**Notizen zu: SECOMO – Cost Model for estimating eco-costs of software-driven systems**

Abstract:

* Concepts: for description of ecological costs
* Models: auxiliary(?) models expressing ecological costs & circumstances causing them
* 🡪 Algorithmic eco-cost estimation models (taking auxiliary models as input)

Goal: Show impact of development decisions IN EARLY stages! (based on information available)

* Approach (Secomo) of the thesis = fulcrum (?) for green software engineering methods, estimate eco-costs before implementation has started

Übersicht:

Foundations:

* Green ICT
* Estimation Techniques in Software Engineering
* (Example: Shopping Kart example specification - KobrA)

Expressing Eco-Costs with SECoMo

* Core Concepts
* Eco-Cost Metrics
* Auxiliary Models

Estimating Eco-Costs with SECoMo

* Basic Concepts: Estimation Techniques
* Early Estimation of Eco-Costs
* Estimating Eco-Costs on the Intermediate Level of Detail
* Estimating Eco-Costs on the Advanced Level

Using SECoMo

* Leveraging SECoMo in Software Engineering
* Evaluation

# Introduction

* Software in our lives
* Software & Global warming / greenhouse gas emissions / Co2 emissions -> software plays a role
* 2 strategies
  + Minimize amount of energy consumed by ICT
  + Increase amount of renewable energy used by ICT
* Video on Demand example – align energy consumption w/ green energy production
* Efforts to counteract
  + Reduce energy of data centers (bas10), (ber12), (zel14), …
  + Projects about better alignment of consumption & production, e.g. about
    - Virtualization -> enable use of unused hw(?)
    - Cooling systems (data centers)
    - End user products / HW 🡪 increased energy efficiency
* BUT – software never had the same big focus before – just now starting 🡪 Green / Sustainable Software Engineering; ICSE / IEEE Software – now deal with it
* Aspects
  + Life Cycle / Process models with SUSTAINABILITY AWARENESS
  + Analysis: which effect do design patterns, programming languages, code refactorings have (Nau11, BS13, Nou12a, SPC14) on ENERGY CONSUMPTION caused by SW
  + Measurement / Estimation tools (WGR13, Kon08, Seo09a) (JouleUnit)
  + Sustainability Metrics (Joh12, MP15)
* Why not more / further evaluated, sophisticated solutions? 🡪 EXPENSIVE! (late in the process, change means costs!)  
  “most of the approaches developed so far focus on optimizing the software during the implementation phase” – better would be possibilities to do it EARLIER! Already in design phase / requirements, etc. – but that requires a way to estimate it 🡪 “analyzing potential optimization early in the sw development process”   
  - some models exist – e.g. greensoft or other lifecycle models   
  – but lack in concrete methods to estimate actual costs to have hard facts

- + NEED to create awareness in the earliest stages!  
- most methods add further complexity with additional documents, not integrate it in sw development process (or not so much)

* 🡪 SECOMO provides a method for that – **Software Eco-Costs estimation model**
* Research Challenges
  + **Unknown ecological impact of development decisions** – hard to say (yet) which impact certain decisions have, thus it’s hard to consider trade-offs to reach higher sustainability / energy efficiency
  + **Inability to define and communicate reasonable eco-goals for programmers** – lack of methods / models that are appropriately integrated
  + **Lack of information** – of what influences sustainability / eco efficiency, early in the sw dev process, hard to define SLA’s etc.
  + **Lack of tailor made metrics** – many metrics suggested; but often very generic, or not fitting for early software development phases 🡪 better: to have metrics expressed in sw engineering concepts (e.g. specifications)
* Research Goals
  + Estimation of Eco-Costs from Different Viewpoints -> C1, C2
  + Better Eco-Cost Metrics -> C4
  + Better Communication of Goals, Trade-Offs and Ecological Impacts ->C1, C2
  + Adaptability (of estimation approach) -> C3

**SECoMo – Software Eco-Cost Model**

* Software Eco-Cost ESTIMATION model, approach for / supporting sustainable software engineering!
* Auxiliary Models (“Hilfs-“modelle) 🡪 enable the representation of eco costs + causes!! 🡪 for a clear communication!
  + Enhance traditional sw specifications (KobrA OR A SIMILAR specification technique?!) with eco-costs
  + Other models can be derived from them (Green Specifications)
  + …
* Estimation Models 🡪 mathematical estimation model: costs of a software system from several perspectives and in several (early) stages
  + ESTIMATE eco-costs that will actually arise when a SW is executed
  + “all levels of granularity” – whole system down to single components, e.g. data
  + Worst-, Best-, Average eco-costs for
    - All HW platforms (?)
    - Different user types (?)
    - Different points in time
  + 3 levels of detail
* Basis: Eco-Cost Metrics 🡪 describe eco-costs in a unified way
  + 11 new metrics
* 🡪 SECoMo allows developers to measure/estimate/realize impact of their design decisions early + enable (cheap) changes, inform them + support them to make “well informed choices”

# Foundations

## GREEN IT

Green ICT

* Starting point: Study by Gartner (Pet07) 🡪 ICT is “a MAJOR contributor” to GHG emissions / global warming, 0.5 % of worldwide emissions in 2007 🡪 thus eco-efficiency of IT MUST be improved
* GREEN IT.. “this will help explain the context of this thesis in relation to other software related approaches and clarify how they can enhance each other” ?!
* (MP15) Green-by-IT and Green-in-IT
  + Green by IT: use IT to reduce greenhouse gas emissions (e.g. use video conferencing instead of face-to-face meetings -> travel)
  + **Green in IT: Increase energy efficiency** of IT artifacts itself (HW, SW) **(focus here)**
* **Optimization** **of data centers**
  + Better infrastructure -> all hardware elements improved in energy consumption
  + PUE metric – how much of DC energy consumption is caused by IT? (AVE+12) 🡪 goal: PUE of 1 = all energy is consumed by IT, i.e. not other cause (like ventilation)  
    🡪 not perfect, not totally accurate, BUT: helps to raise awareness !
  + Best practices: “free cooling systems”, ice storage facilities (to store energy?), solar panels, higher average room temperature (cold not necessesarily needed) …
  + CUE (Carbon Usage Effectiveness) (Aze+10) – how much of the Carbon emissions caused by the DC is due to IT energy consumption
* 🡪 INCLUDE impact of HVAC systems in software eco-cost estimation!
* **Optimization of ENERGY need**(?) of HW – servers
  + E.g. improvement by HP (Hew16); high efficiencies CAN be reached (80 Plus Titanium)
  + Causes: each component could be responsible
    - PSU (power supply unit)
    - CPU
    - Hard disks
  + Approaches to increase CPU efficiency
    - New processor designs
    - Dynamic frequency scaling
    - Disable unneeded cores
* **Optimization of ENERGY need**(?) of HW – components in other electronic devices (laptop, smartphone, tablets) + TV’s, Game Consoles, …
* **DISADVANTAGE:** 
  + Improved HW means NEW hardware -> issue of IT disposal, waste!!!
* 🡪 another important aspect: improve EXISTING HW, leverage it!
  + Virtualization – share resources & allow better alignment, use of unused resources, … 🡪 reduces IDLE energy consumption as LESS HW is needed overall
  + E.g. FIT4GREEN project (Bas+10), All4Green (Bas+13), DC4Cities (Kli+15)
  + Other approaches: Frequency scaling, Dynamic voltage scaling, Standby modes
* 🡪 INCLUDE consideration of these approaches in software eco-cost estimation!
* Optimization by: using SLA’s
  + Sla’s grant the customer a specific level / quality of a service (source?)
  + GREEN SLA’s enhance SLA’s concerning more environmental issues
    - E.g. define LOWER availabibiliy in the night -> allow for better options to increase energy efficiency
  + CLEAR STATEMENTS needed!

Green Software

* Many approaches developed – can be combined with SECoMo!!
* Overview
  + GREENSOFT – reference model, all lifecycle phases  
    🡪 development process enhanced by agile sustainability process (reviews etc.)  
    🡪 create awareness, measure(?) it, communicate it with all stakeholders  
    NO CONCRETE metrics, methods, models about ecocosts…
  + Many models about IMPLEMENTATION phase
    - SEEDS – energy-optimiuation decision support (MPC14)  
      🡪 code optimization to achieve energy consumption improvement
    - General: code optimization not focused on energy efficiency as it is hard to grasp
  + Influence of Software design patterns on energy consumption
    - Lit05, Sah12, BS13
    - Yes, they can have a huge influence
  + Influence of Code Refactorings
    - SPC14
    - (?)
  + Effect of Programming Languages on energy usage
    - Nou12b
    - E.g. pearl high; C, C++, Java,… low
  + Effect of certain components, algorithms, browsers, (von mir: dev. Environments!),
  + Energy MEASUREMENT in general – PREREQUISITE
    - Can be by direct use of HW, by estimation or by simulation
    - GreenTracker
    - Bun09;
    - LEAP framework (MPC14)
    - PowerAPI (estimation) (Nou12b)
    - JouleUnit (WGR13)
  + REQUIREMENTS phase
    - How to better consider sustainability concerns? Pen16   
      🡪 EnvironSiSE and AMDiRE approach for relevant artifacts
  + Green Specifications
    - Eco-cost parameters as part of software specifications
    - Can support SLA’s
    - 🡪 HOW TO OBTAIN the eco-cost information?
* OR – quite a different thing
  + Measuring social & economic impacts of green software   
    MP15, ASK14, Sah14

Green Metrics

* Literature survey: Morage, Bertoa (81 measures?!)
* Example Energy Efficency Measures
  + PUE Gro13
  + Power Consumption Hin12a, Hin12b, KLL12, Nou1b
  + Useful work done Joh12
  + Energy Costs at different levels SMM08b, Seo09b
* Resource Optimization measures
  + Response times (mar12), (kip11b)
  + Availability (Kip12)
  + Learnability (alb10)
* Capacity
  + Workload (goi13)
  + Percentage of used functionality (LFF13)
  + Used functionality fixed to relative energy ratio (Gro13)
* Perdurability
  + Defect density (Alb10)
  + Reliability (Kip12) (Kip11b)
  + Modification requests (Sea03)
* Literature review Bozelli et al (BGL13)
  + 66 metrics for energy consumption of software
    - Energy
    - Performance
    - Utilization
    - Economics (dollar)
    - Performance/energy
    - Pollution (Co2 units)
* Metrics either too abstract or too detailed, too specific for one problem :/

## Estimation Techniques in Software Engineering

Development Cost and Effort Estimation

* Types of Classifications: Algorithmic & NON-Algorithmic Models; (or other classifications, see Boehm or other authors)
* Algorithmic Estimation Models
  + Mathematical models, based on experiments / historical data 🡪 consider cost drivers; basic factors: Size and Complexity
  + Example: COCOMO (by Boehm)
    - A\*Size^b \* SUM of EffortMultipliers
    - Adaptability: can also be used in very early project stages – just leave out unknown information -> use less cost drivers
  + Putnam Model, SEER-SEM (complex), regression trees
* Non-Algorithmic Estimation Models
  + Assumed to be “more intuitive” -> used more often
  + Expert judgement (can vary significantly) -> with Delphi method  
    🡪 UNFIT for eco-cost estimation, not enough expert knowledge there yet(?!) + very complex, hard to grasp simply by “judging”
* Basics: Size and Complexity Estimates
  + Size:
    - Often: source code lines… :/
    - Or: functional characteristics: Function Points (FP) 🡪 “counts system features visible to the end user” 🡪 most common model: IFPUG
    - Many improved versions / alternatives of FP based estimation models
  + Basics for SECoMo, too!

Performance Estimations for Software Engineering

* Part of: Software QUALITY estimation (e.g. sustainability is NOT a part of it!!!)
* Performance – also takes HW utilization into account – thus RELEVANT for ECO-COST ESTIMATION!!
* SPE – software performance engineering methodology (Smith, 90)
  + Sw execution model 🡪 required resources usage
  + Combined with hw specifications 🡪 system execution model
  + Thus describe overall system performance
* UML
  + often used to describe / depict system architecture / software (+ the sw execution model?!)
  + UML profile – scheduling, performance, time 🡪 provides methods to model logical & physical hw models
* Process Algebras
  + Consider time as a very important aspect(?!)
  + Examples: Performance Evaluation Process Algebra (PEPA), …
* Palladio
  + Model / Evaluate Software Architectures regarding quality aspects
  + Component Model (PCM) – a DSL that helps to describe components, infrastructure, component allocation, etc. and for example their resource usage (just example model or part of it?)  
    + Behaviour model (as user behavior is an important aspect for performance?!)
  + Analysis Framework – can provide (early) estimations based on models (of component model?!) as input!
    - Software (Eclipsebased) Available: Graphical editor for models (PCM)
    - -> derive performance metrics with provided tools
    - Use / enhance of existing prediction models (Petri nets, Markov models, …)
  + Development Process
* 🡪 Some approaches adopted for the SECoMo method

Energy / Power Estimations

* Energy consumption = eco costs?! Well, yes, basically it all comes down to the power consumption as the main factor
* Estimation methods / metrics so far – focus on LATER development stages – SO SECoMO is the first to provide support for the earliest stages! – BUT the existing approaches can support / help
* Estimation technique for different usecases
  + Application during use – often based on OS provided functionality
    - powerAPI
    - jouleUnit
  + Energy consumption of whole systems
    - For VM’s: joulemeter (with power models)
  + Energy consumption of pervasive Java applications
    - Seo et. Al 08a/09a
    - For different architectural designs
    - Costs: computational, communication & infrastructure overhead (e.g. garbage collection)
    - Complexity / effect of HW: 3 different ways
      * Profile once
      * Multiple regression analysis
      * Symbolic execution(?)
    - Profiling: count bytecodes, native methods, monitoring operations of JVM **+** platform dependent energy model(!!) 🡪 COMPUTE TOTAL ENERGY COSTS
  + eLens (mobile applications…)
* many exist that work ON A HIGH LEVEL OF DETAIL – but for specific domains and with such high details – not suitable for Goals / the secomo approach
* Last option: using energy measurement equipment - expensive

## KOBRA – the shopping cart example specification

* 3 views that are modeled
* Structural view
* Functional view
* Behavioural view

# Core Concepts of SECoMo

Concepts -> support research goals(?!)

**Eco-Costs**

* Def: all factors influencing the ecological FOOTPRINT
* Covers: Software / in all phases: usage, disposal, distribution, …
* No unit defined, e.g. CO2, watts, EURO

Use Cases of SECoMo - When to use SECoMo? **🡪 how to integrate / practical use?!**

Measured Eco-Costs

* To express Eco-Costs that have already been measured
* To analyze an existing system, find cost DRIVERS; e.g. to rebuild it
* To MONITOR eco-costs (e.g. with SLO’s)

Required Eco-Costs 🡪 Requirement Phase

* To express eco-cost REQUIREMENTS
* To provide contraints / requirements to the developers in design/implementation phase
* Defining low-level SERVICE LEVEL OBJECTIVES in SLA’s // green specifications(?)

Estimated Eco-Costs

* To express / CALCULATE Eco-Costs that are only estimated
* To consider impacts of Eco-Costs BEFORE they occur (before software is built)
* 🡪 lead to more information / better decisions & EARLIER
* To check feasibility

## Dimensions & Eco-Cost Questions

* SECoMo should answer questions about ECO-Costs, from 4 dimensions

Viewpoint

* Different viewpoints are:
  + **data type**, e.g. eco-cost per data type -> which is the one with the highest impact?  
    (good for data intense applications)
  + **operation**, e.g. eco-cost per operation   
    (good for calculation intense applications)
  + **component**/SW based system – abstraction level, eco-costs per component (based on eco-costs of its operations) + MORE FREEDOM for the developers (in how to implement certain things, compared to data type / operation level)
  + **functional type** – abstraction level, operations are grouped by functional type (which resources are used? Eg. Gui operation, algorithmic operation) (other types possible, e.g. for a smartphone)  
    (good to optimize HW infrastructure / allocation of resources)
  + **concrete scenario** – (BASED ON “useful work done” measure by Johann et al.) e.g. eco-costs for well-defined scenarios on a certain level of detail – e.g. for a “check-out” (grouping certain operations)  
    (good for fine-grained optimization)
* To understand the cause of eco-costs -> find options for optimization!

Impact Level

* Main impact levels:
  + Session – eco-costs per session by a particular user (average or special) 🡪 find out which operations have the highest impact, consider frequency of use! + inform the user (after a “session”), increase awareness – give hints how to lower it
  + Instance – eco-costs of a SINGLE USAGE, e.g. single usage of a data type or single execution of an operation; 🡪 good if eco-costs should be minimized in general & user behavior is not known; + act as input for estimation of eco-costs
  + Sequence (only relevant for SCENARIO viewpoint) – precise order of executions & inputs is given – very precise definition of ecological costs possible (Execution paths)

Case

* WORST, AVERAGE, BEST Case to understand eco-cost in extreme situations
* Worst Case – highest possible eco-cost values of cost drivers; necessary in order to be prepared, see if the power systems have enough capacity
* Average Case – the most likely / expected eco-cost values; necessary in order to communicate general information about eco-costs, e.g. used in marketing OR to compare eco-costs of a sw system over time (improvement?)
* Best Case – lowest possible eco-cost values of cost drivers; necessary to get a realistic picture of the eco-costs

Level of Detail – *especially important for ESTIMATING eco-costs!*

* Idea of SECoMo – be applicable VERY EARLY, but then only initial ideas exist – high uncertainty // high level of detail, abstraction -> BUT changes are cheap!  
  🡪 Eco-cost estimations have a low-level accuracy (CONE OF UNCERTAINTY, Boe81)
* Additional idea of SECoMo – be flexible, allow more details & thus more accuracy to be integrated over the course of the dev. Stages!  
  🡪 support distinct level of details (like COCOMO), describe how much information & how precise hey need to be + level of results
* EARLY – basic questions on a high level of abstraction
  + Most common aspects are regarded
* INTERMEDIATE – ask finer grained questions
  + consider “functional type viewpoint”
  + consider more impact levels (for data types): CRUD
* ADVANCED – all questions, and other more accurately
  + Include flexibility statements(?)
  + Include certain scenarios

## Eco-Cost Drivers

Cost Drivers = abstraction of REAL WORLD ELEMENTS that influence the costs (eco-costs here)

🡪 “provide a means to describe the circumstance under which certain costs appear in a unified and easy to use way”

e.g. COCOMO: 4 groups, 17 cost drivers  
 - Product Factors (e.g. Software reliability, complexity)  
 - Platform Factors  
 - Project Factors  
 - Personel Factors (skill set, experience with technology)

NECESSARY to describe the circumstances under which eco-costs appear 🡪 thus define unified cost-drivers!

**SECOMO: 4 groups, 15 cost drivers: Data, Behavior, Environment, Functionality**

Cost drivers influence …

* … Resource usage of PHYSICAL hardware
* … GHG emissions ASSOCIATED with power consumption
* *🡪 proportionally! The more input, the higher the resources usage*

Why…

* Data? – the more input, the higher the resources usage
* Behavior? – certain (e.g. user) behavior has a significant influence on how operations are used
* Environment? – aspects like the hardware and the energy sources make an important difference regarding ecological costs, depending on how “clean”/”efficient” they are (e.g. cloud vs. local, old hw, etc.)
* Functionality? – different functionalities used different resources and are thus important drivers that influence the costs

### Data

Focus: Storing & Processing Information – especially in terms of BIG DATA

**Data Usage** – 4 main actions

Creating data – creating, storing (in memory, persistently, …) VIRTUAL REPRESENTATIONS

* Different resource usage of INMEMORY vs. PERSISTENT
* Persistent storage REQURIES inmemory ‘creation’ first always
* In general: eco-costs for storing persistently higher than for storing in memory  
  BUT -> in memory ‘creation’ happens way more often, so frequency needs to be considered!

Reading data – from persistent OR in memory storage

* INCLUDING the creation of in-memory representation
* Thus including the eco-costs for this “creation” in the reading eco-costs

Updating data – reading, changing, writing data in persistent or in memory storage

* INCLUDING writing the new values (so in memory / persistent creation of new representation)
* Includes all the eco-costs of these steps

Deleting data – removing data permanently, in memory or persistent

* Deleting persistent representations is more “resource intense”
* But again: frequency – in-memory happens more often, thus could in the end have higher eco-costs

**Data Type and Data Size**

The larger the data – the more resources are needed for the USAGE -> the higher the eco-costs

Size ESTIMATIONS 🡪 (as in early stages platform dependent infos might not be available)

* ONLY IF exact numbers are not available, then use this as reference!
* Based on Data Value Points (DVP) (platform independent measures)
* Puts primitive data types into relation, which are the basis for complex data types
* OWN (?!) creation of DVP’s (p. 81/82), 1 DVP ~ 1 Byte
  + E.g. Byte = 1 DVP
  + Integer = 4 DVP
  + Float = 4 DVP
  + Char = 2 DVP
  + (String = 3 DVP – but often based on encoding)
  + Boolean = 0,125 DVP, etc.

**Data Flexibility**

🡪 means the degree of freedom with which data can be used (leeway = Spielraum)

* Time or Resource dependent
* E.g. only perform certain data operation when renewable energy is available
* Limit certain characteristics of data types etc. when no renewable energy is available

### Behavior

HOW is the system being used? – consider FREQUENCY

**User**

* Consider different user types (that have different usage patterns & flexibilities)
* So based on those user types eco-costs can vary

**Pattern**

* Or: usage profile, session data
* Captures “order and frequency of the usage of the operations” – typically for a certain user group
* For WHOLE SYSTEM or for single components, or broken down to certain use cases
* 🡪 express Frequency of usage!

**Flexibility**

* Time or resource dependent; the leeway for reducing more ecological costs
* E.g. leave out certain un-eco-friendly behavior (e.g. checking the shopping cart frequently, i.e. switching operations often and using resource intense operations) to reduce eco-costs

### Environment

**Hardware**

* Hardware infrastructure has a major impact! … but leads to energy consumption (?!)
* Different levels of abstraction possible: CPU properties, Server properties, DATA CENTER properties! 🡪 abstraction level dependent of SECoMo scenario + how much overhead should it be?!
  + Typically on level of computers / servers (good tradeoff)
  + But could also be drilled down if certain technical components could e.g. be exchanged
* Consider for Eco-Costs:
  + Type of component and its individual impacts on eco-costs
  + Frequency and INTENSITY of use of these components
* Resource Factors
  + Eco-costs caused by A SPECIFIC HW RESOURCE (amount of energy used by a hw component)
  + To reach a certain goal (e.g. data type creation, operation execution, …)
  + 🡪 define for one resource like the server OR multiple resources like CPU, memory, HDD

**Energy (Source)**

* Clean energy sources (wind, water) preferred over e.g. coal power plants
* Issue: renewable energy sources NOT ALWAYS AVAILABLE / production fluctuates 🡪 Storage necessary (but that is still not sufficient, high losses)
* GOAL: use as much energy as possible DIRECTLY when it is produced..
* SO, the user would need to be VERY flexible

**Flexibility (of the environment)**

* Freedom in terms of environment aspects / surrounding conditions that enable leeway for eco-cost optimization
* Time or resource dependent
* Options: e.g. virtualization technologies to distribute workload dynamically
* Or SLA related: availability of a system can vary in order to improve eco-costs

### Functionality

…covers all executable aspects of the system, including data operations

**Input/Output/Change**

Input = data required for operation execution

Output = results data that is returned by an operation

Changes = state changes in visible(?) fields of the data

* Size of these influence RESOURCE USAGE and RUNNING TIME of operation -> thus relevant for eco-costs

**Complexity**

Usually used to describe performance of algorithms; running time or growth factor of memory size…

* Big O notation (worst(?) case scenario) is COMMON, but not fitting for SECoMo -> as high loss in detail and accuracy of circumstance description
* More precise description needed:   
  Complexity = relation between input and RESOURCE USAGE (NO data actions!!!!!)
* In SPECIFITCATION phase: hard to estimate the complexity, as it is not clear yet HOW software is implemented.. (and there could be several ways, e.g. sorting)  
  🡪 SO estimations of complexities (e.g. by experts) are needed!

**Usage**

Use of functionality = FREQUENCY of executing an operation within a SESSION of a certain USER TYPE

🡪 Always consider eco-cost per operation in combination with frequency to have a reasonable understanding

(part of the user profile!?)

**Flexibility**

… any degree of freedom related to operation executions (functionality) that increases leeway for optimizing eco-costs

* Meaning: how can the functionality / operation execution be adapted to improve eco-costs?
* E.g. only allow certain (resource intense) operations when e.g. solar energy is available

**Functional Type**

(s. oben??) use functional types based on resource factors to connect functionality with certain hardware resources - and which of them are used influences the eco-costs   
(group operations by functional type, i.e. which hw resources are used?)

# Eco-Cost metrics 🡪 how to relate with existing metrics?

**11 Metrics to express and compare ecological costs, mapped to viewpoints & level of detail**

Goal: new metrics for eco-costs tailored to software specifications! 🡪 help to answer the questions (figure 6.1)

Metrics: NO UNIT -> thus can be used in different categories, e.g. for measuring energy (Joule) or pollution (CO2)

**Primary Metrics** – information that is relevant to ALL stakeholders, most relevant

**Secondary Metrics** – enhance primary metrics, interesting to some stakeholders(?), nice-to-have

**Input Metrics** – just used as input for calculating other metrics (improve estimates)

## Metrics concerning Data

Data = important cost-driver for software (e.g. space, energy)

(other metrics… p. 93)

**EARLY LEVEL**

* ECDUO, secondary (Eco-Costs of Data that is Used Once)
  + for CRUD = “usage”, instance level
  + Inputs: d = data type, c = case, I = infrastructure, u = user
* ECDUS, primary (Eco-Costs of Data that is Used in a Session)
  + Also CRUD operation, but SESSION level
  + Inputs: d = data type, c = case, I = infrastructure, u = user

**(Only) Intermediate & Advanced LEVEL**

More precise description of CAUSE of eco-costs, as more details available about different data types

ALL = INPUT METRICS (help to more precisely calculate ECDUO, ECDUS?)

* ECDC, input (Eco-Costs of Data that is Created once)
  + Persistent/in-memory CREATIOn of ONE instance of a certain data type
  + Inputs: d = data type, c = case, t = time, i = infrastructure, u = user, f = flexibility of user type
* ECDR, input (Eco-Costs of Data that is Read once)
  + Persistent/in-memory reading of ONE instance of a data type
  + Inputs: d = data type, c = case, t = time, i = infrastructure, u = user, f = flexibility of user type
* ECDU, input (Eco-Costs of Data that is Updated once)
  + Persistent / in-memory update of ONE instance of a data type
  + Inputs: d = data type, c = case, t = time, i = infrastructure, u = user, f = flexibility of user type
* ECDD, input (Eco-Costs of Data that is Deleted once)
  + Persistent/in-memory deletion of ONE instance of a certain data type
  + Inputs: d = data type, c = case, t = time, i = infrastructure, u = user, f = flexibility of user type

+ improved primary / secondary metrics, for more specific questions now

* ECDUO
  + Defined as before, but two more inputs: t = time, f = flexibility of user type
* ECDUS
  + Defined as before, but two more inputs: t = time, f = flexibility of user type

## Metrics concerning Operations

Functionality = Create, use, delete data, but also in terms of goals that need to be fulfilled -> drive energy consumption & eco-costs!!

(Other metrics… p. 97)

All metrics for operations are applicable on all levels of detail!

* ECOS, primary (Eco-Costs of the use of an Operation within a Session)
  + How much eco-costs are caused by a certain operation within one session?
  + Input (EARLY LEVEL): O = operation, c = case, I = infrastructure, u = user
  + Input (Advanced(?)): O = operation, c = case, t = time, I = infrastructure, u = user, f = flexibility of user type
  + User type is considered to be the Average user
* ECOO, secondary (Eco-Costs of the use of an Operation that is used Once)
  + How much eco-costs are caused if the operation is executed one time
  + (input???)

## Metrics concerning Components and Functional Types

Functionality clustered -> per component OR per functional type (type of hw resources used!)  
- one metric for each!

(other metrics… but not named… p. 99)

**For all levels of detail**

* ECC, primary (Eco-Costs of a Component)
  + The eco-costs that are caused by one component of a sw system
  + Input: ct = component, c = case, t = time, I = infrastructure, u = user, f = flexibility of user type
  + But time & flexibility only relevant for intermediate / advanced level!

**Intermediate & Advanced Level of Detail**

* ECFT, secondary (Eco-Costs of a Functional Type)
  + The eco-costs that are caused by a certain functional type (so a certain set of hw resources used)
  + Inputs: ft = functional type, c = case, I = infrastructure, u = user  
    (what about time & flexibility???)

## Metric for Specific Scenarios 🡪 could be interesting to integrate with existing models?

Usage scenario HIGHLY influences eco-costs!!

(other metrics.. e.g. time interval or per-execution… p. 100)

Wdh: scenario definition based on “work done” metric

ONLY for ADVANCED level of detail!!

* ECS, secondary (Eco-Costs of a Scenario)
  + Eco-costs of a certain specific scenario that has a fixed sequence of operations / inputs
  + Input: wd = concreted scenario as sequence of operations/inputs, t = time, I = infrastructure  
    (what about the rest of the parameters?!)

# Auxiliary Models

🡪 models that help to calculate metrics FROM SPECIFICATION MODELS + to communicate them to stakeholders

🡪 complete source of information for the calculation of the metrics (?!)

🡪 based on original KobrA models -> enhance / annotate -> sometimes multiple models for different circumstances are needed (e.g. different user type)

Kobra - (but also similar approaches possible)

Option: derive more simple models for different stakeholders

* Auxiliary models are not absolutely necessary, but HELPFUL
* Like Green Specifications – compact way to communicate eco-costs (allowed?)
* Help to understand decisions / trade-offs
  + (e.g. Reu16 – metamodel for architecture)

Best Practice: Create separate models for different sets of characteristics (input values)

* E.g. one for (worst, customer, no (flexibility), 1p.m., DC1)
* … and one for (best, administrator, no, 1p.m., DC1)

## Operations

1 viewpoint: Operation – calculate / estimate eco-costs per operation

2 eco-cost metrics (ECOS, ECOO), 5 cost-drivers () (s. functionality)

🡪 basis in KobrA: Functional View (operation specifications)

* Other two views are important as they describe eco-costs & circumstances

**EARLY LEVEL OF DETAIL**

(not relevant – flexibility & functional type cost drivers)

* Add ECOS & ECOO to Kobra Operation Specification (AM 7.1)
  + Worst Case values – as upper bound  
    e.g. <= 340.20 ug Co2
* Provide description of cost drivers to explain circumstances for eco-costs (AM 7.2 – 7.4)
  + **Inputs / Outputs / Changes** – the data influences the eco-costs significantly, already visible in early phase
    - In Kobra: specified in Receives (input), Returns (output) and Changes clauses!
    - What is missing?
      * HOW is data stored? In-Memory or persistent
      * HOW BIG is the input? (size?)
    - 🡪 Add this information to the clauses
      * (<no. of persist. Data usages>, <no. of in-memory data usage>)
      * [<size of the data>], for input – before operation; for output – after operation
  + **Complexity** – (relation btw. Input & resource usage? – no data actions)
    - ADD complexity statement to operation specification (AM 7.3)
      * (very accurate!!) ~ 1 + numberOfProductsInCartBeforeExecution +1
      * 🡪 variable defined by values on input/output/ etc.
    - ??? wie kommt der zustande ???
  + **Usage** – relecant for ECOS -> session impact based metric!
    - 🡪 NEW AUXILIARY MODEL – based on behavioural model (state chart diagram)
    - Relevant: know how the operation is acutally USED in a session
    - 🡪 create best, average and worst case paths (**based on domain expert knowledge**), e.g. best case = shortest paths
    - Prerequisite: know path of current case -> count times of operation calls
    - ADD Frequency (of use during session) to operation specification (AM 7.4)

**INTERMEDIATE LEVEL OF DETAILS**

Eco-Costs are the same on this level, thus expressed the same way (just more detailed maybe?) (= metrics??)

(Most) Eco-cost drivers are extended though 🡪 more details, differences, more drivers

* **Input / Outputs / Changes** – but now more detailed
  + In-memory or persistent, which action? CRUD   
    🡪 8 new elements (first 4x persistent, then 4x in-memory)
  + ADD these details + size to RECEIVE, RETURN, CHANGE clauses (AM 7.5)
* **Functional Type**  - NEW! ~~(grouped by hw resource usage)~~
  + ADD some information about functional type to operation specification (AM 7.6)
  + E.g. “In-Memory Data”, “algorithmic” ???
* **Usage** – now more information from auxiliary model -> more precise information, not domain expert knowledge needed
  + Basis again: state chart diagram (KobrA)
  + NEW: annotated with additional information about circumstances (e.g. max. number of calls / loops; max. number of single path usages)
  + Best case = shortest path; WORST CASE = now see annotations! -> find longest path;
  + Average case -> use annotations in state chart diagram (this time: transition probabilities) 🡪 DERIVE FREQUENCY OF USE!!! By using a markov chain..
  + HOW TO DO THIS? 🡪 p. 113 – 117 – wdh!!!!!!!!
  + Add Frequency to operation specification…

**ADVANCED LEVEL OF DETAILS**

Same Eco-Cost METRICS, ONE ADDITIONAL eco-cost DRIVER though! + other redefined

+ new: add uncertainty to the drivers / calculations 🡪 create a probabilistic range

* **ECOS / ECOO** – add uncertainty
  + Add +-e as a factor – with e = STANDARD DEVIATION! (see AM 7.7)
  + (Zahlen????)
  + ONLY IMPORTANT FOR AVERAGE CASE
* **Inputs / Outputs / Changes & Complexity**
  + Also enhanced by a standard deviation to express uncertainty (AM 7.8)
  + For COMPLEXITY – define it for different HW resources, e.g. CPU, RAM, etc. (instead of a DC level) (AM 7.9)
* **Flexibility** – NEW: can be expressed in very different ways
  + …and in relation to many other drivers, e.g. functional type or usage and so on (e.g.)
  + … how to express???
* **Usage** – AGAIN: refine the auxiliary model – state chart diagram!
  + Best Case = again, take the shortest path!
  + For others: improve annotations!
  + Transition probabilities before were STATISTICAL (???) -> only rough
  + Now: flexible transition probabilities to allow them to change based on previous steps 🡪
    - Using formulas that express different transition probabilities based on certain factors
    - E.g. -> the probability of removing products becomes higher if more products are in the shopping cart (etc.)
  + Again: derive frequency of use via a markov chain  
    🡪 DYNAMIC markov chain into STATIC markov chain 🡪 derive frequency! (procedure identical to p. 113 – 117) … see p. 124..)   
    WITH uncertainty & standard derivation..
  + ADD standard derivation to operation specification – frequency (AM 7.10)

## Data

1 viewpoint, 6 Metrics, 4 cost drivers

Based on: KobrA Structural View – class diagram (but others also relevant for session metric)

**EARLY LEVEL OF DETAIL**

2 metrics for the early level – type, size and usage are the relevant eco-cost drivers

Annotating Structural View (data types) with details about the metrics (Figure 7.9)

* **ECDUO** – used once – give worst case upper bound
* **ECDOS -**  used in a session – give worst case upper bound
* But also: exact value or lower bound (for average or best case)
* “compostition” relationship = eco-costs of one data type are part of the eco-costs of the other data type! -> provide MULTIPLICITY VALUE (as comment) – which range can the multiplicity (e.g. \* ) have? 10, 50, 1000?
* **Data Type and Size** – two very fundamental eco-cost drivers
  + Data types (of attributes) are naturally included in the structural view (e.g. id: int)
  + + COMMENTS for annotating data types with SIZE
* **Usage** – important for session based usage costs
  + Use information from OPERATION SPECIFICATION (receive, return, change)
  + A) 🡪 is the data type used in the operation?  
    B) 🡪 how often persistent vs. in-memory?  
    + information: how frequently is the operation used?
  + Add up numbers for all operations that include the data type – how often used in-memory / persistently?
  + How to “express” ????

**INTERMEDIATE LEVEL OF DETAIL**

4 additional metrics -> describe data USAGE in more detail; eco-cost drivers remain unchanged

* **Eco-Costs of Data** – ECC, ECR, ECU, ECD – all annotated to Structural View with upper bound (in worst case) (etc.) (Figure 7.11)
* **Data Type and Size**  - expressed in the same way
* **Data usage** – expressed as before(???) but now subdivided into creation, read, update and delete

**ADVANCED LEVEL OF DETAIL**

Same metrics and drivers (+ flexibility), but more detailed information exists -> refine metrics

🡪 NOW: ADD UNCERTAINTY value

* **Eco-Costs of Data** – add standard deviation to ECDUO / ECDUS in Structural View (Figure 7.12)
* + Add uncertainty to MULTIPLICITY values…
* **Data Type and Size** – add uncertainty / standard deviation to data size annotation (Figure 7.13)
* **Flexibility** – expressed by creating MULTIPLE AUXILIARY MODELS, one for each “characteristic” that is flexible (e.g. flexibility about data size – low quality during no-clean energy times OR data types – bytes instead of integer)
* **Usage** – expressed in the same way, but added with UNCERTAINTY VALUES

## Components and Functional Types

One Eco-Cost metric per viewpoint

No new eco-cost drivers, as they are composed of the costs of the operations!!

* **ECC**, all levels of details – eco-costs of component per session
* **ECFT**, only intermediate & advanced level, for different functional types!
* 🡪 Added as Annotation to the Structural View to the <<SUBJECT>> Class!!! (Figure 7.14)
* Enhanced with uncertainty (advanced level)

## Scenarios

One metric, at advanced level of detail

Can describe Eco-Costs on a very precise level

* **ECS**, for one specific scenario
  + NO average values – precise values, worked out in detail!
  + Prerequisite: define all eco-costs and cost-driver factors specifically for each case!
  + Specify clearly: order of operations, input and output, concrete system states in between

Estimating Eco-Costs with SECoMo

# 8. Basic Concepts of SECoMo’s Estimation Technique

**Scope**

SECoMo offers..

* Answering questions about Eco-Costs in an early stage
* Answering questions from many different dimensions
* An algorithmic estimation model (based on mathematical models)
* For 3 different levels of detail
  + EARLY – Inception phase, or general when “small” overhead of estimation is necessary; less details available (thus not as accurate estimations) BUT -> easier / cheaper to make changes
  + INTERMEDIATE – More details available, thus less tradeoff between overhead & accuracy -> more accuracy possible, more precise questions can be answered
  + ADVANCED – “cost of creating estimates is not an issue” (?) -> this level leads to most accurate estimates; highest detail level of input!  
    OR even describe precise scenarios
* Levels of details can be mixed (Figure 8.1 – Homogenious models vs. Heterogeneous models, + Chapter 10/11) 🡪 thus it is possible to always achieve highest level of accuracy with the given information!
* cost-drivers = INPUT to the models!
* Auxiliary Models = input to CREATE estimates (with the details about cost-drivers they contain?) + REPRESENT estimates (as they are included?)
* Metrics = express the different eco-costs!

**Model Calibration -> schwamming, was genau passiert hier & wie?**

Why? – because some parameters are platform dependent 🡪 calibrate MATHEMATICAL models for each environment

* Involved aspects: cost-drivers for ENVIRONMENT & related parameters
  + Resource factors
  + Energy mapping
* **Callibration Set** – set of typical operations and data actions
  + Usually: use a calibration system that uses the same or similar sw components
  + (as it is shown: implementation / programming languages / how sth. Is programmed has an influence on eco-costs)
  + 🡪 thus take many implementation specific aspects implicitly into account 🡪 what about: more detailed analysis of eco-efficiency of style of programming??
* Callibration Set should **be representative** (e.g. if usually more data is read than deleted – this should be in the CS too)
* Units of resource factors = units of Eco-Costs 🡪 the calibration process should be performed accordingly
  + E.g. CO2 -> take ALL Co2 emissions into account, direct AND indirect (e.g. by hw production, disposal, etc.)  
    🡪 this is hard? HOW? Is semcomo lacking sth here??
  + E.g. Watt-Seconds 🡪 only real energy consumption considered
* Separate callibrations for each scenario necessary (as it differes on certain levels, times, cases, flexibilities …)
* **SEE PART IV**

# 9. Early Estimation of Eco-Costs

-> to support this EARLY level estimation is one of the key / core goals of secomo!!

-> to help make decisions early when costs of changes are still low

-> (can be) based on minimal information

## Estimating Eco-Costs of Data

How to calculate ECDUO & ECDUS? (Fehler: “ECDOS”) 🡪 DATA is a major cost driver!

Always also dependent on input parameter

**ECDUO**

* Define inputs
* Size of ecduo – dependent on 2 cost drivers: DATA (Size & Multiplicity) & ENVIRONMENT (resource factors: persistent & inmemory data usage)
  + !! how to get the resource factors? -> calibration process? :S   
    -> and based on input parameter for HW infrastructure
* ECDUO = Hardware mapping \* size of attributes
  + HW mapping = ½ \* (resource factor of P + resource factor of IM) -> durchschnitts resource factor?
  + Size = SUM of (size of attributes \* multiplicities of each)
* Example: p. 145, using auxiliary model STRUCTURAL MODEL (Class diagram)

**ECDUS**

* Now: consider USER behavior ! -> input / output / changes values of how often data type is used iM or P + frequency of OPERATIONS (in which data type is used?!)
* Same Cost drivers Data & Environment + FUNCTIONALITY -> derive how often data types are used in an operation per session -> sum this up for all overations = no. of overall usages of data type in a session
* ECDUS = Hardware Mapping \* Size of attributes
  + HW mapping = Overall frequency of persistent usage (sumof all of operations..)\* Resource factor P + Overall frequency of inmemory usage \* resource factor IM
  + Size = (s. oben – same)
* Example: p. 147, using auxiliary models STRUCTURAL MODEL & (functional model?) OPERATION specification
* DIFFERENCE to ECDUO: Multiplication factor for the resource factors – now taking frequency of use into account

## Estimating Eco-Costs of OPERATIONS

ALL costs can be tracked back to the execution of operations -> major cost factor

Always also dependent on input parameter

**ECOO**

* Data related cost drivers cover multiple data types (all that are used), but functionality related cost driver only refers to the specific operation;  
  Functionality Category -> Input/Output/Changes + COMPLEXITY (NOT session – here: only once)  
  Environment Category -> inmemory / persistent data usage + NON-CRUD part (???)
* REUSING other metrics: ECDUO’s of the used data types! -> combination of data types, order important!
* (Figure 9.4 falsch?? Wo is functionality cost driver??)
* ECOO = none-CRUD related part + CRUD related part
  + None-CRUD: CRUD-independent complexity \* resource factor (of operation)
  + CRUD-related: ECDUO(of each data) \* frequency of use (of data type) -> summed up

**ECOS**

* … eco-costs of operation for A SESSION! 🡪 include frequency / usage!
* SIMPLE! -> ECOS = ECOO \* frequency of use of an operation!

## Estimating Eco-Costs of Components

* Single components, or the whole sw system (as component)
* Based on a predefined “session”
* Helpful to communicate “more general information” to stakeholders
* ECC = SUM of the ECOS of all operations

# 10. Estimating Eco-Costs on the Intermediate Level of Detail

- in the course of a project – more information become available 🡪 possibility to improve metrics

- INTERMEDIATE level: “provide a different balance between estimate accuracy and overhead”

## Estimating Eco-Costs of Data

* 4 additional metrics (about the CRUD aspects of data)

**ECDUO**

* Difference: use more resource factors (data creation, reading, updating, deletion on persistent or inmemory level) + consider DATA VALUE POINT to more specifically describe impact of data type
* AGAIN: derive resource factors from calibration process (HOW?)
* ECDUO = Hardware mapping \* size of attributes
  + HW mapping = 1/8 \* (resource factor of P + resource factor of IM – of each: creation, reading, updating, deleting)
  + Size = SUM of (size of attributes \* multiplicities of each \* data value point)

**ECDUS**

* Difference: (??) functionality cost driver category (existed on early level, too?!); data usage now divided into 8 categories: P & IM for CRUD
* ECDUS = Hardware Mapping \* Size of attributes
  + HW mapping = (Overall frequency of persistent usage (sumof all of operations..)\* Resource factor P + Overall frequency of inmemory usage \* resource factor IM) FOR EACH CRUD OPERATION
  + Size = (s. oben – same: with data value point)

**ECDC, ECDR, ECDU, ECDD**

* (only input metrics for other primary/secondary metrics!)
* But can also be used for “in-depth” analysis (??)
* Similar cost drivers like ECDUO, but of course – only 2 different resource factors (in-memory vs. persistent)
* ECD\* = Hardware Mapping \* Size of Attributes
  + Hardware mapping = ½ \* (resource factor <CRUD-action> for persistent use + for in-memory use)
  + Size of Attributes = sum of (multiplicy \* size \* data value point – of attributes)

## Estimating Eco-Costs of Operations

* Same metrics as early level – ECOO & ECOS
* 🡪 more details available – more precise estimations

**ECOO**

* Wdh: eco cost drivers – data, functionality, environment
* New: 8 resource factors + Data Value Point (DVP) included
* AND: functional aspect: FUNCTIONAL TYPE as driver (FT = map HW resource to operations)
* (p. 162) ECOO: non-CRUD-related complexity + CRUD-related costs of all CRUD actions in the operations
  + Non-CRUD: complexity \* resource factors that relates to the functional type
  + CRUD-related: ECDUC/ECDR/ECDU/ECDD \* frequency of CRUD-action
* Mathematical formula is the same as at early level – but results can change, become lower or higher due to different calculation of input metrics

**ECOS**

* Same function as at early level
* Data & Environment cost drivers the same, Functionality: Input/Output/changes, Complexity
  + + USAGE: provides session impact (frequency), - derived from usage pattern: current operation, subsequent operations, probabilities of them being executed…
  + + FUNCTIONAL TYPE
* Based on average user! (weighting the usage pattern according to number of users)
* (p. 165) ECOS: 1/k \* Sum of all frequencies of operation per user type \* ECOO
  + K = number of different user types

## Estimating Eco-Costs of Components

**ECC**

* Same calculation, but more precise input data = more precise estimation
* ECC = sum of all ECOS (per operation) of the component (p. 165)

## Estimating Eco-Costs of Functional Types

NEW METRIC – ECFT, helps to identify source of eco-costs in terms of hardware

Idea: calculate similar to ECC, BUT – include only operations that have the same functional type

**ECFT**

* ECFT = sum of all ECOS (per operation with specific FT)
* Same cost drivers as ECC

## In between the Early and Intermediate Level of Detail

How can parts of the model elements be interchanges in order to support more precise estimates when only part of the information becomes available (early)?

🡪 calculate metrics in between the levels

🡪 support the adaptability of the metrics and possibility to always calculate /estimate the most precise metrics

* ECDUO: Exchange Hardware mapping & size
  + if the information is available, HW mapping from intermediate level can be used, otherwise the early one is used
  + MIX size calculation: if available, use intermediate level calculation (with DVP), otherwise without
* ECDUS: Exchange HW mapping & size
  + HW mapping: if available, calculate intermediate level details, otherwise use early level one
  + Size: mix calculation if more detailed information is already available
* ECDC/ECDR/ECDU/ECDD: estimate at PRE-intermediate level, with partwise information
  + Calculate already when “not all attribute data types are available” (?)
  + Size: mix calculation if more details are already available (? Not according to text?)
* ECOO: Exchange CRUD-related parts and non-CRUD-related parts  
  WHAT ABOUT ECOS??
  + If available, include information about FUNCTIONAL TYPE, if not, leave it out (non-CRUD part)
  + CRUD-related (data) part: mix information if more details are available – either the ECDC, … metrics from intermediate level OR only the early level ECOO(??)
* ECC: same model for both levels, so no “real” change possible
  + BUT possible to use ECOS from different levels!! (so exchange input parameters)
* ECFT: (pre-intermediate level of detail possible)
  + Use ECOS values that were estimated in the EARLY level

# 11. Estimating Eco-Costs on the Advanced Level

Intermediate 🡪 Advanced Level

Additions: adding notion of Uncertainty (Values -> with Standard Deviation), Time, Flexibility

## Estimating Eco-Costs of Data

**ECDUO**

* Eco-Costs contain TIME & FLEXIBILITY elements
* New Cost driver: Environment – Energy – provide time dependent CO2 values
* COMPLEXHWMODEL is now used – includes multiple resource factors, to describe resource factors in more detail (e.g. CPU, memory, HDD instead of only server)
* ECDUO =
  + x/8 \* (HW mapping +- Sum of all uncertainties of resource factors (e))
  + Size, Multiplicity, DVP +- uncertainty factor (e) EACH
* Uncertainty e = SD (standard deviation) \* factor of confidence interval (e.g. 3)
* INCLUDE CO2 factor – X/8
* Remember: each resource factor is / can be composed of resource factors (of same kind) (p. 173)
* Application of equation with uncertainties: p. 174

**ECDUS**

* New: complex hardware models, CO2 value, uncertainty, time, flexibility
* Add uncertainty e to frequencies & resource factors, multiplicities, sizes, DVPs
* Calculation p. 175/176

**ECDC, ECDR, ECDU, ECDD**

* Similar to ECDUO, but only focusses on two resource factors each (e.g. in-memory & persistent data CREATION etc.)
* As before: uncertainty added + time & flexibility
* Metrics -> p. 177/178

## Estimating Eco-Costs of Operations

**ECOO**

* Again: extended by TIME & FLEXIBILITY; + ENERGY cost driver, + ComplexHWModel
* ECOO = Energy part \* non-CRUD-related part + CRUD related part
  + Non-CRUD related part: can include multiple complexity values
* Formula is basically the same as on the intermediate level, but input values are more specific

**ECOS**

* Wdh: session based on frequency of use of operation – based on usage pattern of user type
* + complex hw model to describe lower-level infrastructure elements
* ECOS = ECOO \* frequency of use for average user (+ uncertainty value)
* Formula is the same for all cases, user types, etc -> only INPUT VALUES change

## Estimating Eco-Costs of Components and Functional Types

* ECC & ECFT
* Both metrics: group eco-costs of operations – per component or per same functional type
* Cost-Drivers like ECOS, but of course for multiple operations (so cardinality to functionality is >1)
* Formulas: p. 184

## Estimating Eco-Costs of SCENARIOS

* New metrics: ECS
* COST-DRIVERS similar to those of ECC
* Input values for formula depend on scenario execution order and input values
* ECS = Energy \* SUM OF (ECOO of every operation in execution order, with current state of system and individual input values – as sometimes they are dependent on previous actions)
* Use exact values for ECDC, ECDU, ECDR, ECDD, but – UNCERTAINTY still exists, e.g. in combination with the resource factors

## Mixing Levels of Detail !!!!!

ECDUO p. 187

* Advanced level metric: 3 parts (energy mapping, hw mapping, size)
* Possible to exchange the HW mapping based on information about calibration process(??) – use either one of the 3 level hw mappings  
  🡪 COULD be mixed, but usually does not make sense as calibration info is usally existent for a whole level of detail
* Possible to exchange size calculations / mix them based on the details known

ECDUS p. 188

* Possible to exchange / OR MIX size calculations
* Possible e.g. to also Exchange or MIX hardware mappings
  + E.g. when the frequency for SOME operations is given in more detail (e.g. with uncertainties) than for others

ECDC, ECDR, ECDU, ECDD p. 189

* Possible to calculate “pre-advanced” version of metrics – partially include information from advanced level – integrate it with earlier information
* E.g. mix hardware mapping
* E.g. mix size ESTIMATES for the attributes based on information given

ECOO p. 190

* CRUD-related part = can be a mix of early, intermediate & advanced levels!!
* Non-CRUD-related part = exchange early, intermediate or advanced level!!

ECOS p. 191

* Very similar to ECOO
* Exchange FREQUENCY VALUE from different levels

ECC, ECFT p. 191

* Wdh: only a sum of multiple ECOS values
* What can be mixed? Use ECOS that are calculated on different levels and mix them together

ECS (?? P. 192)

* Can be calculated in a pre-advanced form
* Based on ECOO’s & execution sequence
* Use ECOO’s from different levels or mixed ones to calculate ECS
* ???

Using SECoMo

# 12. Leveraging SECoMo in Software Engineering

Pro’s and Con’s of using SECoMo – Costs and Benefits

* Usually: saved ecological costs outweigh monetary costs, but not always
  + Costs of creating auxiliary models – overhead, but not if connected to a process which already creates the models
* When is SECoMo beneficial?
  + When ecological friendliness is an important aspect in the SW system usage environment
  + When energy consumption should be minimized
  + When local energy sources should be used – as it provides information about the actual energy consumption
  + When new HW should be purchased to minimized the overall costs of usage – with the knowledge about the resource usage, optimal hw can be purchased

Using SECoMo in Existing Software Engineering Processes

… in **Traditional Approaches** (e.g. Waterfall, **V-Model**) – FIGURE 12.2!!

* SECoMo can be used throughout the whole development cycle!!
* Requirements Engineering
  + Ecological costs = key concern for formulating requirements
  + Easier to make decisions as more information about ecological costs are available
  + Good: supports early decisions about sustainability -> less actual costs (than changes later, e.g. after implementation & then measured)
* Design Phase
  + Adaptability of model 🡪 estimations can be redefined & improved
  + Design decisions can take eco-cost estimations into considerations
  + E.G. Architecture design – possible to find a solution that is more eco-efficient
* Implementation Phase
  + 🡪 Eco-cost constraints can act as Non-Functional Requirements for development
  + Possibility to define specific or general eco-cost requirements that developers have to adhere to
  + E.g. in combination with green specifications / SLA’s
* Verification / Validation
  + Use auxiliary models to validate the SW against! Does the software adhere to the eco-cost requirements 🡪 necessary to MEASURE ecological costs precisely
  + Analyze reasons why it does not fit
    - Requirement too strict?
    - Not implemented well enough?
  + + update the ecocost models with actual values 🡪 improve accuracy of future estimates
* Maintenance
  + SECoMo 🡪 helps to analyze situations which were not foreseen
  + Update auxiliary models -> to adapt models to support future projects
  + Helps to support change situations 🡪 calculate new estimates! To find optimal solution

… in **Agile Approaches (e.g. SCRUM)**

* Mainly 🡪 early-level estimates relevant, less the formal documents!!
* Backlog Creation / Requirements Engineering
  + Must Have vs. Nice To Have 🡪 nice to haves can be regarded relating to eco-costs 🡪 remove them if too high!
  + Perform estimates for NiceToHaves
  + Optimize Must Haves for Eco-Costs
  + Add new eco-cost requirements
* Sprints
  + Keeping track of ecological costs – estimated vs. actually measured(?)
  + At the end: validate and analyze if successful, if not – reasons?
* Estimations
  + Less relevant, as rapid implementation is the focus..
  + But if not regarded 🡪 possibly wasted time -> overhead
* Goal: LOW OVERHEAD – vs. high overhead of specific estimates..
  + Prefer EARLY level estimates 🡪 (why?)
  + Documentation still important
* End of Sprint
  + Use Secomo to create new estimate / update existing ones
  + + at the very end 🡪 auxiliary models provide a very good documentation + measured ecological costs are already included

… COMBINING SECOMO WITH EXISTING SUSTAINABLE SW ENGINEERING APPROACHES!!!

* GREENSOFT -> analyze in more detail / example of how to fit?
  + SECoMo fits into the general frame of the model
  + Relates to the development / usage phase
  + Related to the effects (first, second, third-order) – how it can effect sustainability issues
    - Reducing hw and resource usage
    - Facilitate use of renewable energy
  + OFFERS NEW SUSTAINABILITY metrics
  + + adds recommendations
* Requirements Engineering
  + Integrate with Penzenstadler Approach (MP15) – sustainability goals & rules
  + Information of the requirements -> important input for SECoMo
  + Quality requirements 🡪 can use metrics of SECoMo
  + (integrate with others)
* Help for architectural / design decisions?
* Specification
  + Of data types, functions, etc.
  + Optimize specifications
* Development
  + Eco-cost requirements provided
  + Approaches (from chapter 2?)
  + Choice of development language – how does SECOMO fit with languages like eco?
  + Choice of framework (e.g. SEEDS) – to support code optimizations – other frameworks?
* Validate fulfillment of requirements
  + With existing tools, e.g. JouleUnit
  + (others?)

# 13. Evaluation -> später wichtig für Masterarbeit!?

Evaluation = how well does the method perform for the SHOPPING CART EXAMPLE?

Compare: ESTIMATED eco-costs vs. the REAL eco-costs measured during system execution

**Experimental Set-Up**

* Special test environment
* Hardware description: Processor, RAM, HDD, OS – description
* JouleUnit & JMH (???) (Pon14) -> measure costs & establish resource factors!!
  + JouleUnit using PowrProf dll (windows) -> estimate energy consumption btw 2 timestamps
  + Do CALIBRATION RUN before to remove idle state consumption
  + IF other processes energy consumption is stable -> possible to derive power consumption of application under test
  + JMH = benchmark harness (?) – avoid “common benchmarking pitfalls” 🡪 make results more reproducible
* Creation of AUXILIARY MODELS – using MS Visio, based on the KobrA method (see chapter 4)
* Eco-Cost ESTIMATES – calculated in MS Excel
* COMPARISON of measured vs. estimated costs:
  + Set up Measurement Framework
  + Create Auxiliary Models (on 3 levels of detail), based on KobrA method specification & expert knowledge
  + Calibrate Estimation Model using
    - A calibration set ???
    - The described test set-up
    - 🡪 LEADING TO THE RESOURCE FACTOR VALUES!?
  + ECO-COST ESTIMATION
    - Using resource factors
    - And relevant elements from the auxiliary models
    - + inclusion of estimated values into auxiliary models
  + Implementation of Service/Application under test (e.g. shopping cart service) (in JAVA)
  + DERIVE eco-costs from execution of Service & results from JouleUnit / JMH -> using described measurement set-up
  + COMPARE Estimated eco-costs (SECoMo) vs. Measured Eco-costs (JouleUnit/JMH)
    - + analyze reasons for possible deviations

**Evaluating SECoMo’s Estimation Approach**

* JouleUnit -> produces only “estimates” as well, but here they act as our “MEASUREMENTS”
* Scenarios
  + 1. Login, addProduct, checkout, pay, logout
  + 2. Login, addProduct, removeProduct, addProduct, addProduct, removeProduct, checkout, pay, continueToShop, addProduct, checkout, pay, logout
* Functional Types
  + ALGORITHMIC - ??
  + INMEMORY - ??
* Cases – imitate the ones from the auxiliary models???
* Table 13.1 -> which LEVEL?????????? Or is it only ONE metric that is measures (e.g. the “most advanced” value?)
* Estimates of the EARLY level - data
  + Accuracy: btw. 15 – 39 %
  + Deviation may lay in
    - General lower accuracy at this level, as many factors are not considered
    - JVM optimizes a lot
    - JouleUnit uses estimates itself
  + 🡪 Resource factor was set too high!
  + RE-Run of evaluation with adapted resource factors (for estimation??)
    - Better accuracy: -4 to 21 % , sd of 8 %
    - New factors were used for all other early level estimations
* Estimates of the EARLY level – operation
  + First iteration of evaluation – VERY HIGH DEVIATION 🡪 complexity value did not fit (due to wrongly expected sorting algorithm)
  + (??? – did they do a second iteration???)
* Estimates of the EARLY level – component
  + Very good results – probably because the errors of the ECOS (as base) balance each other out in sum
* Estimates at the INTERMEDIATE Level – data
  + Accuracy: btw. -4.2 and 13.9 %
  + Possible reasons: resource factors or DVP’s might have been calibrated wrongly…
  + OR: factors that are not even considered as cost drivers, e.g. programming language, design patterns, etc. !! – they could be implicitly included in resource factors through calibration.. but that is rather (schwammig) 🡪 ADD ADDITIONAL METHOD FOR THAT!? In software engineering?!
* Estimates at the INTERMEDIATE level – Operation
  + Accuracy improved: -12.2 and 10 %
  + Causes: cost drivers that are not included… or jouleUnit inaccuracy
  + Calibration values for resource factors based on functional types may not be accurate, since they include implicit assumptions(?!) – not really all operations with the same functional type DO use the resources in the same way
  + Could also be the complexity issue…
* Estimates at the INTERMEDIATE level – component
  + Same calculation – but more accurate underlying ECOS values -> lead to more accurate ECC
  + Deviation of 3.5 %, accuracy of -1.1 to 2.2 %
  + But again.. the errors balance themselves out, so this is probably not a “typical” result
* Estimates at the INTERMEDIATE level – functional type
  + Calculation similar to ECC!
  + Very good estimation
  + Use results e.g. to find high-consumption hw components
* Estimates at the ADVANCED level - GENERAL
  + Uncertainty values can be picked!
  + Here: certainty of 99.7 % is picked!
* Estimates at the ADVANCED level – data
  + Accuracy: -2.3 % to 7.8 %
* Estimates at the ADVANCED level – operation
  + LIMIATION: no possibility to measure ENERGY CONSUMPTION AT THE HW COMPONENT LEVEL ☹ -> these complexities were left out
  + Deviation is between -5.3 % and 5.8 %
* Estimates at the ADVANCED level – component
  + Not all estimates improved – in the worst case accuracy decreased
* Estimates at the ADVANCED level – functional type
  + Most of the ECFTs accuracy has improved, but not all
  + 🡪 probably due to inaccuracies in JOULEUNIT!!
* Estimates at the ADVANCED level – scenario
  + Overall estimation is not much better than e.g. ECC, BUT – on level of ECOO per operation it IS better! (than the general values)
  + Better because real input values are used and not averages + ECC is way too good anyway

**Threats to Validity**

Internal Validity

* JouleUnit -> not meant to be used at such a fine grained level -> absolute accuracy of results is questionable
* JVM makes many optimizations -> invisible to the user and cannot be covered by SECoMo
* 🡪 improvements should be reached by using JMH, but that still leaves some leeway for errors
* -> also improvement: stopped all additional services so as not to mess up measurement with other services energy consumption.. but that is also not completely possible / accurate

External Validity

* Academic example… limitied size…
* E.g. the extremely high accuracy of ECC – rather by chance, not really representative

**SUMMARY**

* VERY IMPORTANT PREREQUISTIE: calibration needs to be performed well and with care 🡪 then SECoMo can perform very well
* Example evaluation
  + Early level estimates for whole shopping cart – 6 %
  + Intermediate level estimates for whole system - -1.1 % to 2.2 %
  + Advanced level estimates for ECC / ECFT : more than 2.8% (??)
* 🡪 LEAST accurate estimates at early level as there are not so many information yet – BUT – still good values that give an idea about the eco-costs, and the reasons – e.g. which components are more costly? 🡪 good to identify eco-cost hotspots 🡪 supports stakeholders with valuable information
* Eco-cost drivers represent / account for most of the eco-costs
* Auxiliary models are a good way to express costs & circumstances

# 14. Conclusion

SONSTIGES

* Secomo in early stages, example
  + E.g. no PLATFORM dependent information is available, like programming language (p. 82)

Ideas for my seminar thesis

(Explore eco-costs for application features AND architectures -> architecture foundation for seminar thesis, too (& compare) – can be used to define REQURIEMENTS! (also requirements part can be in there / compared)

* Go through a list of methods and models -> find some parts where “practical” method is missing to MEASURE / ESTIMATE – important for architecture / requirements part?!
* THERE it fits

Limitations – calibration model – not always feasible?!

+ aspects like programming language etc. not considered much

+ ONLY focus on execution of software – less design phase itself, or disposal -> so that is where to put it in Greensoft