Nordea

Migrating accounting flows - from legacy platforms to Big Data

A short introduction

Goran Amcoff, BI process specialist

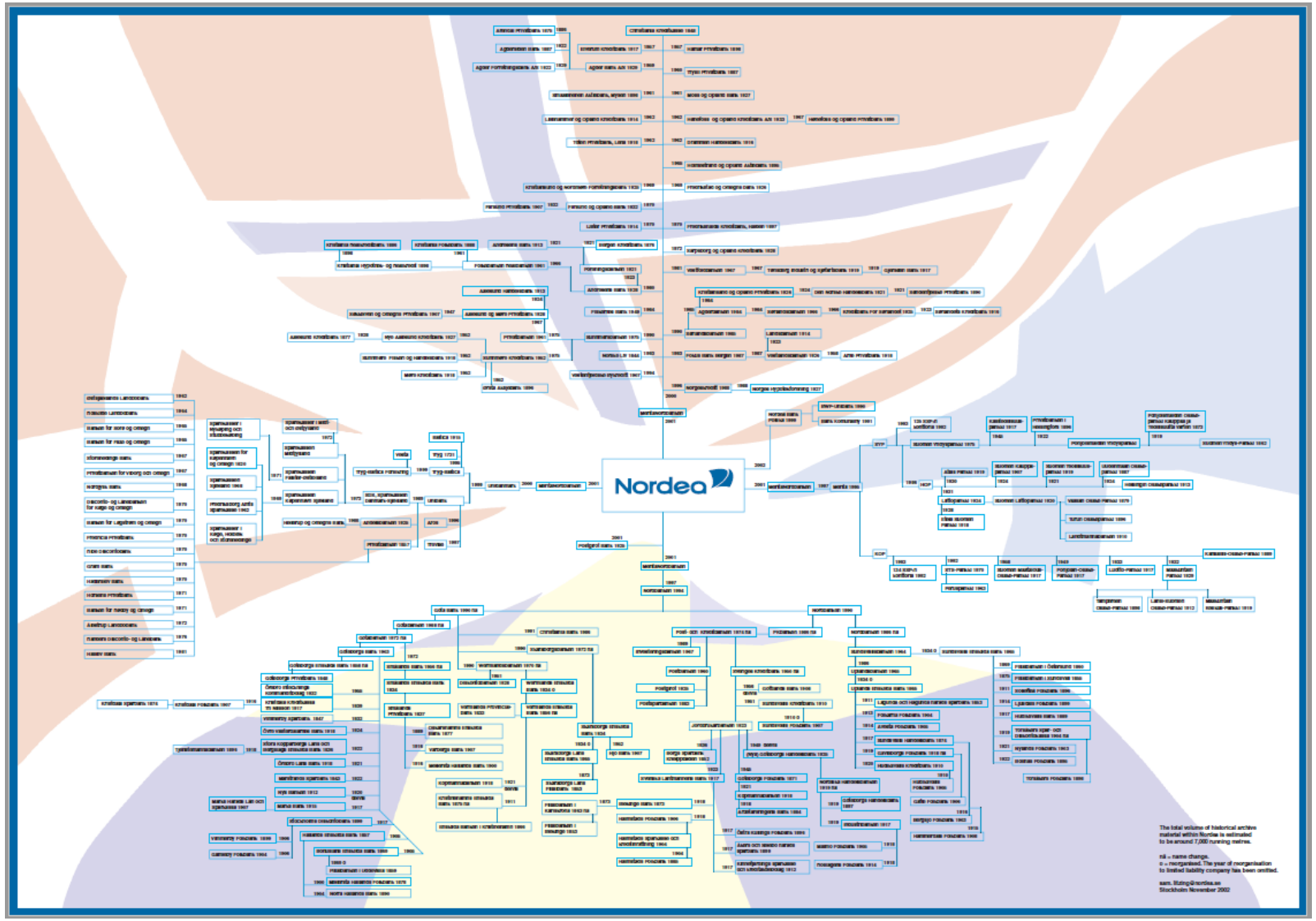
Nordea Bank – at glance

- Nordic financial services group operating from Helsinki
- >32000 employees
- >9 billion EUR revenue
- ~11 million private and corporate customers
- >600 billion $ in total assets
- Major presence in Sweden, Norway, Denmark and Finland

Mergers & acquisitions present unique challenges for accounting flows

- Mergers & Acquisitions = inheriting a lot of technology from the companies you are merging with.
- It is not easy to merge mature organizations into common accounting system.
- Accounting must work from day 1, but many things are done differently.
- Pragmatic approach: countries/branches send accounting through pre-defined API. Integration is taken care by “abstraction layer” = Nordea GAI 1.0.

Nordea family tree



Nordea

Current accounting solution – GAI 1.0

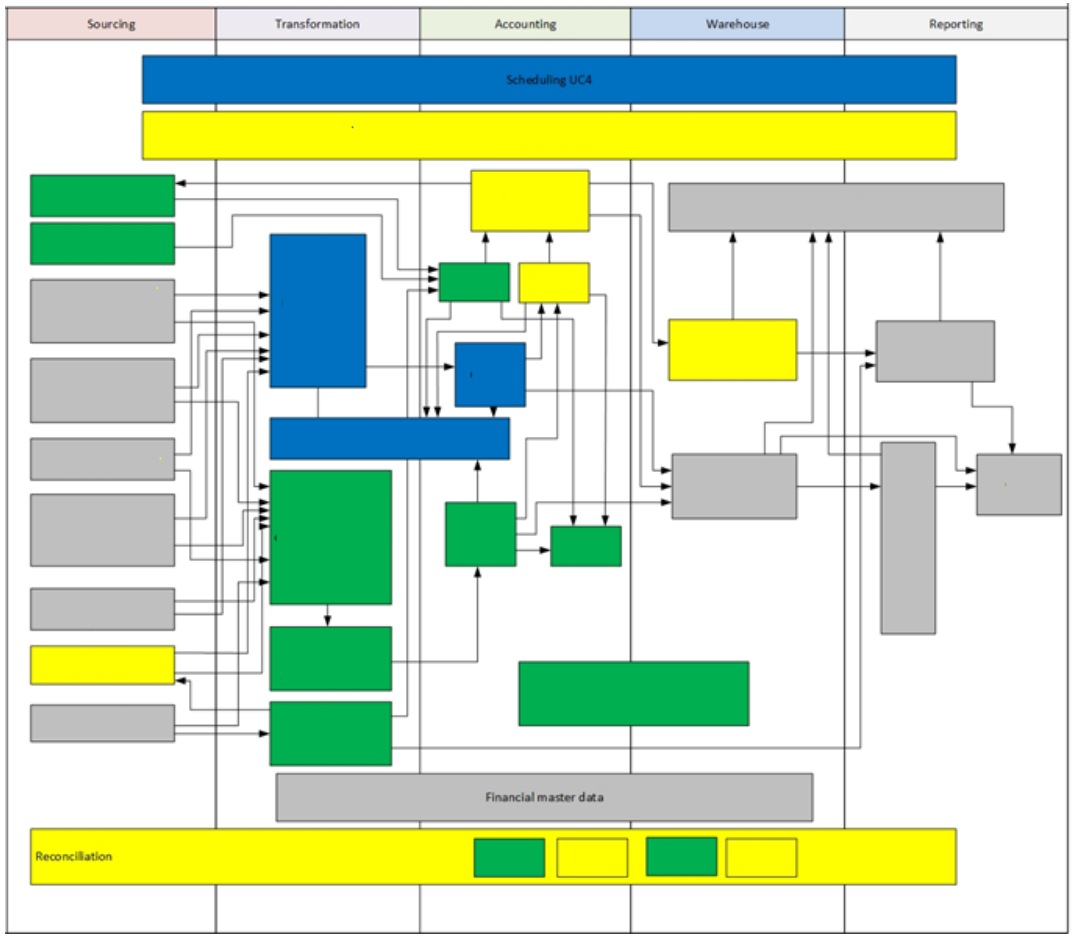
- Pragmatic solution for complicated landscape: General Accounting Interface - GAI 1.0
- Countries/Branches run different core systems, but are all delivering data in common format.
- Master Data is used for lookups and/or translation of dissimilar accounts
- Data ingest and light transformation done by TWS/Data Stage.
- Heavy logic/transformation is done in COBOL.

PROS and CONS of legacy GAI 1.0 solution

- Mainframe is proven and robust technology
- Good single thread performance.
- Stable suppliers that have been around for a long time
- Processes in place (debugging, testing, deployment etc.) have been rehearsed for years or even decades.

- Pricey
- Slow change process
- Securing resources with MF skills is getting harder
- Horizontal scaling not easy
- Most of tech stack is closed source

Accounting is complicated



Nordea

What is driving migration from GAI1

- Number of transactions in core banking systems is continuously growing. We plan for >300Mtrans/day processed within hours
- Certain operations present unique challenges: Markets produce prodigious amount of transactions.
- Legal requirements: we are required to store all transactions for 10 years, including meta-data. Archive grows continuously and storage on MF is pricey.
- People who built GAI 1.0 and understand its intricacies are retiring.
- Largest performance driver: Treasury is pushing for larger implementation of FTP (Funds Transfer Pricing). https://en.wikipedia.org/wiki/Funds_transfer_pricing

FTP

- A great tool for financial performance evaluation
- …also an formidable IT issue
- FTP can introduce a large multiplier on amount of transactions. What was 10M transactions might quickly becomes 500M transactions.
- Eats lots of CPU, requires lookup tables and consumes storage for all meta data.

New accounting platform feature list

- Audit Trail
- Transaction processing

- Newer technology (Java? x86?) * *
- Horizontally scalable in with small granularity (adding cheap nodes vs. adding new racks/10 years contracts)
- Possible to migrate to “cloud” if/when needed * *
- Able to **cheaply and reliably** store and do light processing/retrieval of vast amounts of data/meta-data (Multi Petabyte)
- Easier to use modern UI libraries (React, node.js etc) * *
- Open Source building blocks * *

GAI2 architecture

- UC4 instead of Data Stage/TWS
- Java instead of COBOL for calculating engines
- WebLogic/React for UI layers (replacing legacy UI)
- Hadoop cluster for archival storage (replacing DB2/MF datasets)
- Hadoop cluster for landing areas (replacing SAN’s and MF)
- Keep RDBMS at minimum
- REST-protocol for M2M communication (replacing scheduled FTP/flat files/ODBC etc.)
- Hadoop for Data Quality checks

Challenges/lessons re: GAI2

- You still need relational databases, Hadoop alone is not enough. (Joins are easier in SQL than Java/Map-Reduce).
- Developers should work near process specialists and SME's. Otherwise they might assume things. (...what needs to be logged and what can be discarded)
- If it works in demo it might not work with production volumes. (Java garbage collection, performance, DB logs etc).
- Performance profiling: system must be able to handle volumes within reasonable time window…
- Optimization skills are still very much needed, some things are not solvable by more hardware.

Showcase for Big Data tech stack: TRAT2 (TRansaction Archive Tool 2.0)

TRAT2 is tool for tracking and storing accounting data. It is meant to replace TRAT1.

Legacy: TRAT1 - uncompressed datasets on mainframe, Web-based UI. Data Stage/TWS used for ETL, flat files for M2M communication.

- Business search (search for any field) not possible.
- Search speed varies, usually minutes.
- Search for historical data very slow (>20 minutes)
- Not easily scalable performance wise. Scalable capacity-wise.
- Old data “nearline”, not “online”.

New: TRAT2- Cloudera Hadoop cluster, Apache Hive, Apache SOLR indexer, React/Node.js for UI. UC4 for ETL. REST services for Hive<->Web Services communication.

- Business search possible , logical and text operators (LIKE, <,> etc.) possible
- 50 - 100 times (!) compression ratio compared to TRAT1 due to data being highly formalized and repetitive.
- Horizontally scalable. Need more storage/CPU? Add more nodes…
- Schema on read: Transaction files are stored “as is”.
- Huge performance increases when it comes to search times for indexed “hot data” (<3 sec).
- Relatively good search times for “cold data” (<5 mins).
- All data is “online”.

Challenges re:TRAT2

- Which Hadoop release/version/package to use? Cloudera? Map-R?
- Hadoop concept differs from legacy hardware. Infrastructure people must understand it (lot's of small/simple nodes instead of expensive/powerful servers.)
- Making Hadoop resilient: node replication factor? Failover strategy? Offline backup?
- How do you display large volume of data through React-based UI without crashing the browser or running out of memory on server side?
- Indexing vs. "cold search": how many generations of data are to be indexed (=requires extra storage/processing) vs. just stored and searchable via "table scan"
- Ingestion/migration of >1PB from legacy system (TRAT1):
  - Do you convert data to "intermediate" format?
  - How to transport 10^{15} bytes and make sure data is intact (checksums, data sampling)
  - Ingesting and indexing 10^{15} bytes…

Do not forget about people doing the work!

- People management is as important as technology.
- People working with “legacy” systems mare just as important as those developing new system! **Who will tell you how it works**? Avoid “Us and Them”.
- Create cross-functional teams. Do not forget to involve business people. They might not know much about technology but know a lot about what system should do and what is important and what is not.
- Beware of vendor lock-in and outsourcing both architecture and execution. Key stakeholders/drivers should come from the organization.

Q/A