytics</p>

	Views.
STYLE/PATTERNS USED	JUSTIFICATION / INTENT
<b>Data-driven Style: Data Factory (Data Warehousing)</b>	N/A - Same as above/previous section.
<b>Domain Pattern: Shared Database</b>	N/A - Same as above/previous section.

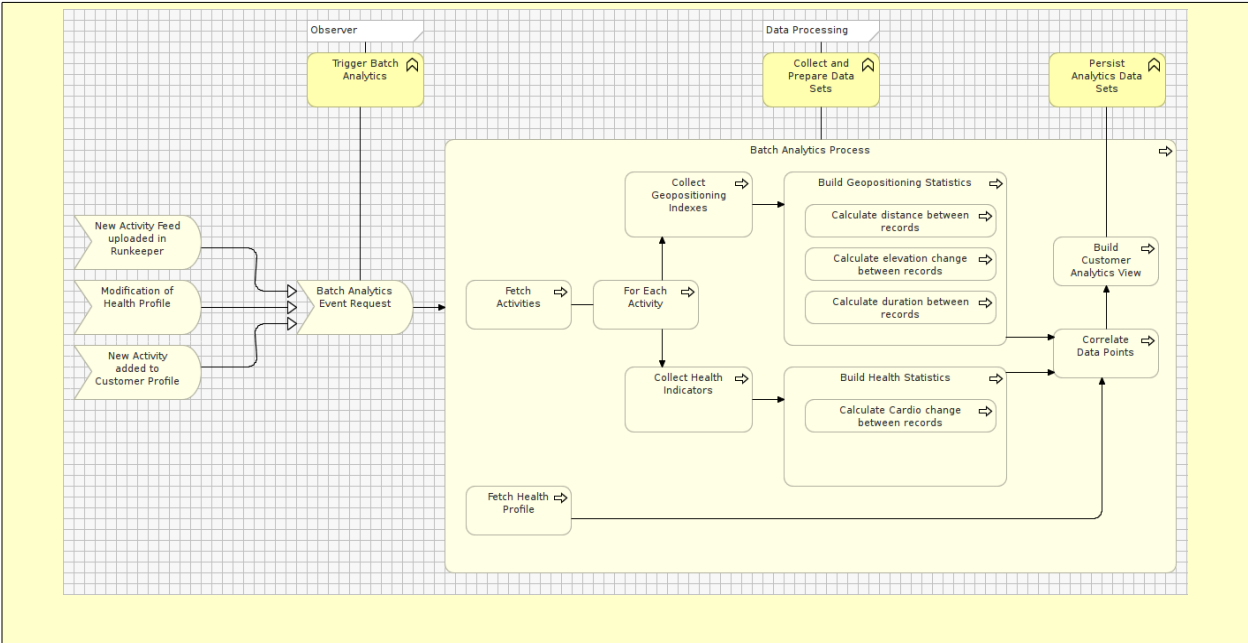
## B.3 BEHAVIORAL VIEW

### B.3.1 VIEW INTENT

MODEL NAME	PURPOSE
<b>Data Preparation</b>	<p>This model describes the data analytics preparation process. The process is triggered every time a User uploads new data points (e.g. a new activity feed, a modification of health profile).</p> <p>The flow described in this model is an asynchronous data processing daemon, observing when new user information is uploaded, then triggers an ETL data job to generate new (i.e. refreshed) User Analytics View in Runkeeper's Data Warehouse.</p>
<b>Dashboard Rendering</b>	<p>This model describes the type of tasks the Analytics Dashboard performs to fetch the Analytics Views from the Data Warehouse, and transform the data to render it as Dashboard Views – for example: a histogram chart, a pie chart, a terrain topology map, a table, other.</p>

### B.3.2 VIEW MODELLING ARTIFACTS

#### B.3.2.1 DATA PREPARATION MODEL

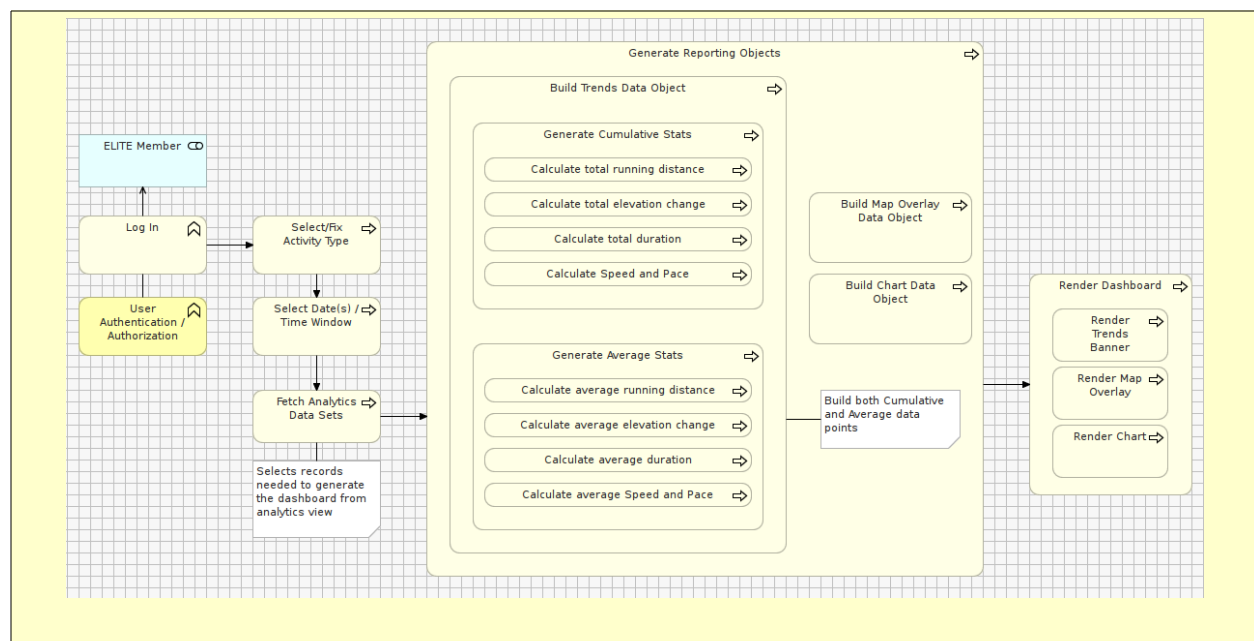


VIEWPOINT USED	JUSTIFICATION / INTENT
Behavioral View Type	<p>To outline the main flows defining how data gets collected and prepared for purposes of analytics reporting.</p> <p>To outline Triggers and Events, flow-logic and flow-synchronization.</p>
MODEL KIND USED	JUSTIFICATION / INTENT
Process Flow Modeling	<p>This model shows how a scheduled batch process monitors periodically for any new feed of data uploaded by an ELITE</p>



	member.
STYLE/PATTERNS USED	JUSTIFICATION / INTENT
<b>Process Control Pattern - Orchestrated Flow</b>	An orchestration flow coordinates the preparation of data analytics views. The process can be parallelized but central coordination is required to start processes and synchronize results in the target data warehouse.

### B.3.2.2 DASHBOARD RENDERING MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
<b>Behavioral View Type</b>	To outline the process by which data analytics get fetched and rendered on the analytics reporting dashboard. To outline intermediate business logic steps manipulating data before it is displayed in the analytics dashboard.
MODEL KIND USED	JUSTIFICATION / INTENT
<b>Process Flow Modeling</b>	This model shows the sequence of steps collecting pre-generated user analytics views, performing calculations to aggregate values, then mapping the results into a data object readily available for consumption by the Web Desktop Dashboard UI.
STYLE/PATTERNS USED	JUSTIFICATION / INTENT
<b>MVC Pattern - Synchronous Request / Response</b>	A classic Web-server based Model-View-Controller pattern to render dashboard pages based on the data collected &

	transformed from the data warehouse.
--	--------------------------------------

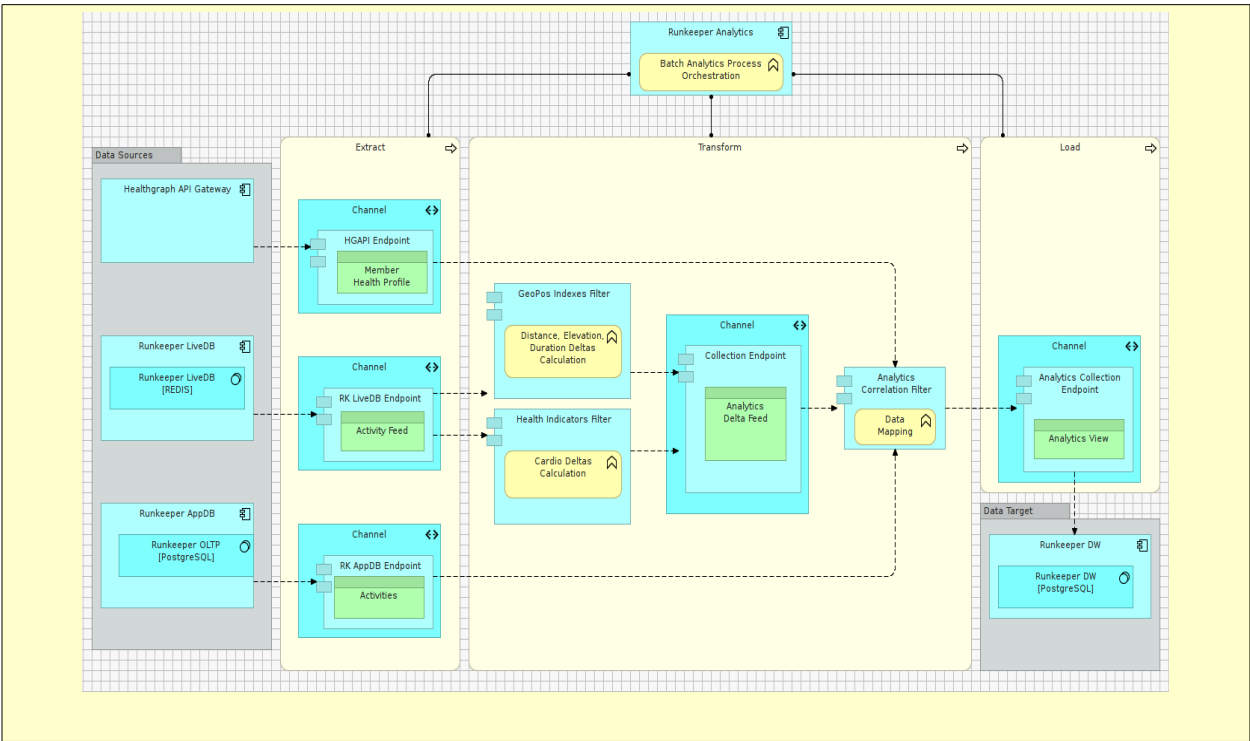
## B.4 INFORMATION VIEW

### B.4.1 VIEW INTENT

MODEL NAME	PURPOSE
<b>Batch Analytics data flow</b>	<p>This model outlines the components at play to transform data pulled from three different data sources.</p> <p>The data flow described in this model is a set of ETL jobs implemented as RDBMS Stored Procedures generating User Analytics Views in Runkeeper's Data Warehouse.</p>
<b>Dashboard Rendering data flow</b>	<p>This model presents the propagation of request/response objects throughout application components: triggered by User actions from the Dashboard Web UI, down to the collection of data, to the generation of model views, and rendering of data in the browser / from time of page rendering.</p>

### B.4.2 VIEW MODELLING ARTIFACTS

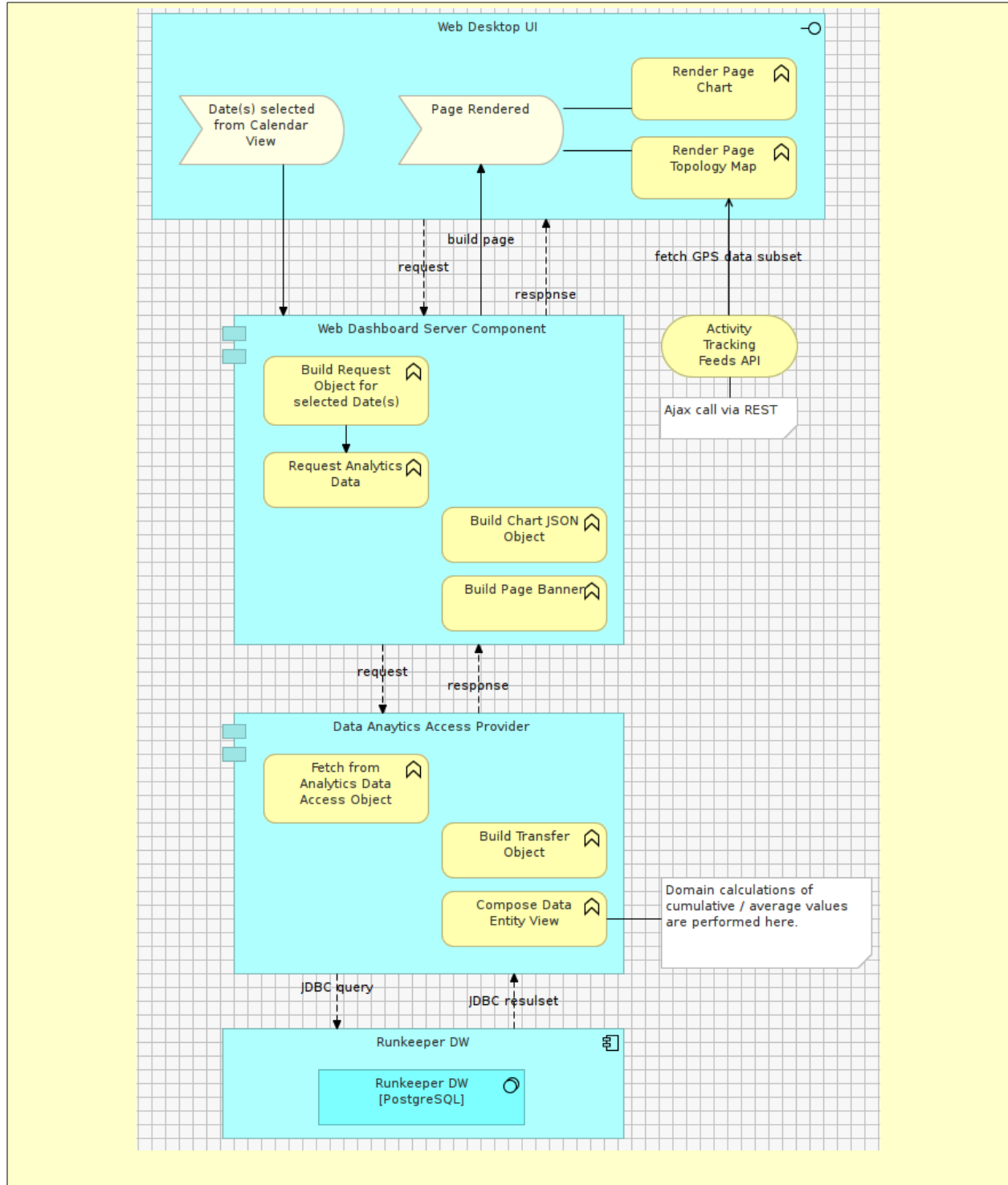
#### B.4.2.1 BATCH ANALYTICS DATA FLOW MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
----------------	------------------------

<b>Information Services View Type</b>	<p>To outline the routes of core application data-in-motion, i.e. how the batch analytics data services takes input to manipulate, transport and ultimately distribute information to target sources.</p> <p>To outline any data format, timeliness/latency, transactional integrity risks between application components.</p>
<b>MODEL KIND USED</b>	<b>JUSTIFICATION / INTENT</b>
<b>ETL Orchestration (also loosely referred as Batch Processing).</b>	To describe the type and roles applications components at play in the Extract phase, the Transform phase, the Load phase of analytics data.
<b>STYLE/PATTERNS USED</b>	<b>JUSTIFICATION / INTENT</b>
<b>Pipes &amp; Filters Pattern</b>	<p>Implementation of a message processing which is platform-independent, re-usable and extensible to generating future analytics views.</p> <p>Identify the type of transformation functions and data objects involved in Input/Output of each Pipe/Filter mechanism.</p>
<b>Message Translator Pattern</b>	Convert message payloads from 3 different sources, using 3 different communications protocols, in 3 different data schemas to one unique data model – i.e. schema of analytics views.

#### B.4.2.2 DASHBOARD RENDERING DATA FLOW MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
<b>Information Services View Type</b>	Same as described in section above.
MODEL KIND USED	JUSTIFICATION / INTENT
<b>Data Flow Modeling</b>	A classic model show the life-cycle, the triggers, and direction of data flows within the Web Dashboard application (only – the analytics preparation component above holds a different and independent process that isn't related to events or flows of this component).
STYLE/PATTERNS USED	JUSTIFICATION / INTENT
<b>Template View pattern (for Synchronous Web Page Request/Response)</b> <a href="https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller">[https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller]</a>	In this situation, some of the View, most of the Model and Controller logic are implemented on Runkeeper's JAVA server stack. The Dashboard UI is a combination of "thin" client (in which some decoration is generated by JSP server tags), and Javascript libraries (initialized with JSON objects prepared in the server BEFORE page rendering). When the page is rendered, the JS gets executed and reaches out of the JSON objects to initialize Pie charts, Topology Maps, and other visual components.

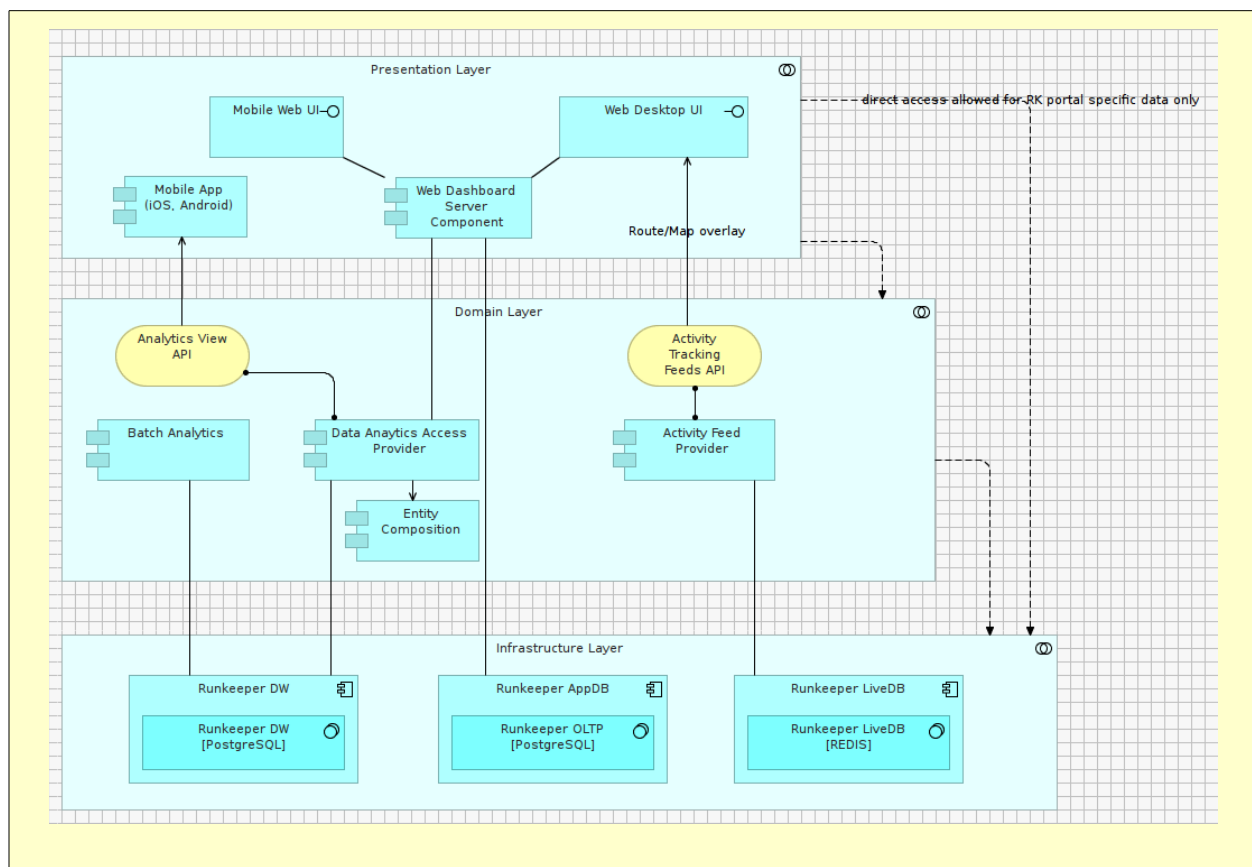
## B.5 FUNCTIONAL VIEW

### B.5.1 VIEW INTENT

MODEL NAME	PURPOSE
<b>Indirection Layers</b>	To outline the structure of the Dashboard Application (all components included) and identify coupling risks. To describe logical groupings of key components collaborating to achieve a similar function.
<b>Data Provisioning</b>	To outline what paradigm is adopted to shield application components from the specific of data storage forms/infrastructure – i.e. isolating the upper layers from the physical implementation mechanics of lower layers → data persistence model / data abstraction.

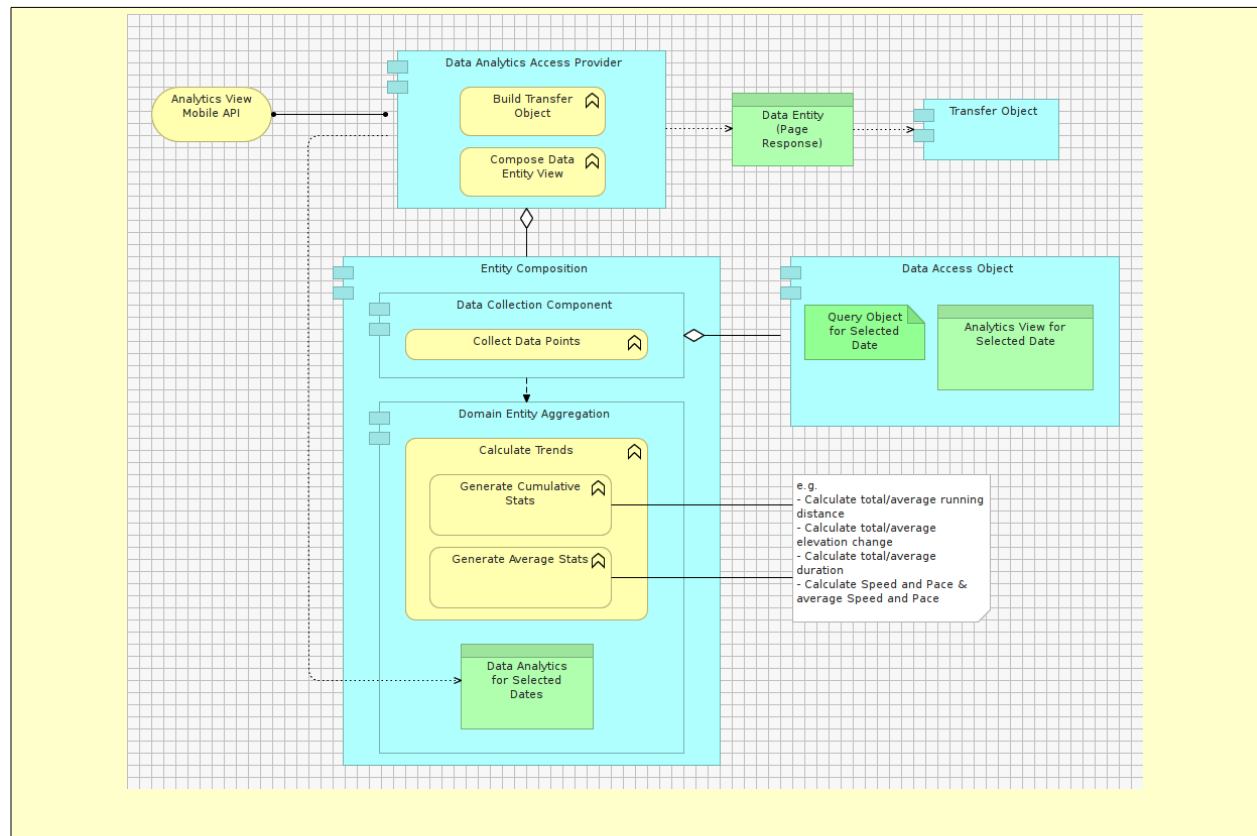
### B.5.2 VIEW MODELLING ARTIFACTS

#### B.5.2.1 INDIRECTION LAYERS MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
<b>Functional View Type</b>	To describe the composition relations between components collaborating for the specific purpose of (1,) fetching data (i.e. Data Access Objects - DAO), (2.) aggregating data into Transfer Objects (TO) propagated back up to the Dashboard Client UI.
MODEL KIND USED	JUSTIFICATION / INTENT
<b>Application Structure</b>	Used in this situation to identify coupling between layers of components of a same nature. Also used to identify servicing points and data contracts reducing coupling (typically Application Services, API, etc.)
STYLE/PATTERNS USED	JUSTIFICATION / INTENT
<b>Layered Style – Relaxed Variant (includes 2 API service points)</b>	The variant of the Layered pattern used in the Analytics Dashboard architecture aims to primarily isolate domain model components – but also to eliminate any dependency between data infrastructure concerns and user interface experience, so to prepare for Mobile UI Clients with a minimum of rework.

#### B.5.2.2 DATA PROVISIONING MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Functional View Type	Same as described in section above.
MODEL KIND USED	JUSTIFICATION / INTENT
Application Structure	Break down the structure of the domain layer to show the lower level design of data-centric components, with a specific focus on Data Analytics Access Provider component.
STYLE/PATTERNS USED	JUSTIFICATION / INTENT
<b>3 Object-Relational Data Patterns:</b> <ul style="list-style-type: none"> <li>- Data Access Object Pattern</li> <li>- Domain Entity Aggregation</li> <li>- Transfer Object Pattern</li> </ul>	<p>Data Access Object to retrieve analytics view datasets from PostgreSQL Data Warehouse.</p> <p>Domain Entity Aggregation to perform calculation / aggregations on data points retrieved from data analytics views.</p> <p>Transfer Object to feed Web Client UI JS components (and in the future Mobile Client UI).</p>



## B.6 ARCHITECTURE PERSPECTIVES

### B.6.1 PERFORMANCE PERSPECTIVE

#### B.6.1.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	<b>Need to prevent session time-out for users of the Analytics Dashboard.</b> <b>Response time of Analytics Dashboard must be less than 8 seconds for 5,000 concurrent users.</b>	<b>Decision:</b> Data Analytics View will not be generated on-demand when an ELITE User connects to a dashboard. Generating data analytics on-demand is technically feasible but this architectural options doesn't scale above 2,500 concurrent users given the current infrastructure in place at Runkeeper. <b>Trade-off:</b> Data Analytics will be generated asynchronously by a background process named (Batch Analytics).
#2	<b>Some ELITE Users will want to see their stats immediately after upload. At that point, the analytics job will not be yet complete, and the dashboard will not reflect reality.</b>	According to current usage statistics of simpler Runkeeper dashboards, 95% of Users are not checking their stats right after an Activity takes places. 5% of ELITE Users may want to see their new activity stats immediately reflected after upload of their Health or Activity data. <b>Trade-off:</b> For these users, an indicator will be displayed on the Dashboard page to notify that their last upload of data points is in progress. Eventual consistency in this case is a satisfactory option for Customers. Bottlenecks can be avoided with the Asynchronous Batch Analytics option.
#3	<b>Performance MUST BE independent of processing volume.</b> <b>Throughput of data MUST BE predicatble.</b>	<b>Ref to #1</b>
<#>		

### B.6.2 SCALABILITY PERSPECTIVE

#### B.6.2.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	<b>Cost of required processing cycles and memory must be predictable, function of the number of concurrent users.</b>  <b>Starting from 5,000 concurrent users, the supporting infrastructure MUST dynamically scale up 100,000 concurrent users by means of infrastructure configuration.</b>	<b>Decision:</b> No new scalability strategy will be introduced for the Rendering of the Web Data Analytics Dashboard. The Dashboard UI is based on a classic Web architecture sourcing data from a shared database, and as such follows the same scalability strategies as any other form of dynamic web-content currently generated by Runkeeper (using HAProxy, Load Balancer and Caching mechanisms). Runkeeper has the infrastrucutre and tools in place to deal with workload going way beyond 5,000 users.  <b>Decision:</b> A new strategy will be defined to Scale-out the

		<p>generation of Data Analytics Views.</p> <p>The Asynchronous Batch Analytics architectural option allows for a better exploitation of server resources (CPU and memory), minimizing the use of the same components and communication paths.</p> <p>The batch analytics process can be optimized for repeated processing. Contention can be reduced via replication of ETL jobs.</p> <p><b>Trade-off:</b> Analytics processes can be prioritized for Users who are waiting to see results immediately, or queued for Users that are not connected providing with: i.e. distributed processing over time.</p>
<#>		

## B.6.3 TESTABILITY PERSPECTIVE

### B.6.3.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	<b>Beyond testing the UI features of the Analytics Dashboard, the architecture MUST allow of load testing so monitoring of service levels of the application can be automated.</b>	<p><b>Decision:</b> Given above decisions and trade-offs for the Performance and Scalability Perspectives, it makes sense to automated Load Testing for the Batch Analytics process since it is the most heavily loaded component of the architecture.</p> <p><b>Decision:</b> A “load test” batch analytics process creating “dummy” analytics views will be consistently running, exercising the server components at all times. The end-to-end duration of the job will be constantly controlled as a Key Performance Indicator of the architecture. If the “load test” job falls below a certain amount of time, it means that SLA drops and that server configuration needs a change.</p> <p><b>Trade-off:</b> A new ETL Server will be acquired and act as a distribution node for other data processing ETL servers. At a minimum, 4 servers will compose the backend of Runkeeper data analytics (i.e. 1 Main Node, 3 Slaves). Steps for scaling out further from there-on are, by order of preference: (1.) CPU and Memory additions to ETL server, (2.) Insertion of new node.</p>
<#>		

## B.6.4 FLEXIBILITY PERSPECTIVE

### B.6.4.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	<b>The proposed architecture must be extensible to an Android/iOS version of the Analytics Dashboard</b>	<p><b>Decision:</b> A Layered Architecture will abstract the presentation layer from data analytics views. A data provider components will act as a Facade to access Analytics Views from any Client UI.</p>

	with minimum rewrite of the core system logic.	<b>Trade-off:</b> No Service API will be defined for the first revision of the Analytics Dashboard. The Web UI Pages will exclusively use Transfer Objects passed via Server-Side Scripting mechanisms (i.e. Web MVC). The Web UI will not use any Service API. The reason for this is that, since a Facade component acts as a gateway to access Analytics Views, it is relatively easy to accommodate an API Service making use of it. Domain Logic will not change, only the Data Provider, offering Transfer Objects passed via REST to the Mobile UI.
<#>		

## B.7 VIEW MAPPINGS

### B.7.1 For each VIEW MAPPING

Not covered in this example AD.
---------------------------------

VIEWPOINT USED	JUSTIFICATION / INTENT
<<VIEWPOINT NAME>>	<<DESCRIPTION>>

## **C) References**

This section to reference external sources of additional information relevant to the proposed solution design.