The International Journal of Life Cycle Assessment (2021) 26:483–496
https://doi.org/10.1007/s11367-021-01870-9

LCI METHODOLOGY AND DATABASES

Prototypes for automating product system model assembly

Brandon Kuczenski[1] · Chris Mutel[2] · Michael Srocka[3] · Kelly Scanlon[4] · Wesley Ingwersen[5]

Received: 7 October 2020 / Accepted: 13 January 2021 / Published online: 4 February 2021

Abstract

Introduction The flexibility of life cycle inventory (LCI) background data selection is increasing with the increasing availability of data, but this comes along with the challenge of using the background data with primary life cycle inventory data. To relieve the burden on the practitioner to create the linkages and reduce bias, this study aimed at applying automation to create foreground LCI from primary data and link it to background data to construct product system models (PSM).

Methods Three experienced LCA software developers were commissioned to independently develop software prototypes to address the problem of how to generate an operable PSM from a complex product specification. The participants were given a confidential product specification in the form of a Bill of Materials (BOM) and were asked to develop and test prototype software under a limited time period that converted the BOM into a foreground model and linked it with one or more a background datasets, along with a list of other functional requirements. The resulting prototypes were compared and tested with additional product specifications.

Results Each developer took a distinct approach to the problem. One approach used semantic similarity relations to identify best-fit background datasets. Another approach focused on producing a flexible description of the model structure that removed redundancy and permitted aggregation. Another approach provided an interactive web application for matching product components to standardized product classification systems to facilitate characterization and linking.

Discussion Four distinct steps were identified in the broader problem of automating PSM construction: creating a foreground model from product data, determining the quantitative properties of foreground model flows, linking flows to background datasets, and expressing the linked model in a format that could be used by existing LCA software. The three prototypes are complementary in that they address different steps and demonstrate alternative approaches. Manual work was still required in each case, especially in the descriptions of the product flows that must be provided by background datasets.

Conclusion The study demonstrates the utility of a distributed, comparative software development, as applied to the problem of LCA software. The results demonstrate that the problem of automated PSM construction is tractable. The prototypes created advance the state of the art for LCA software.

Keywords Product system model · Foreground · Background · Linking · Software · Hackathon

1 Introduction

Modeling the potential direct and indirect environmental impacts of chemicals, materials, and the products is the aim of LCA studies. Including all indirect activities in the economy requires significant data on extractive, industrial, agricultural, transport, and disposal processes that capture their material and energy requirements and their environmental emissions and wastes. Practitioners must rely on existing datasets to fulfill these "background" data requirements. Furthermore, the data must all be interoperable so that they can be used together, and be based on coherent modeling assumptions. Existing life cycle

Communicated by Martin Baitz.

✉ Wesley Ingwersen
ingwersen.wesley@epa.gov

[1] GDIT, Santa Barbara, CA, USA

[2] GDIT, Zurich, Switzerland

[3] GreenDelta, Berlin, Germany

[4] Department of Defense, Office of the Assistant Secretary of Defense, Washington, DC, USA

[5] Office of Research and Development, US Environmental Protection Agency, Atlanta, GA, USA

inventory data are often not interoperable (Ingwersen 2015; Suh et al. 2016), but achieving this has been acknowledged as essential for broader LCA data sharing and usage of data from multiple sources (Ingwersen et al. 2015; Canals et al. 2016).

Identification, acquisition, and integration of data continue to pose a significant challenge to those who create models for LCA studies, particularly for studies that do not rely on a single life cycle inventory database for background data. This task involves acquisition of data from various sources, structuring the data for use in an LCA as a network of processes, and linking the processes together to build a product system model (PSM). Data discovery applications and data portals (e.g., Nexus (GreenDelta 2019), the Federal LCA Commons (USDA 2019), the Global LCA Data Access Initiative (UNEP/SETAC Life Cycle Initiative, 2019)) that provide structured life cycle inventory data from multiple sources are in their early stages. Data mining approaches are also in early development to create more data for life cycle assessment from public sources (Cashman et al. 2016; Ciroth and Srocka 2017). Building variations of PSMs using a single LCI database have also been automated in the Wurst software for ecoinvent (Mutel 2017). While there are great challenges to finding and or developing new LCI data for life cycle assessments, the greater challenge in the field right now might be linking LCI data together from different sources and assembling them in a structured PSM that can be reviewed, revised, and reused by others (Kuczenski et al. 2018). Because PSMs may consist of hundreds or thousands of processes combined from various sources, it is necessary to be able to describe those models precisely and in a transparent fashion.

Often, the description of a PSM can be done in two parts: the description of the foreground as a network of unit processes, nominally a tree whose root is the functional unit of the model, and the linking of terminal nodes or "leaves" in that network to process models found in background databases (Kuczenski 2019). While this description may not hold for some activities, particularly in chemical production (which may involve loops in the foreground system), it is suitable to describe the foreground of most product LCAs. The linking of LCI processes from various data sources could be accomplished using various methods. One method that has been proposed is through the creation of "bridge" processes, which prevent the need to alter the existing datasets that are linked (Ingwersen et al. 2018).

The purpose of this study was to develop prototype software capable of generating reusable product system models from a standardized product specification and automatically linking these models with existing LCI data from disparate sources to create PSMs. The study adopted a methodology that draws from the "hackathon" model (Briscoe and Mulligan 2014), in which participants are invited to develop code to solve to a well-specified technical problem. The contributions are available for review and adaptation by the general public.

2 Methods

A challenge was initiated to create prototype software that can assemble a full PSM from disparate data sources. Three experienced LCA software developers were invited to participate, with a fourth participating as a convener and moderator. Each developer was allotted up to 80 hours of independent work to perform the task. A set of requirements was given to each developer to follow (Table 1). Neither a graphical user interface nor elementary and product flow harmonization were required.

Table 1 Software requirements

No	Requirement
1	The software may make use of any existing software or source code, as long as the software/code license permits its reuse without any licensing fees
2	LCI data used may come from open or proprietary sources. However, any proprietary source data must be aggregated in a manner suitable for data sharing
3	The software must include automated data discovery functions to identify LCI to satisfy process requirements
4	The software must be functional with LCI datasets from the Federal LCA Data Commons
5	LCI data product flow names used in the PSM should be modified in content to the most limited extent possible. "Bridge processes" provide an alternative method of linking unit processes without altering the product flow names (Ingwersen et al. 2018)
6	All dependencies must be documented and be publicly available or otherwise made available to the project team
7	The software should read and generate LCI data in a described LCA data exchange format, preferably in JSON-LD according to the openLCA JSON-LD schema, but minimally in JSON-LD, ILCD, or ES2 formats
8	Development in the Python 3 language is preferred, but not required
9	The software must provide metadata on the PSM content and structure. Model contents may include description of sub-PSMs within the modeled PSM. See Kuczenski et al. (2018), Sect. 5
10	The software must work on current Windows and Macintosh operating systems and preferably on others, including Linux

The developers were given data for an assembled product and asked to use their software to develop PSMs consisting of a foreground system for the selected assembly along with selected supporting background systems. The selected assembled product was the landing gear for a Boeing F/A-18E/F "Super Hornet" aircraft (F-18 LG). The landing gear consisted of independent nose (1) and rear (2) systems in the form of assemblies, where each assembly consisted of one or more subassemblies and/or individual components (Fig. 1). The data were provided in the form of a standard Bill of Materials (BOM) and were confidential.

We performed two tests of the extensibility of the prototypes, testing their performance in creating PSM models from two additional datasets. The first dataset was a BOM spreadsheet for the assembly of a printed circuit board (PCB). The spreadsheet included twelve components, of which seven were assigned part numbers and five were given only text descriptions. The second dataset was a life cycle inventory unit process for the sorting of construction and demolition debris in a material recovery facility (CDD MRF) taken from the Federal LCA Commons (USDA 2019). This dataset was provided in JSON-LD openLCA format and had approximately 65 exchanges, including both intermediate and elementary flows.

3 Results

The input BOMs could all be interpreted as trees of assemblies, sub-assemblies, and components that could be converted into a foreground PSM. The components, or leaf nodes of the trees, represent inputs to the foreground from the background. Converting a BOM into a complete PSM was shown to require three modeling steps: constructing the trees; determination of quantitative properties of the leaf nodes; and linking the leaf nodes to background databases. These were accompanied by a fourth step of describing the models in a

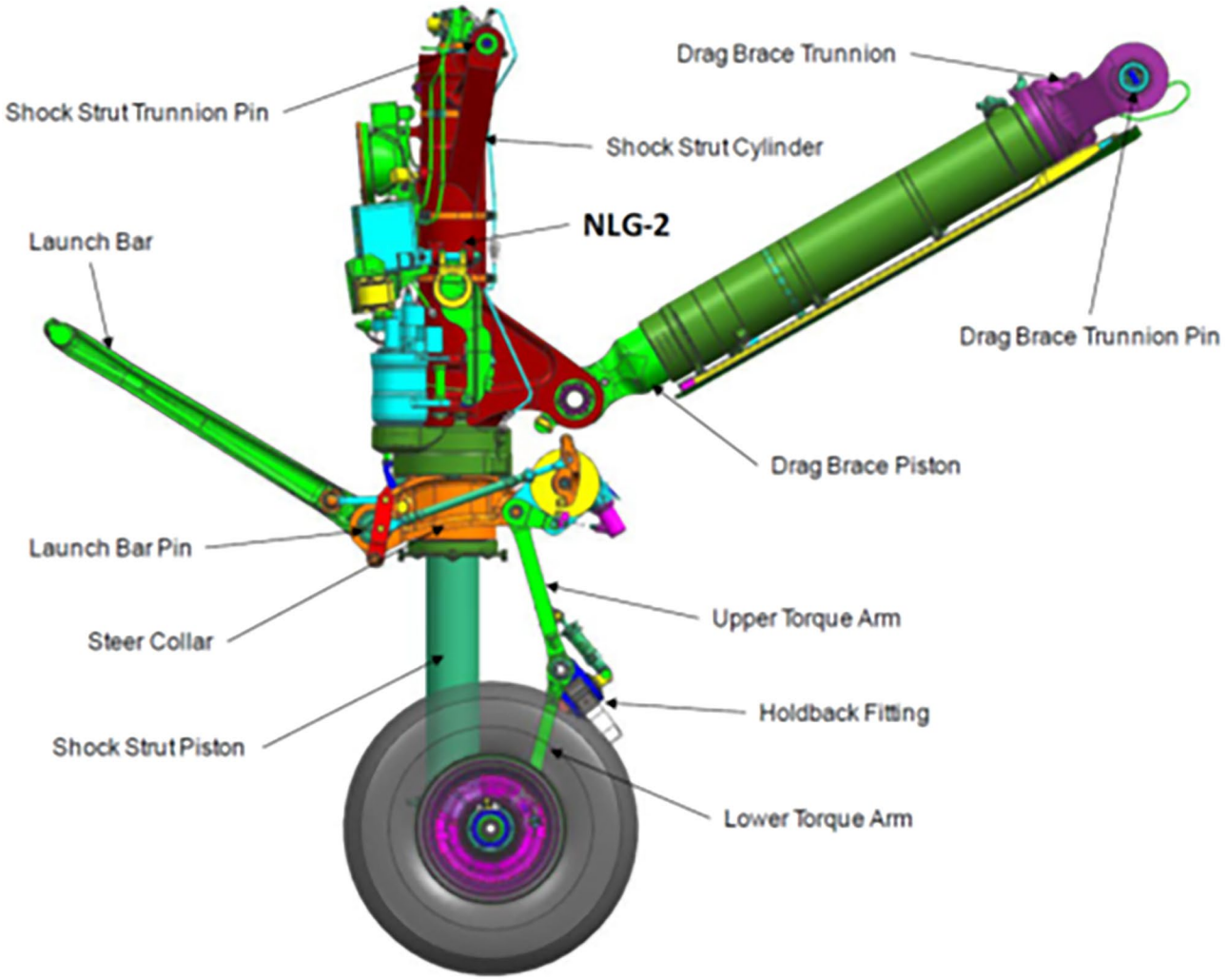


Fig. 1 Schematic drawing of one landing gear product system

format that can be read into an LCA program, referred to as "serialization." A complete solution to the study problem would complete all four steps. In this section, each prototype is presented along with a brief summary. This is followed by an assessment of the prototypes in their success in creating PSMs from the F-18 LG dataset and with alternative datasets.

3.1 Presentation of prototypes

Three prototypes *pslink*, *antelope*, and *perdu* were initially constructed by the three developers. All prototypes are written in Python and require a Python interpreter to use or launch. The prototypes are designed to run independently, though they required different levels of configuration and interaction.

3.1.1 pslink

pslink imports and stores BOM data as trees and uses them to create a foreground PSM. The user must then collect or mine data on the material composition and dimensions of the component inputs to the foreground model/leaves of the PSM tree, and store these attributes as key-value pairs in a text file for each component. The *pslink* quantitative solver tests each attribute file against a collection of symbolic formulas that compute the volume of various solids. These formulas can be dynamically registered, allowing a user to specify formulas specific to the input data. If a formula is found that can be computed using the supplied attribute information, the result is taken to be the volume of the part. In case multiple formulas could be satisfied by the supplied data, only the first formula encountered is used. *pslink* also requires that the user store density data on the materials identified in the composition data. Keywords from the component attribute data are matched against the stored densities. If a match is found, an exchange can be created, converting the previously calculated volume to a mass for a given material in a component. If multiple matches are found, the mass is partitioned equally among them.

The linking step in *pslink* starts from a simplified semantic graph that a user creates for the materials present in the attribute data. The users input is a simple text file where each line declare the material name and then one or more related names, and their relationship to the material name, where relationship types are "same as, ="; "broader than,^"; or "derived from, <." An example of records related to a steel is as follows:

"steel," "ferrous metal"^
"steel product," "steel" <
"stainless steel," "steel"^
"stainless steel alloy," "alloy"^, "stainless steel"^
"alloy 660," "stainless steel alloy 660" =, "stainless steel alloy"^

A network of semantic relations is created from this file, like shown in Fig. 2, and stored as a graph.

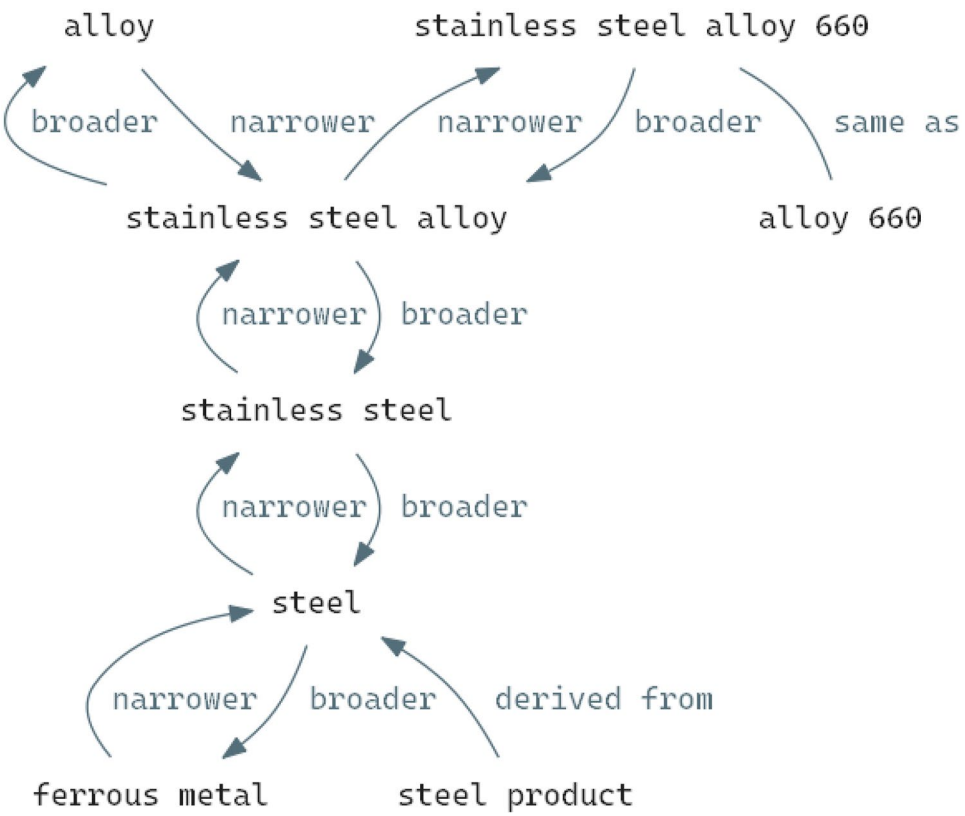


Fig. 2 Semantic network of product relations in pslink

In order to derive the product flow information from the background database of interest, the user is required to extract a list of processes and product flows names and compile them in a comma-separated text file. The products of the background database are bound to the products of the graph via a lexical analysis. The function of the lexical matching can be configured, and different functions are implemented in *pslink* that consider specific syntactic elements of product names in LCI databases. For each binding, the lexical matching factor, f_l, is stored which has a value in the range of [0,1].

For each product in the foreground system, entry points into the semantic network are again searched via a lexical analysis. Starting from these entry points the graph is traversed along the relations to the nodes with bindings to products of the background database. Each relation is assigned a relation factor, f_r, that is specific for the relation type (e.g. "same as": 1.0, "broader"/"narrower": 0.75, "derived from": 0.5). A traversal factor f_t is calculated for each path from an entry point to a binding of a background product by multiplying the lexical and relation factors across the segments, s, of the path p:

$$f_t = \prod_{s \in p} f_l f_r$$

Figure 3 illustrates this procedure. The calculation of the traversal factors can be documented with the integrated *explain* function, which presents a list of the relations used to determine each factor.

The products of the background databases with the highest traversal factors are finally assigned to a product of the foreground system in the linking step. The collection of flows and linked unit processes are then serialized as JSON-LD following the openLCA schema (Srocka et al. 2020), allowing them to be imported into OpenLCA software for practitioner use. If the model is imported into an existing database that includes the linked background processes, then those processes would be linked to the designated datasets upon import.

3.1.2 Antelope

antelope is a Python tool based on the Antelope LCA framework (Kuczenski and Beraha 2015). *antelope* also converts BOMs into product system models with a tree-like structure, with the reference flow at the root of the tree. *antelope* includes an algorithm for detecting and removing duplicate subassemblies using tree isomorphism (Valiente 2002). By treating duplicated assemblies as distinct product models, the duplicates are removed and the models are re-used in separate places in the product system.

When a PSM is described as a tree, the interior nodes and leaf nodes serve different purposes (Fig. 4). Branches connecting interior nodes reflect direct relationships between foreground elements and define the structure of the PSM. Leaf nodes, on the other hand, represent flows which enter or leave the foreground system, and which must be linked in order to compute the model. If they are linked with a background

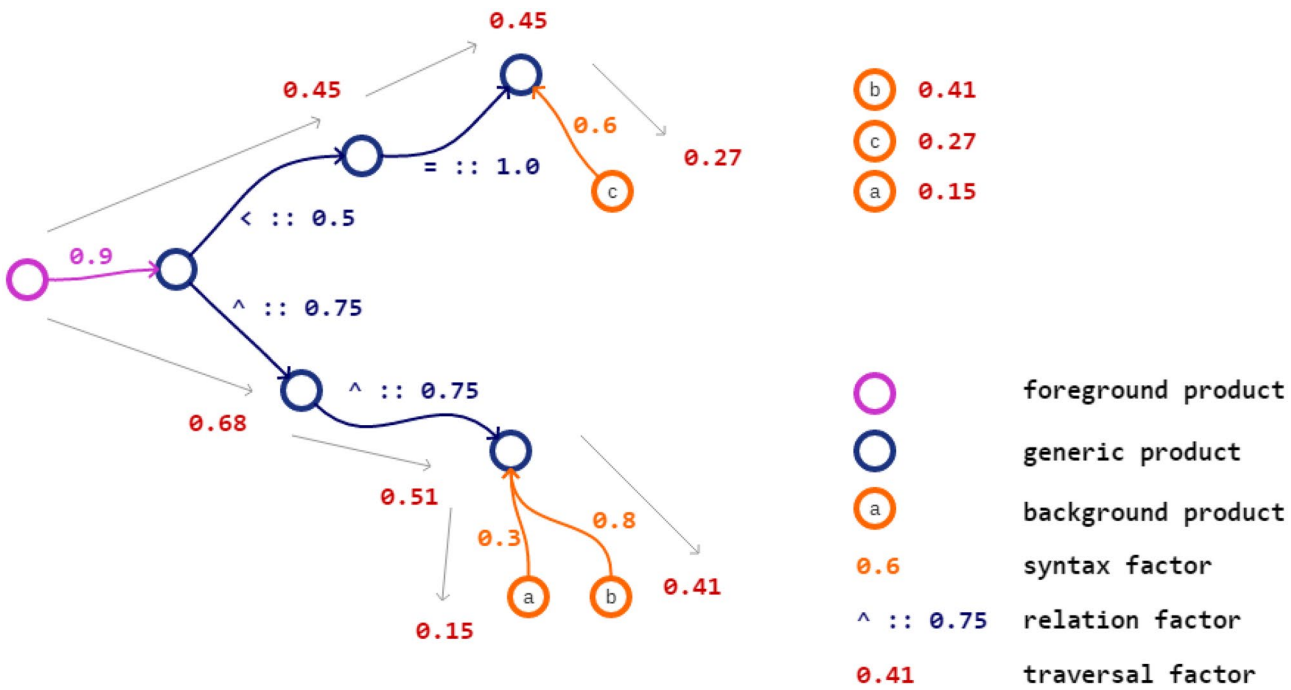


Fig. 3 Calculating the traversal factor in pslink

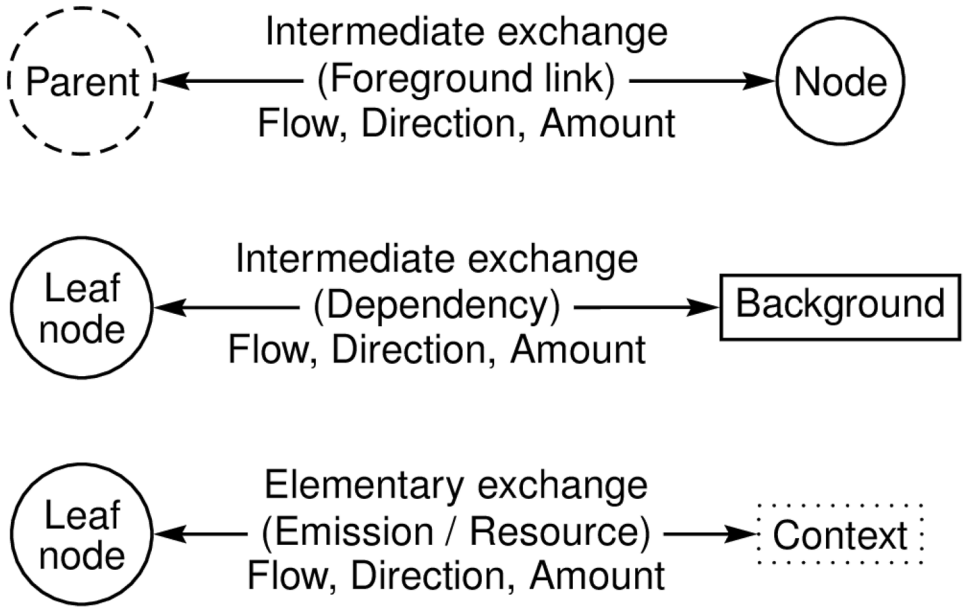


Fig. 4 Exchanges and dependency relationships in a product system model. Circles represent foreground nodes, solid rectangle represents activities in a background database, and dotted rectangle represents an environmental context. The direction of the exchange can be either as an input to or output from the foreground

activity, they become intermediate exchanges that represent a dependency of the foreground on that activity. Note that the direction of dependency is unrelated to the physical direction of the flow (i.e., a foreground process could depend on both material suppliers and waste disposal services). If they are linked with an environmental context, they become elementary exchanges (none of those were included in the current study).

Each link in the model must be assigned a weight, which is the exchange value of the link, or the quantity of the child flow that is required by the parent. During traversal of the tree, each node's weight is calculated by multiplying its parent's node weight by its exchange value. The record of this traversal can be used to reconstruct the model. In the antelope traversal implementation, any node can be specified as private, in which case the nodes interior to that node's subtree will be concealed and only the aggregate result reported.

Once the product models are created and linked to background datasets, the product system can be described as an LCA disclosure, a data structure that simplifies the expression of the model structure (Joyce 2019; Kuczenski 2019). Disclosures are constructed from a traversal of the product tree under a particular scenario. The *antelope* user can specify certain sub-assemblies or the whole assembly to be aggregated during the traversal, so that private or proprietary information in the PSM can be concealed from view while still contributing to the disclosure. The disclosures are saved as excel files for easier accessibility.

3.1.3 Perdu

perdu is a Flask (Pallets Project 2020) web application for matching product data against standard industry and product classifiers. *perdu* embeds classification systems, including the Global Product Classification (GPC), the North American Industry Classification System (NAICS), and US Bureau of Economic Analysis Detail Input-Output schema as implemented in the USEEIOv1 model (Yang et al. 2017). *perdu* extracts names and descriptions from these classification systems and uses that data to build indices for searches. Various search algorithms provided by the Whoosh library (Chaput 2012) are implemented to search these indices using terms from the product data.

The web application was designed around a simple workflow (Fig. 9). A user can upload a spreadsheet or table (Excel or csv file) with "name" and "description" fields or an openLCA JSON-LD archive with processes and flows to be matched. *perdu* will extract the names and descriptions from the tabular data or the processes and flows from the JSON-LD and store them as a term list and present them to the user for interactive matching. With the product data loaded and names presented to the user, the user selects a classification system to use, and an initial list of match suggestions is provided. The search terms can be refined in cases where the given labels are difficult for automatic algorithms to interpret. Multiple matches per query are possible. The user then selects a proposed match for each term and describes the relationship type; matches are either "exact," "approximate," "narrower than," or "broader than." The interface does not include any mechanism for quantitative characterization (i.e. the mass or dollar value of each node). The review process occurs in a browser window, and after it is complete, the user can export a static file that contains the saved matching information. Following interactive search, review, and linking, *perdu* can produce output files of matches in a CSV, JSON, or Turtle (TTL) formats. The web application also includes a standalone search window.

Table 2 Summary characteristics of three initial prototypes

Prototype	Primary dependencies	Data discovery	Supervised or unsupervised matching?	Datasources linked	Product system statistics?	Output formats
pslink	olca-py	Semantic similarity	Unsupervised	USEEIOv1.1, LCA Commons, ecoinvent v2.2	Y	JSON-LD
antelope	Antelope LCA	Tree isomorphism	Supervised	USEEIOv1.1, USLCI, ELCD	Y	JSON, Excel
perdu	Whoosh, RDFLib	Metadata search	Supervised	USEEIOv1.1, GPC, NAICS	N	CSV, TTL, JSON-LD

3.1.4 Summary

Table 2 presents a summary of the prototypes.

3.2 Prototype testing

3.2.1 Landing gear model

The principal test case, the F-18 LG dataset, consisted of eight assemblies supplied in two BOMs in the form of Excel spreadsheets. The BOM was anonymized using a software routine included in the *antelope* contribution and is henceforth referred to as F-18 LG ANON. The resulting foreground models were observed to have a total of 1528 distinct elements, where each element corresponds to a row in a BOM. These models included 1045 distinct flows, of which 810 were leaf nodes; some of which were reused more than once.

In order to prepare *pslink* for use with the F-18 LG ANON foreground model, a mechanical parts database (PartTarget Inc. 2019) was mined to extract material composition and dimensions. As different components can have very different attributes that describe their dimensions, it was necessary to accommodate a wide range of attribute names. Density data for materials identified in the parts were manually input from various online sources (amesweb.info 2019; MatWeb, LLC 2020; ThoughtCo Inc. 2020). A semantic graph of associated material relationships was manually created. *pslink*, enabled with these input data, constructed PSMs from the F-18 LG ANON dataset and selected background datasets. The linkages between the F-18 LG ANON dataset and background databases are summarized in Table 3.

The data mined had identified 27 unique materials in the F-18 LG ANON dataset that became the basis of *pslink*'s matches to background database products. In all three background sources, corresponding product flows were found for nearly all materials (25 or 26 out of 27). Each product flow could then be linked to one or more providing processes; the database with the most unique product flows (Ecoinventv2.2) able to provide the most potential matches to these materials (66). But the number of foreground model leaf nodes for which linkages were created was the same between the USEEIOv1.1 and Ecoinventv2.2 sources (77) and only slightly less for the LCA Commons source (70), which could be explained by the one less material provided by the LCA Commons in comparison with the other two databases.

The created models were exported from *pslink* in JSON-LD and successfully imported into openLCA (Fig. 5) where it was verified that life cycle inventory results could be calculated to generate nonzero scores (Fig. 6). However, the simple tree-traversal algorithm led to doubled exchanges in 19 duplicate processes.

antelope captured the tree structure of the F-18 LG ANON dataset, including detection and removal of duplicated subassemblies. Within the structure of the model, nineteen duplicate subassemblies were detected, making up 126 elements or about 8% of the total. The models were exported as disclosures in spreadsheet format to facilitate review. Visualizations of portions of the foreground and background for the study test case are shown in Figs. 7 and 8. This prototype did not use data discovery to determine flow properties of parts, and instead used manual designation of each part's "part type," "material," and "size" as a basis to link to valid datasets in the USLCI, USEEIO, and ELCD databases. Because this procedure was done by

Table 3 *pslink* match statistics for F-18 LG ANON dataset

FS	FS unique materials	BS	BS unique product flows	Materials with at least 1 possible match	Possible material matches	Leaves with links established
F-18 LG ANON	27	ecoinvent v2.2	1924	26	66	77
F-18 LG ANON	27	USEEIOv1.1	388	26	30	77
F-18 LG ANON	27	LCA Commons	752	25	32	70

FS foreground data source, *BS* background data source

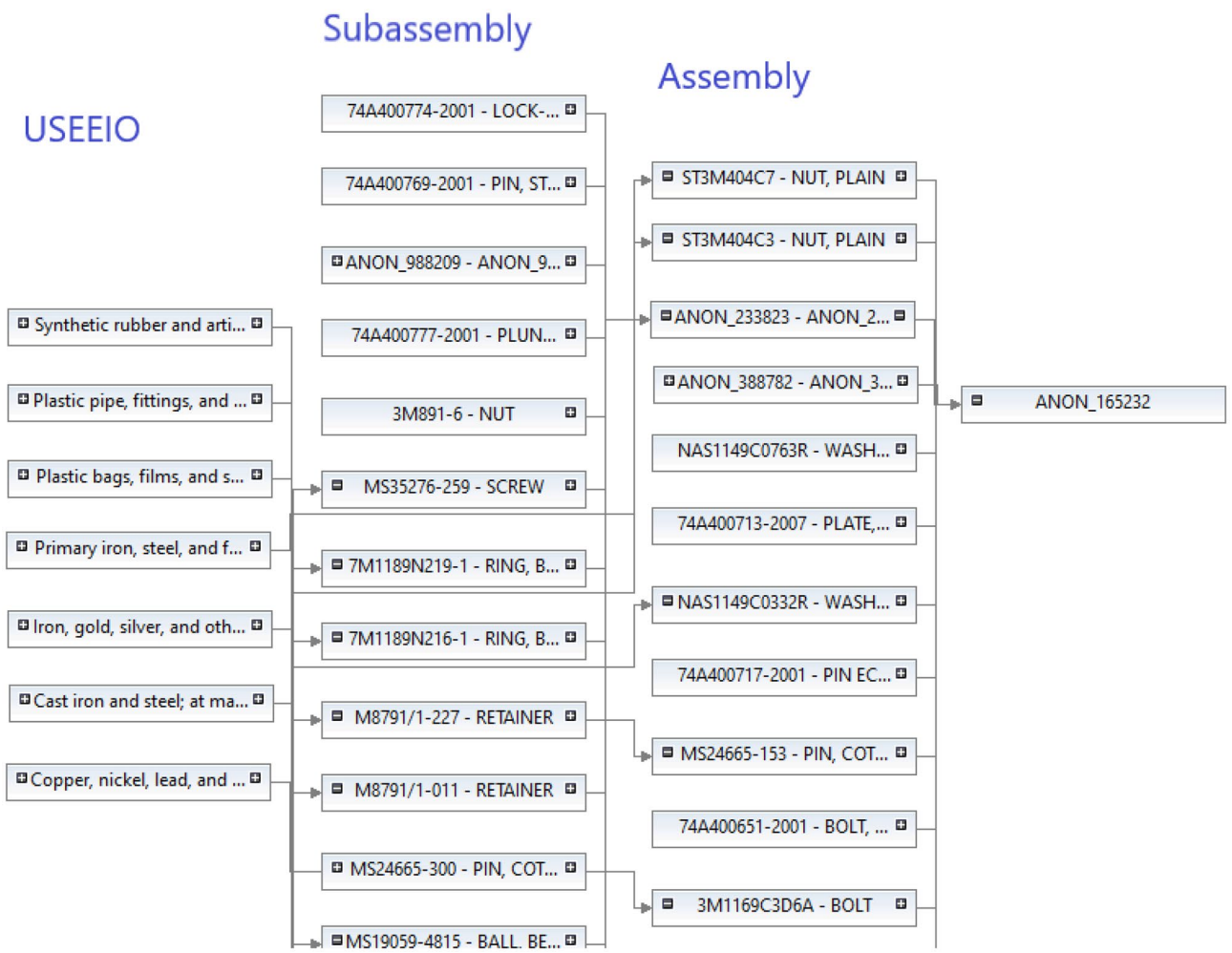


Fig. 5 PSM of landing gear connected to USEEIO in openLCA

hand, it was very time-intensive and potentially error-prone, but could be applied consistently to all 810 unique components. A total of 25 USLCI processes, 19 USEEIO sectors, and 17 ELCD processes were identified as linking targets. The curated characterizations were included in the *antelope* contribution in spreadsheet form. The price and mass characterizations were random.

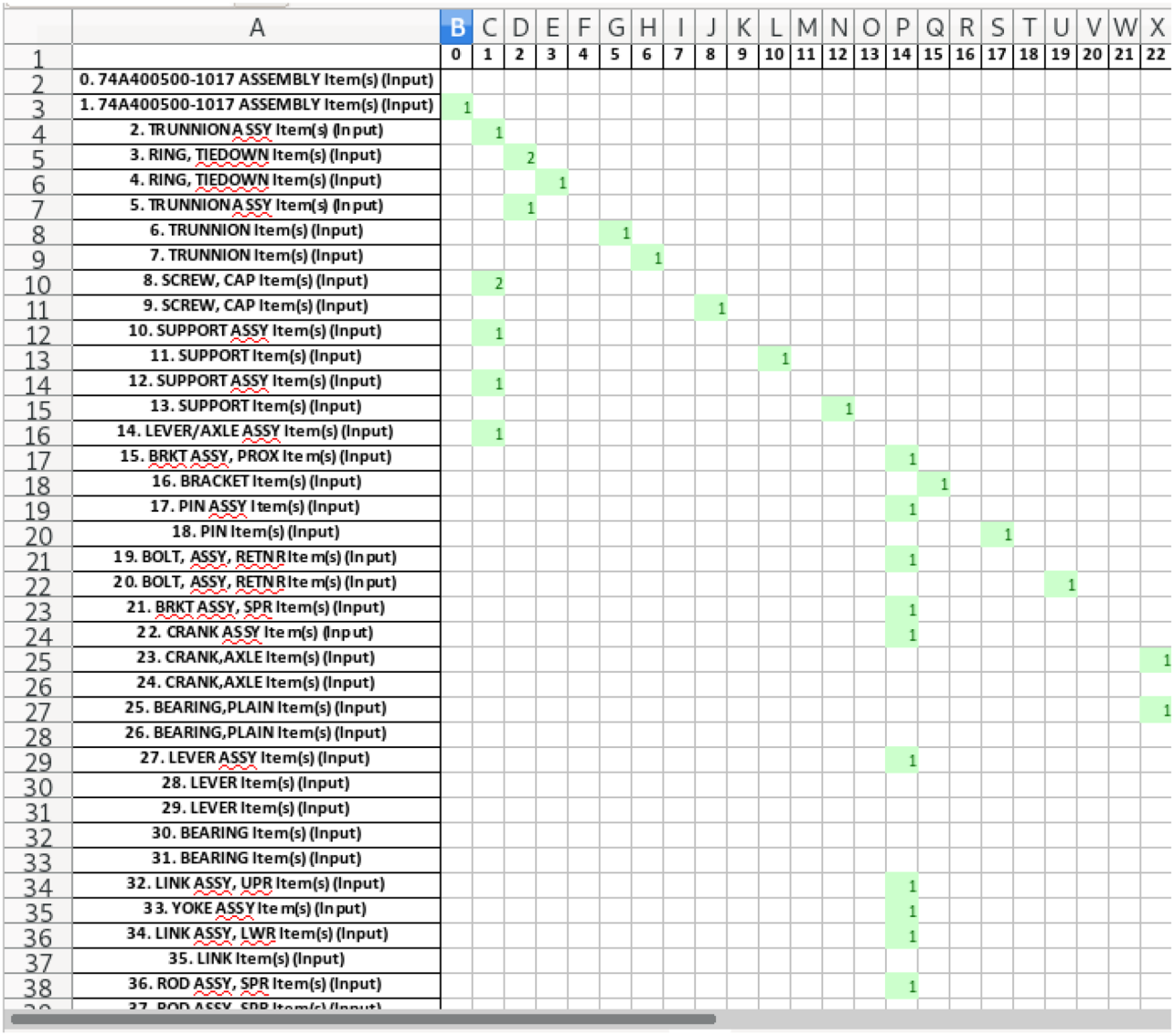
perdu did not perform automatic model generation, but instead focused on the classification of leaf nodes in the model. The prototype included search indexing frameworks for three different classification schemes: NAICS, GS1, and the US system of national accounts, as implemented for LCA in the USEEIO database (Yang et al. 2017). Other classifications can be added by following the patterns laid out in the prototype.

3.2.2 Handling of alternative input datasets

Because of the similarity of first alternative dataset, the PCB BOM, to the main test case, all three tools were successful

Contribution	Process	Amount	Unit
100.00%	ANON_165232	4.04240	kg
64.45%	ANON_233823 - ANON_233823 Sub-assembly	2.60531	kg
61.69%	[redacted]	2.49373	kg
61.69%	Stainless steel; Manufacture; Production mix, at plant; 316 2B, 80% recycled - GLO	2.49373	kg

Fig. 6 LCIA results computed from the linked landing gear model in openLCA



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	0. 74A400500-1017 ASSEMBLY Item(s) (Input)																							
3	1. 74A400500-1017 ASSEMBLY Item(s) (Input)	1																						
4	2. TRUNNION ASSY Item(s) (Input)		1																					
5	3. RING, TIEDOWN Item(s) (Input)			2																				
6	4. RING, TIEDOWN Item(s) (Input)				1																			
7	5. TRUNNION ASSY Item(s) (Input)			1																				
8	6. TRUNNION Item(s) (Input)						1																	
9	7. TRUNNION Item(s) (Input)							1																
10	8. SCREW, CAP Item(s) (Input)		2																					
11	9. SCREW, CAP Item(s) (Input)									1														
12	10. SUPPORT ASSY Item(s) (Input)		1																					
13	11. SUPPORT Item(s) (Input)											1												
14	12. SUPPORT ASSY Item(s) (Input)		1																					
15	13. SUPPORT Item(s) (Input)													1										
16	14. LEVER/AXLE ASSY Item(s) (Input)		1																					
17	15. BRKT ASSY, PROX Item(s) (Input)															1								
18	16. BRACKET Item(s) (Input)																1							
19	17. PIN ASSY Item(s) (Input)															1								
20	18. PIN Item(s) (Input)																		1					
21	19. BOLT, ASSY, RETNR Item(s) (Input)															1								
22	20. BOLT, ASSY, RETNR Item(s) (Input)																				1			
23	21. BRKT ASSY, SPR Item(s) (Input)															1								
24	22. CRANK ASSY Item(s) (Input)															1								
25	23. CRANK,AXLE Item(s) (Input)																							1
26	24. CRANK,AXLE Item(s) (Input)																							
27	25. BEARING,PLAIN Item(s) (Input)																							1
28	26. BEARING,PLAIN Item(s) (Input)																							
29	27. LEVER ASSY Item(s) (Input)															1								
30	28. LEVER Item(s) (Input)																							
31	29. LEVER Item(s) (Input)																							
32	30. BEARING Item(s) (Input)																							
33	31. BEARING Item(s) (Input)																							
34	32. LINK ASSY, UPR Item(s) (Input)															1								
35	33. YOKE ASSY Item(s) (Input)															1								
36	34. LINK ASSY, LWR Item(s) (Input)															1								
37	35. LINK Item(s) (Input)																							
38	36. ROD ASSY, SPR Item(s) (Input)															1								

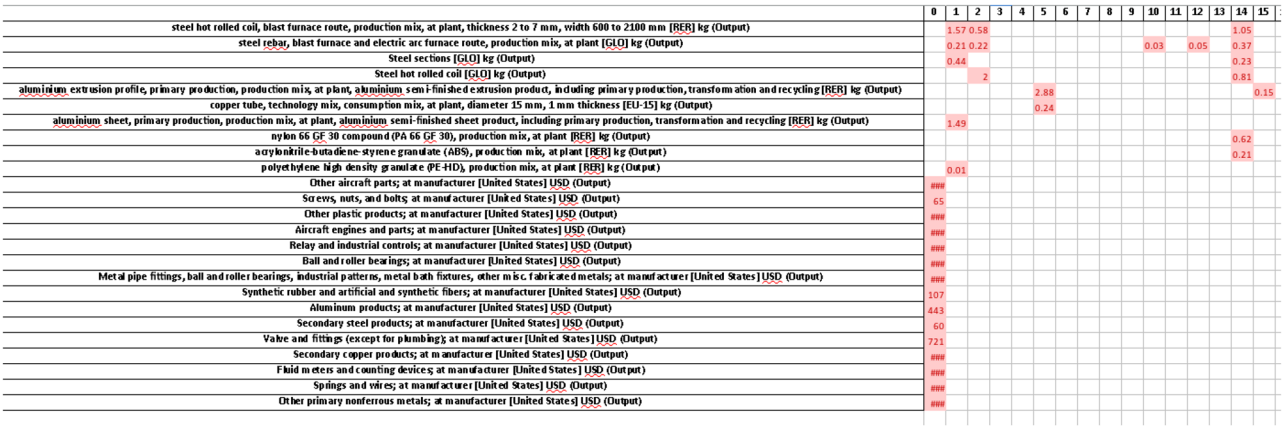
Fig. 7 A portion of a model's internal nodes (foreground) documented in a spreadsheet

in processing the PCB BOM. There were no duplicate flows. *antelope* and *pslink* prototypes correctly interpreted the spreadsheet and constructed a foreground model; however, the *pslink* prototype omitted the flows that had no part numbers. Because of lack of any text descriptions accompanying the part numbers in the BOM, the flow property description step failed for both *pslink* and *antelope*, and no linking was performed. With the *perdu* prototype, on the other hand, it was possible to identify matching classifications from the available schemes.

The attempt to use the second alternative dataset, the CDD MRF unit process, was less successful. Although openLCA and the *antelope* core software can both access OpenLCA datasets, *perdu* was the only prototype able to accommodate input data in this format directly (see Fig. 9). The CDD MRF test revealed another issue that was relevant to the project. Ten exchanges in the model included links to "bridge processes" for various activities found in the USEEIO database, using the "defaultProvider" property defined in the openLCA schema (Srocka et al. 2020). However, the bridge processes themselves were not included along with the CDD MRF dataset, and so there was no way for any software to "build" the links or to identify the linking targets. This outcome revealed that further development is needed to successfully make use of bridge process specifications.

3.3 Autoprox: an additional prototype for direct open LCA software integration

The challenges associated with the second alternative dataset—namely an existing unit process—inspired creation



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
steel hot rolled coil, blast furnace route, production mix, at plant, thickness 2 to 7 mm, width 600 to 2100 mm [RER] kg (Output)		1.57	0.58												1.05	
steel rebar, blast furnace and electric arc furnace route, production mix, at plant [GLO] kg (Output)		0.21	0.22								0.03		0.05		0.37	
Steel sections [GLO] kg (Output)		0.44													0.23	
Steel hot rolled coil [GLO] kg (Output)			2												0.81	
aluminium extrusion profile, primary production, production mix, at plant, aluminium semi-finished extrusion product, including primary production, transformation and recycling [RER] kg (Output)						2.88										0.15
copper tube, technology mix, consumption mix, at plant, diameter 15 mm, 1 mm thickness [EU-15] kg (Output)						0.24										
aluminium sheet, primary production, production mix, at plant, aluminium semi-finished sheet product, including primary production, transformation and recycling [RER] kg (Output)		1.49														
nylon 66 GF 30 compound (PA 66 GF 30), production mix, at plant [RER] kg (Output)															0.62	
acrylonitrile-butadiene-styrene granulate (ABS), production mix, at plant [RER] kg (Output)															0.21	
polyethylene high density granulate (PE-HD), production mix, at plant [RER] kg (Output)		0.01														
Other aircraft parts; at manufacturer [United States] USD (Output)	###															
Screws, nuts, and bolts; at manufacturer [United States] USD (Output)	65															
Other plastic products; at manufacturer [United States] USD (Output)	###															
Aircraft engines and parts; at manufacturer [United States] USD (Output)	###															
Relay and industrial controls; at manufacturer [United States] USD (Output)	###															
Ball and roller bearings; at manufacturer [United States] USD (Output)	###															
Metal pipe fittings, ball and roller bearings, industrial patterns, metal bath fixtures, other misc. fabricated metals; at manufacturer [United States] USD (Output)	###															
Synthetic rubber and artificial and synthetic fibers; at manufacturer [United States] USD (Output)	107															
Aluminum products; at manufacturer [United States] USD (Output)	443															
Secondary steel products; at manufacturer [United States] USD (Output)	60															
Valve and fittings (except for plumbing); at manufacturer [United States] USD (Output)	721															
Secondary copper products; at manufacturer [United States] USD (Output)	###															
Fluid meters and counting devices; at manufacturer [United States] USD (Output)	###															
Springs and wires; at manufacturer [United States] USD (Output)	###															
Other primary nonferrous metals; at manufacturer [United States] USD (Output)	###															

Fig. 8 A portion of a model's leaf nodes (background) documented in a spreadsheet

of an additional prototype, *autoprox*, that can link a unit process with background datasets, and make these links in the form of bridge processes (Ingwersen et al. 2018). *autoprox* is written in Kotlin and depends on the openLCA core API and a Java virtual machine. It reads directly from an openLCA database rather than from input files like the Excel files storing the BOMs. The user supplies two inputs: the id of the unit process in the openLCA database for which linking to datasets to provide product flow inputs is desired, and one of three alternative *matchers*, which are implementations of common matching algorithms. The best matches are then used to automatically create bridge processes within the given openLCA database between the given process and other available processes in the database.

The three different *matchers* each use principles of lexical and/or semantic matching between flow names (Pawar and Mago 2018), but use different forms of information to evaluate the match. The first matcher,

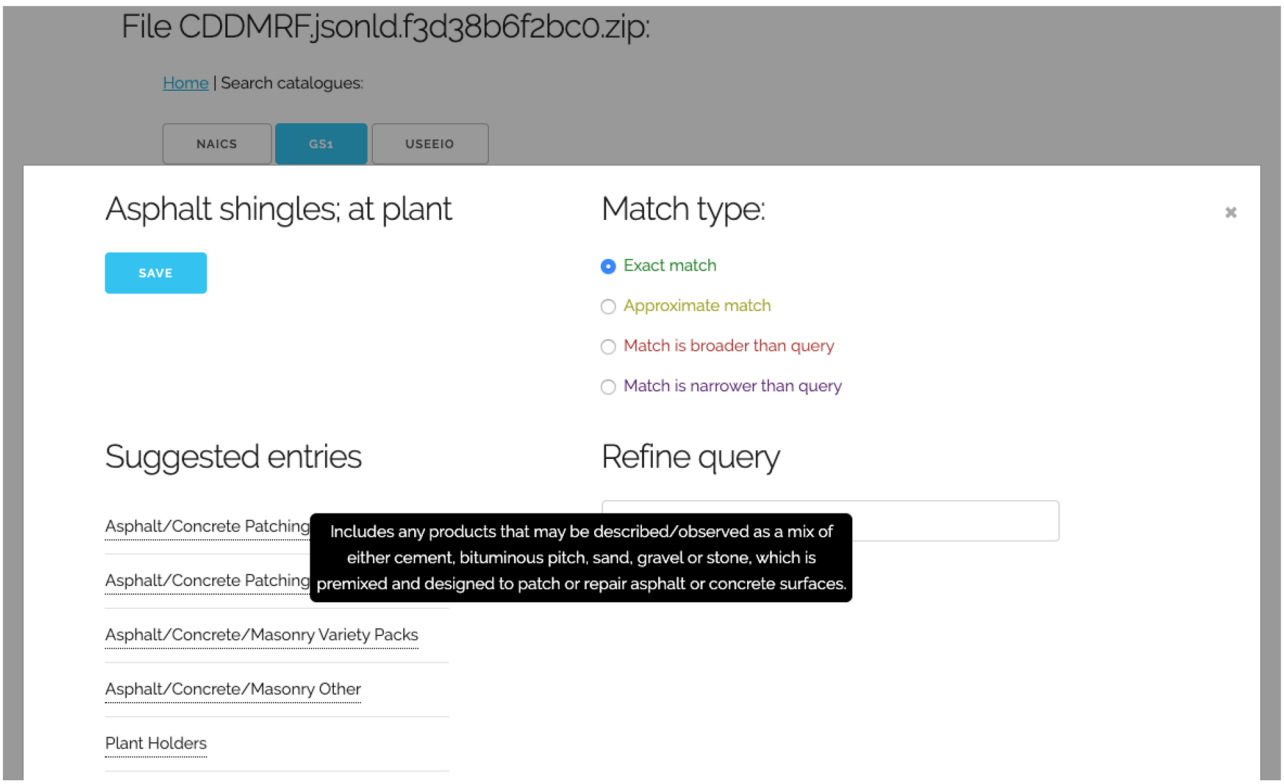


Fig. 9 Matching the CDD MRF dataset using Perdu

BigramsDiceMatcher, uses "bigrams," which are sets of two consecutive letters, syllables, or words in a text string. It works well for matching flows that have distinctive words or phrases, like "asphalt shingles," but less well for flows that are made up of common or generic words. The second matcher, *InfoContentMatcher*, uses the "information content" of a word, defined as the length of the word times the inverse of its frequency in the collection of candidate processes. Longer words and less-frequent words have higher information content. The information content of flow words is then compared with the information content of provider flows to find the best matches. The third matcher, *WordNetPathMatcher* also uses the information content of words, but combines this information with a semantic similarity score derived from a large natural language corpus. The reader is encouraged to read more about these matchers on the *autoprox* site (link provided in the Code Availability section).

Like the other solutions, *autoprox* is limited to identifying possible matches from the background processes that are available, which in this case come from the given openLCA database. The techniques implemented in the *autoprox* prototype could provide inspiration for further research in semantic and lexical matching for LCA applications.

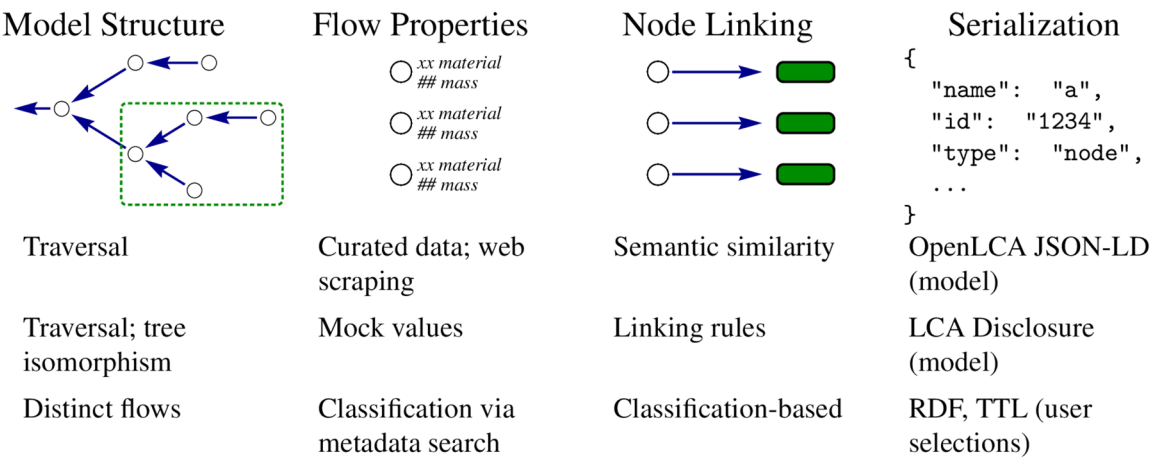
4 Discussion

While each prototype has its strengths and deficiencies, none was capable of transforming a product specification into a complete PSM with all product flows provided by background datasets without manual intervention. We found that the solution to automatically generating product system models from standardized product data has several steps, and that some of those steps are more amenable to automation than others. Each prototype demonstrated progress toward this end, yet all still require user involvement for one or more steps.

Figure 10 shows the overall problem and identifies the key contributions of each prototype.

The *pslink* and *antelope* prototypes were capable of transforming BOM input data into a product system model to serve as a foreground system. Notably, this was not the first demonstration of BOM to LCI automation (Sundaravaradan et al. 2011; Tao et al. 2014; McIntosh and Koffler 2014; Mishra and Singh 2019), but these are the first open source implementations of such conversions. Detecting and removing duplicate subtrees may be required to avoid double counting, with *antelope* the only prototype designed to detect and remove duplicate subtrees. *antelope* is also capable of optionally anonymizing assemblies and subassemblies in a BOM, which adds value for protecting primary data confidentiality. Modeling of the second test dataset, a unit process, had different requirements, which were met through the creation of an additional prototype, *autoprox*.

The flow property description step was found to be the most challenging of the four. No prototype was able to perform this step without user input. Each component from a BOM that was determined to be a leaf node was identified only as a "piece," but background datasets provide flows in terms of physical extent, most frequently mass, or by price. Therefore, before linking could be performed, each component required qualitative and quantitative description. This step has not received much attention in the literature for process LCA, although it is also present in hybrid LCA, where physical flows must be expressed in terms of their monetary value in order to link to economic input-output models (Suh and Huppes 2005). Each prototype took a unique approach to flow description. *pslink*



pslink	Traversal	Curated data; web scraping	Semantic similarity	OpenLCA JSON-LD (model)
antelope	Traversal; tree isomorphism	Mock values	Linking rules	LCA Disclosure (model)
perdu	Distinct flows	Classification via metadata search	Classification-based	RDF, TTL (user selections)

Fig. 10 The four steps for developing a PSM from a product dataset identified in the study, and the contributions of each prototype to each step

takes a material-centric approach, where it uses component material composition and dimensions along with density data and registered volume formulas to calculate the mass of each material in a component. For *pslink* to perform this step, the user is required to collect these physical data and prepare them in defined formats, some of which can be automated. In the *antelope* case, the qualitative description of each leaf node is performed by the user by manually mapping a flow to a reference background dataset product flow, and then the mappings were used to generate mock quantitative data according to heuristics. *perdu* works on the assumption that the flow property description problem is a nomenclature problem, recognizing the lack of a common nomenclature for products/intermediate flows across life cycle inventory sources. Classifications systems are a common way of product identification used in product category rules and environmental product declarations (Ingwersen and Stevenson 2012; Minkov et al. 2015), and *perdu* classifies flows according to one of these formal systems. In an application, it is possible that a modeler will already have direct access to information about the parts contained in their product system, and this information must be used in a tool-dependent manner to describe these properties.

In the linking step, the use of a semantic linking demonstrated by *pslink* and *autoprox* shows great promise to select "best-fit" background process to provide the required foreground input flow. The use of large lexical databases and machine learning algorithms has been identified as a promising approach for reconciling different background LCI databases (Davis et al. 2010; Kuczenski et al. 2016). The *pslink* prototype implementation depended on a locally-defined semantic network, and the *autoprox* prototype extended this approach to use a variety of matching algorithms, including more robust semantic databases. These techniques can be used for foreground systems when sufficient lexical content is available. The novel linkage metrics from *pslink*—the lexical, relation and traversal factors—demonstrate a way to assess the validity of machine-based links. However, the approaches may generate false matches, suggesting that the results would still need to be manually reviewed. They also can require substantial computational resources. Each tool was limited to searching one user-specified background datasource specified by the user at a time.

Serialization is the simplest problem to solve, with the *pslink* contribution demonstrating straightforward implementation of JSON-LD export and linking to target background datasets present in openLCA. The *perdu* prototype does not construct LCA models. *perdu* prepares lists of database matches for further use in a linked data approach, which would be possible to integrate with emerging data resources such as BONSAI (BONSAI 2020). The LCA disclosure framework used in *antelope* can define the relationships, but further development is required to support serialization. Only *pslink* met the study requirements to generate direct inputs to openLCA but a linked dataset is dependent upon the presence of the background dataset in the software to forge the link. In contrast, the *antelope* prototype uses an LCA disclosure format that is not limited to a single source of background data. This is intended to be accomplished by providing "stable semantic references" to the background datasets used. In practice, this is accomplished by naming each background dataset with both an "origin" that identifies the specific data source, and an "external reference" that unambiguously specifies a particular dataset. In principle, an LCA software system could obtain the exact dataset specified as long as it can properly interpret the named origin. While it would be straightforward to express the LCA disclosure format using the openLCA schema, this functionality is not yet implemented. It would still be subject to the same constraints as the *pslink* serialization: the referenced datasets would need to be present in the database in order to be linked.

4.1 Future use in LCA software

These prototypes are not yet integrated into the LCA software tools already used in the community by practitioners. However, a number of assets they share would facilitate their incorporation. These assets include the following: (1) each prototype is already packaged in a way that facilitates automated install or embedding in other tools; (2) the prototypes are written in a language (Python) that can already be run by tools such as openLCA or Brightway via their existing interfaces; (3) the tools exports matches and PSM in standard exchange formats already readable by LCA software tools like JSON-LD and ILCD; (4) the open-source licenses permit reuse, embedding, and modification without licensing; and (5) management of the source code for the prototypes in a version control system that supports collaboration and issue tracking (github).

The incorporation of one of more of the prototypes into an LCA tool would make them more accessible to a larger body of practitioners.

5 Conclusion

The PSM automation project demonstrated success in both principle and practice. In principle, this method of challenging experts to solve specific problems by independently and creatively writing code within a set of technical and logistical constraints can be effective in the LCA domain. In practice, the present study generated three operable prototypes that together describe the solution space of the study problem. The process of converting a foreground into a product system model requires four different steps: description of the model structure, determination of flow properties, linking flows to background databases, and serializing the model into a format that can be interpreted by LCA software. The three prototypes, along with the additional *autoprox* tool, showed that experienced LCA

software developers could write software to address, in some cases, all of the steps, in a relatively short period of time. Thus, with greater investment of time into refining or improving these prototypes, or into further development of methods to tackle one of the four steps delineated by this study, is likely to yield very useful product system assembly tools. It is our hope that the community continue to build on the work that is included in this project.

Acknowledgments Bill Michaud, CSRA, managed personnel. Chrys Starr, NAVAIR, provided the primary test dataset. IPC International, Inc. provided the PCB BOM. Rebe Feraldi, LAC Group, and Bill Barrett, USEPA, provided initial reviews. Two anonymous reviewers provided formal reviews and recommendations through Int. J LCA.

Authors' contributions WI conceptualized the study, developed the methodology, and administered the project. BK, CM, and MS conducted the investigation/wrote the software and provided visualization of their respective software. KS and WI acquired funding and resources. BK and WI validated the prototype builds and resulting product systems and were the primary writers. Terms for contributions come from CRediT.

Funding This research was supported by the SERDP-ESTCP research program under project WP-2757 and conducted via EPA contract EP-C-15-02 WA-20 with CSRA.

Code availability The project and prototypes are available as open-source software on the collaborative coding platform GitHub. The resources provided to the developers, including test datasets can be found in the Product System Assembly Resources repository. Each prototype was published independently by its author, and the repositories are publicly available:

antelope: https://github.com/bkuczenski/antelope_epa
pslink: https://github.com/msrocka/pslink
perdu: https://github.com/cmutel/perdu
autoprox: https://github.com/msrocka/autoprox

Declarations

Disclaimer The findings and conclusions in this journal article have not been formally disseminated by the US EPA and should not be construed to represent any agency determination or policy.

References

amesweb.info (2019) Amesweb - Advanced Mechanical Engineering Solutions. amesweb.info

BONSAI (2020) BONSAI. bonsai.uno

Briscoe G, Mulligan C (2014) Digital innovation: The hackathon phenomenon. http://www.creativeworkslondon.org.uk/wp-content/uploads/2013/11/Digital-Innovation-The-Hackathon-Phenomenon1.pdf

Canals LM i, Boureima F, Brago T et al (2016) Towards an interoperable network of LCA databases

Cashman SA, Meyer DE, Edelen A et al (2016) Mining available data from the United States Environmental Protection Agency to support rapid life cycle inventory modeling of chemical manufacturing. Environ Sci Technol 50:9013–9025. https://doi.org/10.1021/acs.est.6b02160

Chaput M (2012) Whoosh. https://whoosh.readthedocs.io/en/latest/index.html

Ciroth A, Srocka M (2017) Introducing the LCA data machine. http://brussels.setac.org/

Davis C, Nikolic I, Dijkema GPJ (2010) Industrial Ecology 2.0. J Ind Ecol 14:707–726. https://doi.org/10.1111/j.1530-9290.2010.00281.x

GreenDelta (2019) openLCA Nexus. https://nexus.openlca.org/

Ingwersen WW (2015) Test of US federal life cycle inventory data interoperability. J Clean Prod 101:118–121. https://doi.org/10.1016/j.jclepro.2015.03.090

Ingwersen WW, Stevenson MJ (2012) Can we compare the environmental performance of this product to that one? An update on the development of product category rules and future challenges toward alignment. J Clean Prod 24:102–108. https://doi.org/10.1016/j.jclepro.2011.10.040

Ingwersen W, Lesage P, Mutel C et al (2015) LCA data interoperability: new solutions to old challenges. Special Session, Vancouver, Canada

Ingwersen WW, Kahn E, Cooper J (2018) Bridge processes: a solution for LCI datasets independent of background databases. Int J Life Cycle Ass 23:2266–2270. https://doi.org/10.1007/s11367-018-1448-6

Joyce PJ (2019) Lca_disclosures. https://github.com/pjamesjoyce/lca_disclosures/tree/typed_flows

Kuczenski B, Beraha S (2015) Antelope a web service for publishing life cycle models and results. In: Proceedings of the International Symposium on Sustainable Systems and Technology

Kuczenski B (2019) Disclosure of product system models in life cycle assessment: achieving transparency and privacy. J Ind Ecol 23:574–586. https://doi.org/10.1111/jiec.12810

Kuczenski B, Davis CB, Rivela B, Janowicz K (2016) Semantic catalogs for life cycle assessment data. J Clean Prod 137:1109–1117. https://doi.org/10.1016/j.jclepro.2016.07.216

Kuczenski B, Marvuglia A, Astudillo MF et al (2018) LCA capability roadmap: product system model description and revision. Int J Life Cycle Ass 23:1685–1692. https://doi.org/10.1007/s11367-018-1446-8

MatWeb, LLC (2020) Matweb. http://matweb.com

McIntosh BS, Koffler C (2014) Using Bill of Materials imports and dynamic EPD templates to minimize the costs of Type III Environmental Product Declarations. https://www.researchgate.net/profile/Christoph_Koffler/publication/280880839_Using_Bill_of_Materials_imports_and_dynamic_EPD_templates_to_minimize_the_costs_of_Type_III_Environmental_Product_Declarations/links/55ca0e0808aebc967dfbd7d5.pdf

Minkov N, Schneider L, Lehmann A, Finkbeiner M (2015) Type III Environmental Declaration Programmes and harmonization of product category rules: Status quo and practical challenges. J Clean Prod 94:235–246. https://doi.org/10.1016/j.jclepro.2015.02.012

Mishra S, Singh SP (2019) Carbon management framework for sustainable manufacturing using life cycle assessment. IoT and carbon sequestration BENCHMARKING ahead-of-print: https://doi.org/10.1108/BIJ-01-2019-0044

Mutel C (2017) Wurst. https://wurst.readthedocs.io/

Pallets Project (2020) Flask. https://palletsprojects.com/p/flask/

PartTarget Inc. (2019) Part Target. http://parttarget.com/

Pawar A, Mago V (2018) Calculating the similarity between words and sentences using a lexical database and corpus statistics. arXiv:180205667 [cs]

Srocka M, Ingwersen, Wesley, Ciroth, Andreas (2020) openLCA-schema. https://github.com/GreenDelta/olca-schema

Suh S, Huppes G (2005) Methods for Life Cycle Inventory of a product. J Clean Prod 13:687-697. https://doi.org/10.1016/j.jclepro.2003.04.001

Suh S, Leighton M, Tomar S, Chen C (2016) Interoperability between ecoinvent ver. 3 and US LCI database: a case study. Int J Life Cycle Ass 21:1290–1298. https://doi.org/10.1007/s11367-013-0592-2

Sundaravaradan N, Marwah M, Shah A, Ramakrishnan N (2011) Data mining approaches for life cycle assessment. In: Proceedings of the 2011 IEEE International Symposium on Sustainable Systems and Technology. pp 1–6

Tao F, Zuo Y, Xu LD et al (2014) Internet of Things and BOM-based life cycle assessment of energy-saving and emission-reduction of products. IEEE Trans Industr Inf 10:1252–1261. https://doi.org/10.1109/TII.2014.2306771

ThoughtCo Inc. (2020) ThoughtCo. https://www.thoughtco.com/

UNEP/SETAC Life Cycle Initiative, (2019) GLAD: Global LCA Data Access network. https://www.globallcadataaccess.org/

USDA (2019) Federal LCA Commons Data Portal. https://www.lcacommons.gov/

Valiente G (2002) Tree isomorphism. algorithms on trees and graphs. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 151–251

Yang Y, Ingwersen WW, Hawkins TR et al (2017) USEEIO: A new and transparent United States environmentally-extended input-output model. J Clean Prod 158:308–318. https://doi.org/10.1016/j.jclepro.2017.04.150