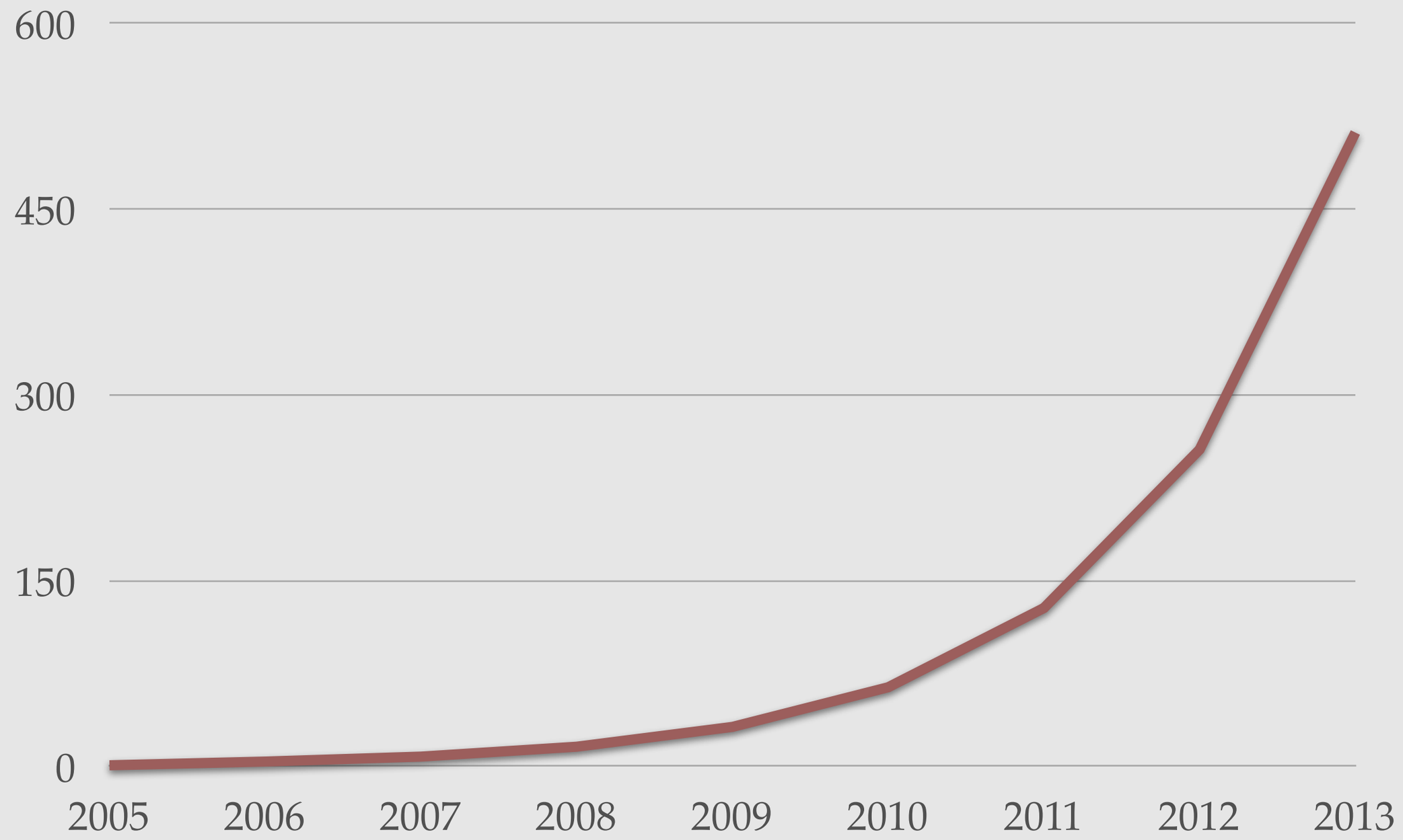
Of Heisenberg and Hawthorne - measurements, visibility and guidance

Cons T Åhs
@lisztspace

Exponential Growth probably means Success



Success does not mean that you do not have problems

- Doing the right things doesn't mean you are doing things right
- Choosing the right technology isn't enough - you can write bad programs in any language
- Fast initial growth with
  - more merchants
  - more customers
  - expanding into new countries
  - more products
- Lots of challenges..

Too Agile?

- Fast pace with steady flow of new requirements
- Developers want to deliver on expectations
  - short term delivery is more important than long term
- Solve problems with available resources
  - problem: we want more productivity
  - with money available, adding more developers is easy
  - not the best idea (The Mythical Man Month, 1975)
- Long term quality and adaptability suffers
  - the slowdown is gradual
  - productivity/developer vs productivity for a growing development department

Klarna

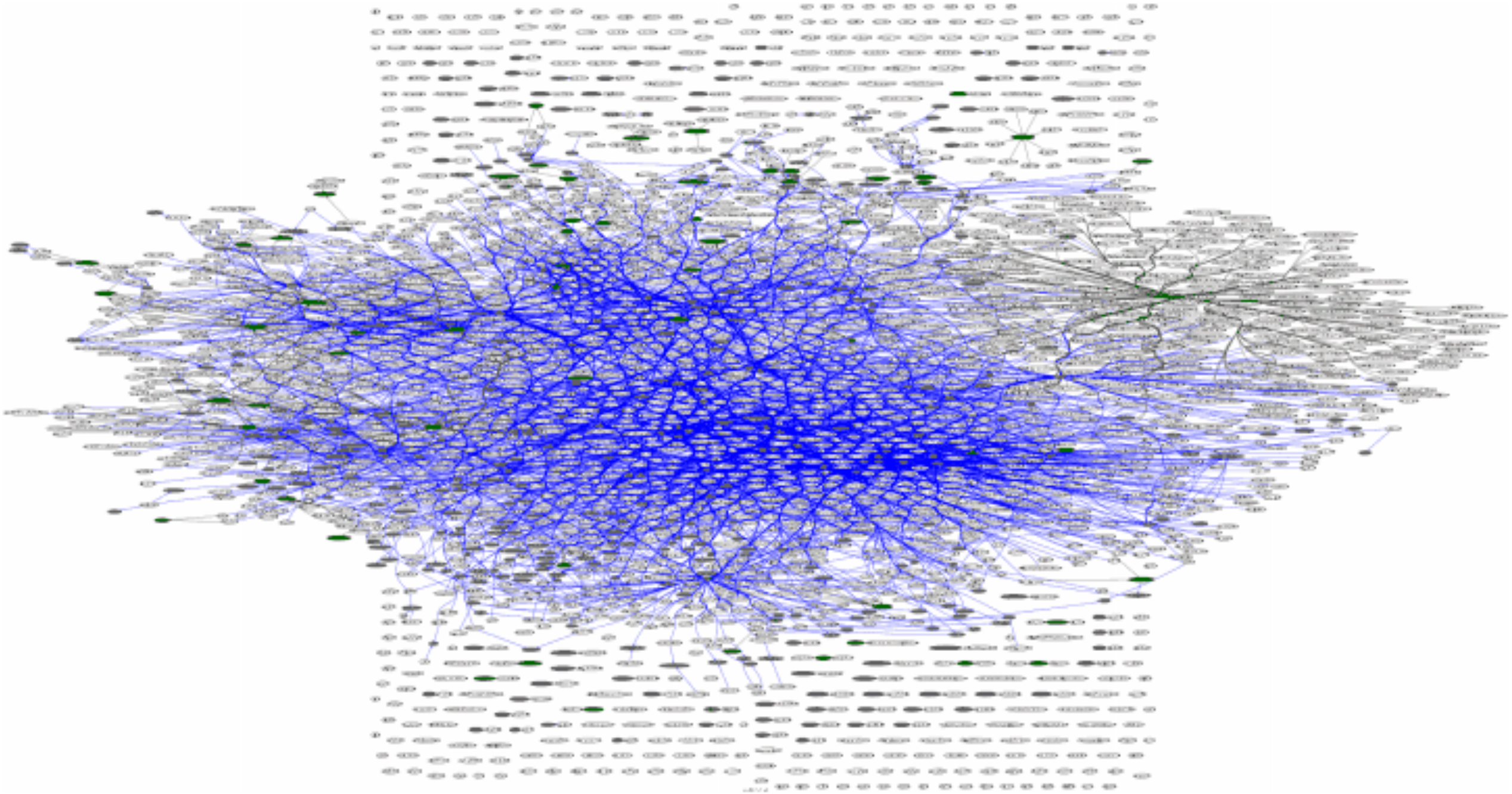
Evolution vs Design

- Evolution means adaption
- Having something to adapt to is good - it means you are successful
- Evolution leaves artifacts
  - best case is that your appendix does nothing, but it might actually kill you

- Design carries more purpose and a sense of direction
- Adaptability might suffer
- You can’t design for every possible change

Klarna

Evolution and Adaption gives us..



Zoom in..


estore_lib_xmlrpc.erl
kdb_export_fsm.erl
contact_compare_at.erl
wtwiki.erl
gmu.erl
dog_contact.erl
kdb_sup.erl
doc_gen_util.erl
op_stat_file.erl
contact_test_data.erl
contact_compare_no.erl
contact_compare_nl.erl
contact_dict.erl
comp_list.erl
klog_govern.erl
soliditet_cu_info_fetcher.erl
cell_test_data.erl
contact_compare_de.erl
contact_filtering.erl
kstore.hrl
ko_tuple_exporter.erl
bankfile_store_key.erl
zip_test_data.erl
contact_compare_fi.erl
nl_sc_cm_default.erl
contact_sources.hrl
risk_reports.hrl
estore_lib_verify.erl
contribution_model.hrl
restapi_services.erl
file.hrl
product_category.erl
estore_lib_rejection.erl
hare_utils.erl
ocr_finder.erl
bix_kcase_stat_bo.erl
soapbox_type.erl
risk_reports_denmark.erl
incoming_payments
iso4217.hrl
scorecard.hrl
tulib_gen_cache.erl
contact_compare.erl
bix_pclass_stat_bo.erl
risk_reports_netherlands.erl
risk_reports_germany.erl
ing_gmu.erl
tulib_gen_db.erl
policy_data_pd.hrl
os_cmd.erl
ing_ftp.erl
bix_full_export.erl
meshup_promises.erl
tulib_maybe.erl
iso3166.hrl
currency_file.erl
api_version.erl
pulib_hofs.erl
bix_inv_stat_bo.erl
bank_integration.erl
n_lock.erl
meshup_resolver.erl
policy_lookup.erl
scorecard_bif.erl
policy_data_db.hrl
contact_reader.erl
bix_inv_return_stat_bo.erl
bix_delayed_bo.erl
kred_upg_lib.erl
tulib_util.erl
meshup_matcher.erl
pcontact_ops.erl
estore_config.erl
emsg.erl
risk-671-nodob-scenario-performance_UP.erl
tulib_processes.erl
meshup_caps.erl
var_engine_ctx.erl
person_test_data.erl
contact_sources.erl
bix_pacc_purchase_stat_bo.erl
intermediate_vars.hrl
policy_engine.erl
acceptance_vc_ops.erl
fee.erl
kcounters.erl
bix_pacc_stat_bo.erl
payment_src_account.erl
ts.erl
meshup_lint.erl
acc_scorecard.erl
bankfile_store.erl
bo.erl
estore_common.hrl
intrum_file.erl
meshup_lib.erl
var_fun_checker.erl
pno_cno.erl
person_resolver.erl
kcrash.erl
se_sc_cm_default.erl
no_sc_cm_default.erl
fraud_engine.erl
kuser_api.erl
normalization.hrl
kula_cache.erl
merp_lib.erl
site_country_util.erl
bg.erl
pno_yaws_util.erl
person_static.erl
currency.erl
sftp_client.erl
fraud.erl
goods.erl
kdb_users.erl
fi_sc_cm_default.erl
co_contact_lookup.erl
idata_server.erl
note.erl
kcase_reconciliation.erl
types.erl
periodic_fee.erl
var_lib_contrib.erl
risk_reports.erl
alarm.hrl
sie.erl
parse_german_db.erl
pbal.erl
risk_reports_lib.erl
checkout_payment_settings_yaws.erl
contribution_gui_cb.erl
inet.hrl
euser_info_yaws.
var_lib.erl
contrib_vars.erl
pacc_stat.hrl
fix-667-swap-order-on-split_UP.erl
contract_yaws.erl
logic24_estore.erl
scorecard_gui_cb.erl
scorecard_setup.erl
pclass.erl
sell_archive_invoices.erl
transaction_report.erl
cases_list_yaws.erl
site_state.erl
payment_src.hrl
scorecard_parameters_yaws.erl
contact_tags.hrl
suspect.erl
merp_reservation_yaws.erl
payment_src_kd_pay_legacy.erl
invoice_return.erl
euser.erl
kinv_mk_yaws.erl
store_costs_list_yaws.erl
var_lib_misc.erl
ptrans.erl
fraud_gui_suspect_list_export_yaws.erl
contact_private.hrl
send_letter.erl
pacc_invoice.erl
etrans.erl
invoice_goods_yaws.erl
reports.erl
supro.erl
kdb_sync.erl
scorecard_variables.erl

Problems..

- Dependencies way too complex
- Code base shared among too many developers - everybody “owns”, and changes, everything
- Development does not scale
- Testing mostly done at system level, not a unit level
  - difficult to test parts in isolation
  - turn around times are long
- Code smells
- Rigid culture of how development and testing is done
  - difficult to change, both in terms of code and behaviour

Klarna

Technical Debt

- Technical Debt is like taking a loan when buying a house
  - Instead of having everything upfront, you can get what you want sooner.
  - You have to pay you mortgage
  - Be careful about taking more loans if your income does not match
- Any product will carry technical debt
  - Time to market over The Perfect Product
  - Adaption to insights gained and new directions
  - Not knowing where you are going from the start

Klarna

Solving Technical Debt is Trivial..

.. if you have a time machine ..

Klarna

The Turning Point

- Be open about your technical debt
  - to yourself, the team, management
  - if not, it is reasonable to assume that a solution has been made according to best practices
- Make it clear that some choices done for short term gain will increase the technical debt
- Change of directions will increase technical debt
- Increasing technical debt will slow you down
- Not decreasing technical debt will keep you slow

Klarna

Measure, don't guess!

- Technical debt needs to be quantified
- Monitoring change shows if you are going in the right direction
- Know where to focus your efforts
  - bad spots
  - prepare for future change
  - be pragmatic; don't fix things that aren't touched

Klarna

Observation Effects

- This is Heisenberg and Hawthorne in action
- Doing measurements will have effects on behaviour
  - you want to look good according to the metrics
  - doing good code reviews will increase quality
- Effects are not better than the measurements
  - anything can be gamed
  - trends are often more important than single values
  - secondary, long term, effects show effectiveness, e.g, does increased code coverage actually result in better quality?

Klarna

Possible Metrics

- Quality
- Productivity
- Static code analysis
  - dependencies
  - access patterns
  - type checking
  - code smells
- Code coverage by unit tests

Klarna

Tools
Dependencies

- Define an additional module hierarchy
  - applications are collections of module
- We measure bad call dependencies in our code
  - Allow only certain modules (the API) in an application to be called from other applications
  - Define layers in the system and define how layers are allowed to call each other
  - Compute the call graph and find violations to the rules
  - Works on beam code

Klarna

Tools
Dependencies

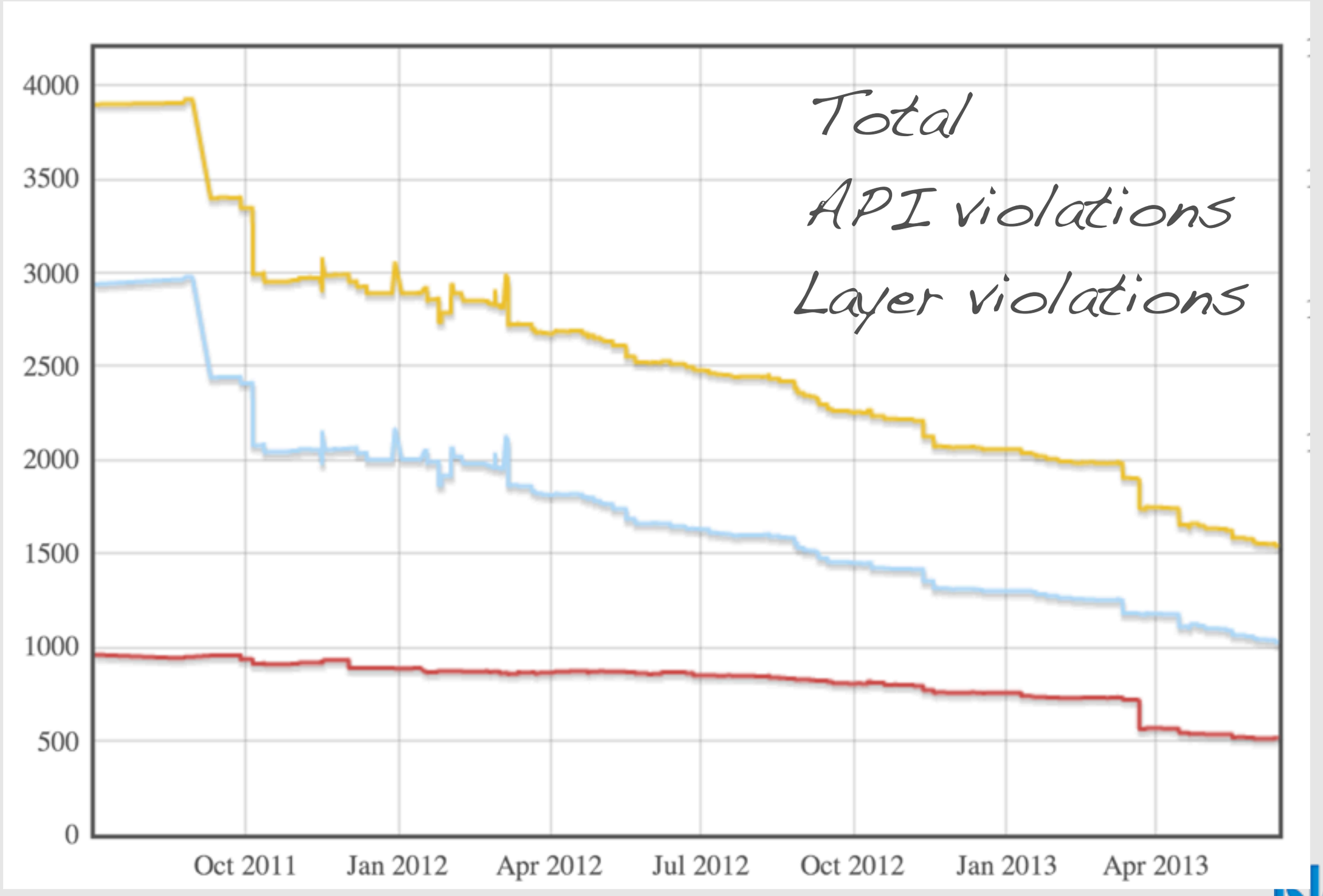
- Effects of measuring
  - Prevent new bad dependencies
  - Encourage active work on removing bad dependencies
    - define and use APIs
    - move and refactor code

Tools

- Find invalid record access from other modules
  - Using knowledge of actual representation is long term bad - you will change representation at some point
  - Prevents easy representation change
- Measure code coverage to encourage writing unit tests
  - currently measure coverage from eunit
  - adjust rebar for more precise measurement

Klarna

Count of Bad Dependencies



Results

- Teams that actively work with increasing code coverage from unit tests have less incidents
  - secondary effect of initial effort
- Bad dependencies are decreasing
- Sense of code ownership is increasing
  - “sharing” ownership between teams is a bad idea
- Pride within teams are increasing
  - produce at good quality

Klarna

Closing Remarks

- You will have technical debt - don't pretend otherwise
- Control you technical debt - or it will control you
- Measure you technical debt - don't guess
- Measurement will have effects
- Changing the culture is difficult and slow
- We are building a new architecture while running 24/7
  - This poses it's own set of challenges
  - You "legacy" system will live longer than you expect and want
  - The Second System Syndrome is real

Klarna